An Expertext System for Building Standards

Andrew Casson[1] and David Stone[2]

Abstract

The paper describes work on a current collaborative research project involving UK Universities and the Scottish Office's Building Directorate. Previous work by the Directorate had identified limitations in a conventional expert systems approach to the development and management of building Standards information. Work in the current project seeks to overcome these limitations by combining representational paradigms. In particular, a system is being designed and implemented which is based on and seeks to extend the concept of Headed Record Expertext [Diaper 90] in which formalised information can be attached, where appropriate, to nodes in a hypertext version of the Standards. The system also handles navigation rules and provides intelligent guidance for the reader through the hyperdocument. The underlying node/arc representational structure is also intended to support the capture of argumentation material generated during the development, maintenance and use of Standards information.

Introduction

This paper reports on a current, collaborative research and development project involving three UK Universities and the Building Directorate of the Scottish Office. In a previous paper Stone & Tweed [Stone 91] reported on the background to the project and some of the research issues which were to be addressed. This paper is in two distinct parts. The first describes the characteristics of the project's application context and discusses system design objectives; the second describes the system architecture of a prototype system and the technical details of its implementation.

The project is application orientated with the Directorate acting as the project's industrial partner and the work of the Directorate's Building Control Division providing the application context. The Directorate is responsible for the drafting and maintenance of the Building Standards Regulations for Scotland and their associated Technical Standards and for the administration of certain of the procedures available under the building control system in Scotland.

All national or state Regulations and Standards would seem to have in common the generic objectives of safeguarding public health and safety and of conserving energy in the built environment. However, there may be significant differences in the ways in which these objectives are translated in different countries into specific technical requirements and in how they are administered in practice. It may be useful, therefore, to describe briefly the project's application context in order to show how its particular characteristics have influenced the research objectives and system design in the project.

The Application Context

In common with all other UK regulations, the Building Regulations for Scotland come into existence because of an Act of Parliament; an Act is to enable, or make operational, a policy

[1]Department of Artificial Intelligence, University of Edinburgh
[2]Building Directorate, The Scottish Office

of government. The relevant Act for the Regulations is the Building (Scotland) Act 1959. This Act enables the Secretary of State for Scotland to prescribe standards by Regulation for a defined set of issues affecting building design and construction.

The document domain

This fundamental statutory framework gives rise to two key documents; the Building Standards (Scotland) Regulations and the Technical Standards. The Regulations document, although it is the primary statutory document, is relatively slight. It contains, in addition to a series of schedules detailing exemptions, classifications of building type, rules of measurement and so forth, a set of *requirements* for the sixteen subject areas covered by the Regulations. These requirements are succinct and are expressed in quite general terms. For example, Regulation 12, covering structural fire precautions simply states:

(1) Every building shall be so constructed that, for a reasonable period, in the event of fire -

(a) its stability is maintained;

(b) the spread of fire and smoke within the building is inhibited; and

(c) the spread of fire to and from other buildings is inhibited.

The Regulations document provides that the requirements of the Regulations can be met by compliance with the *relevant standards* set out in an accompanying Technical Standards document. These relevant standards may in turn be met in two ways:

- by conforming with a provision set out in the Technical Standards and which the Technical Standards document states to be *deemed to satisfy* the relevant standard; (a provision is essentially a model construction detail or specification); or

- by any other means which can be shown to satisfy the relevant standard; (this allows that the deemed to satisfy provisions do not exclude other methods of satisfying the standards).

It will be evident that the Technical Standards document is the primary source document for setting out the standards required or the means by which the requirements of the Regulations can be met. It is, as a consequence, a more substantial document than the Regulations. It contains, in addition to the usual preambles and appendices, sixteen sections, one for each of the subject areas covered by the Regulations and each addressing one or more Regulation requirement. These sections, which make up the bulk of the document, typically consist of an introduction which describes the intent of the part, a verbatim copy of the relevant Regulation clause, the set of qualifying standards and the set of provisions deemed to satisfy the standards.

One critical characteristic of the Standards document, from the point of view of systems design, is that it references a range of other published standards and codes, most usually as part of a deemed to satisfy provision. The Standards currently reference nine different classes of other documents, the most important of which is the British Standards; over 160 British Standards are referenced. This referenced network of other standards is potentially even more extensive than the specific documents cited as a reference to a British Standard has to be construed, because of the UK's membership of the European Community, as a reference to any national technical specification of a member state of the Community which provides an equivalent standard. In addition to the directly referenced documents, authors

of Standards must also take into account other UK legislation which, although not directly concerned with the construction domain, may impinge upon particular issues which the Regulations address; there are, then, extensive implicit and indirect links to other statutory documents in the Standards document.

Processes of change

The Regulations and Standards requirements come into existence or are modified in response to a range of imperatives. The information content of the Standards document is, then, not static but evolves and changes through time. For example, revisions may become necessary simply to correct errors or anomalies in the texts; pressure may develop, either from Government policy or users of the Standards for a change to a requirement or for an entirely new requirement; periodically, major revisions of the text may be necessary when piecemeal revisions are no longer sensible or some major event demands a comprehensive review; and, finally, there may be pressure to harmonise technical requirements across national boundaries. The Technical Standards document itself actively encourages change and it invites users to make suggestions for improving the scope or content of the Regulations. The process of change is managed by way of procedures and controlled by constraints established by statute. The process is essentially consultative and involves advisory committees and discussions with interested and affected bodies. Changes in the Standards and their underlying rationale, then, can be thought of as the end result of a *bargaining* process.

Administrative procedures

In practice, the requirements of the Regulations and Standards apply to any construction, alteration, extension or demolition of a building or part of a building or to any change of use which attracts additional or more onerous requirements. The Scottish building control system, which ensures that the requirements are met, is administered by local government authorities. All proposals for building works have to be submitted to a local authority who have an obligation to check that the proposals comply with the Regulation requirements. If proposals are satisfactory, an applicant is issued with a *building warrant.* During construction, the local authority periodically checks that the work is being carried out in accordance with the warrant drawings and on satisfactory completion a *certificate of completion* is issued. In support of this building control system there are a number of appeals procedures available to applicants. For example, any applicant for a warrant can appeal formally against any decision by a local authority on a warrant application matter. Less formally, where an applicant is of the opinion that he or she cannot reasonably comply with a Regulation requirement, procedures are available for applying for a *relaxation* or *dispensation* of the requirement. There are similar appeals procedures applicable to individual building products and classes of building.

All of these approval and appeal procedures which characterise the building control system are essentially *judicial* in nature. Their exercise requires considerable expertise and necessitates an understanding of the enabling principles of the legislation, a knowledge of the underlying intent of particular Regulations and a detailed knowledge of individual requirements and a familiarity with case history material. The experience and information generated by the administration and application of the Regulations and Standards, then, represents a valuable resource that can provide an important input into the processes of revision and development of Standards information.

Characteristics of the application context

Let us look now at the significant characteristics of the application context and the kind of

problems inherent in authoring and maintaining Standards information. Our starting point is the observation that the Standards field consists essentially of three, interdependent domains of information. The first, and most evident, of these is the domain of published *documents* which contain the formal, procedural and technical requirements and related material. The second, and more elusive domain, is the experiential information and knowledge that is generated by the development, administration and use of the requirements and which provides explanations for and interpretations of the intent, meaning and rationale of the requirements. We have termed this domain of information *argumentation*. In an analysis of similar regulatory contexts, Garzotto & Paolini [Garzotto 89] make the same kind of distinction between documents and argumentation and argue for the binding of what they term *operational* and *domain* knowledge in order to retain and make explicit expertise in the domain and to ensure proper maintenance of the domain's information. The third, and final, domain of information we identify is the set of concepts, principles and models implicit in the Standards' domain but which is largely independent of any particular document or set of requirements. We have termed this domain *meta-knowledge*. Argumentation and meta-knowledge are discussed more fully in System Design below.

It will be clear that this total information domain in the Standards field is potentially very large. As we have seen, the domain of published documents is itself extensive and is not confined to documents from a single source. There are, then, problems of *scale* in the application context which manifest themselves primarily as difficulties of information *organisation* and information *retrieval*. A major difficulty of organisation is the problem of recording and maintaining links between semantically related material. The conventional way of retaining this kind of control of the domain information relies on the use of physical filing systems and individual's memories to maintain the appropriate connections and relationships. However, physical file systems become unwieldy, individual's memories are unreliable or there may be changes in personnel. The collective and individual memory that links information and gives it meaning is, then, only too easily diminished or lost. The problems of information retrieval parallel and are linked to the problems of organisation. For any given task, a user typically needs to abstract only a subset of the material available but needs to be assured that all the relevant information has been located. This problem is particularly critical for authors when drafting or revising requirements. Before proposing changes they need to explore the consequences of change relative to existing material and ensure that their proposals reflect and properly respond to current thinking and do not conflict with established principles.

These problems of scale and information organisation and retrieval are compounded in that the domain information is not static but evolves through time. In the case of the document domain, changes in the short term, may be relatively minor although periodically there may be quite radical revisions of both content and format. Information in the argumentation and meta-knowledge domains is potentially more volatile, at least in an operational context, although analysis suggests that there is a more stable level of consensus at which important issues and the domain's controlling concepts and principles can be made explicit and recorded. This characteristic of *change* in the domain's information does not of itself lead to difficulties for authors other than to compound the problems of organisation and retrieval. It does, however, have implications for the kinds of representational formalisms which can be realistically used in a machine system.

The characterisation of the problems of authoring and maintaining Standards information we have set out here apply equally to end users of the information. Users, whether professionals in practice or administrators of the building control system, must exercise judgement in order properly to interpret and apply requirements and would benefit from access to exemplars, illustrations of principle and other explanatory material. A system

which can capture and represent all the domains of Standards information we have described is likely, then, to be of benefit to authors and users alike.

System design

We shall look now at the kinds of system functions which might address the problems outlined above and provide support for authors and users of Standards. Central to the consideration of system functionality is the issue of *representation.* We may distinguish four main representational paradigms which have, or are, being used for Standards information. These are:

- - hypertext
- - decision logic tables and rules
- - building and product models
- - algorithmic code

Each of these paradigms has its advantages and drawbacks. Hypertext provides more convenient and structured access to texts than conventional linear documents; it does not, however, directly provide functionality with regard to the content of the text nodes and a proliferation of links can lead to difficulties of navigation. Decision logic tables and their equivalent if-then rule sets articulate the logic of requirements and can support consistency checking and inferencing; but they do not cope well with ambiguity in texts and may be difficult to maintain in large application contexts. Product models offer the possibility of providing automated compliance checking and of interfacing CAD systems to Standards information; they are, however, difficult to construct and to be successful may depend upon radical changes in the protocols and procedures of Standards' agencies and authors and innovations in CAD modeling. Finally, conventional algorithmic code is easily implemented but probably has limited application in the present Standards; more promising are simulation systems but to be effective these depend upon a decisive shift to performance Standards.

We may note that whereas hypertext is primarily concerned with providing *access* to texts, the remaining paradigms are concerned with representing the *content* or semantics of the material. This representation of content typically entails one or more transformations or mappings of the source texts into the chosen representation. At the moment, these transformation processes are not automated and involve difficult technical and conceptual issues, not least of which is the problem of ambiguity and the preservation of meaning. Reed [Reed 91] recognises that, whilst work on the representation of the content of Standards is technically challenging and of research interest, progress depends on establishing a common and more rigorous development framework and ultimately on its acceptance by Standard's agencies and the construction industry.

It seems likely, then, that the paradigms directed at representing the content of Standards may be currently impractical in application contexts of any scale or in which the information is subject to change. For this reason we have adopted as the project's primary representational paradigm hypertext or, more precisely, node-and-link structures with the principal objective of supporting the organisation of and access to Standards information.

Expertext

We are proposing to enhance the basic hypertext paradigm along the lines first proposed by

Barlow et al. [Barlow 89]. Their proposals seek to combine the properties of both expert systems and hypertext and Rada & Barlow [Rada 89] subsequently coined the term *expertext* to describe such a system. Rada and Barlow [ibid] argue that both expert systems and hypertext share the same underlying graph-theoretic model. However, they suggest that there are substantial differences in the two approaches in their treatment of nodes and links. Whereas the nodes in hypertext are semantically rich, in that they contain natural language texts, the links have little or no semantic content. Conversely, whereas the nodes in an expert system are semantically impoverished, because they contain a formalisation of the source information, the links are well specified, typically as predicate names in a rule-based system. Expertext represents an attempt to combine the best properties of expert systems and hypertext and provide systems which have the semantically rich nodes of hypertext and the well specified, computable links of expert systems.

It is interesting to look at this proposal from the perspective of how intelligence is distributed between the user and the system. In an expert system the intelligence lies primarily with the system; the user is usually relegated to responding to system generated queries. In hypertext, by contrast, the intelligence lies largely with the user, who interprets the node content but must also specify the order of link traversal. In expertext there is potentially a more balanced distribution of intelligence. Understanding and interpreting the content of nodes is the user's responsibility and is the task most appropriate to human intelligence. On the other hand, because in the expertext model there exists defined and computable links between nodes, the system's intelligence can be focussed on the task of navigation, that is on selecting and presenting the most appropriate set of nodes for the user's consideration. It is argued that this distribution of intelligence should reduce if not eliminate the navigation problems conventionally associated with hypertext.

The expertext model potentially fits well with the perceived problems of the application domain. Firstly, the problems of the organisation of material and the maintenance of links across domain information can be resolved by expertext's underlying node-and-link structure. Secondly, the problems of information retrieval can be resolved by expertext's intelligent navigation functionality. Thirdly, expertext is not inherently limited by the problems of scale or change in the domain's information. And finally, it seems possible that the functionality and intelligence of the system can be incrementally enhanced as experience in its use is gained.

A particular architecture for expertext was proposed by Barlow et al. [Barlow 89]. This has subsequently been developed and termed Headed Record Expertext (HRExpertext) by Diaper and Rada [Diaper 89] and Diaper and Beer [Diaper 90]. The underlying model of HRExpertext remains the node-and-link semantic net. Nodes, however, consist of *headed records* which have two parts. These are, a *record* containing the user readable, natural language texts (or potentially any form of user understandable material) and a *header* which contains an abstraction of the semantic content of the record. Header material is a formalisation of the record's content and is, therefore, an impoverished representation of the record but it has the property that it can be used computationally by the system to select and make available the records consistent with a user's objectives. It is this HRExpertext model which is being implemented in the project and which is described below.

Information domains

It is proposed that three distinct but inter-related domains of information will be represented in the system. These three domains, identified earlier in the section on Characteristics of the application context, are:

- - documentation
- - argumentation
- - meta-knowledge

The nature of the documentation domain is largely self-evident and will not be elaborated here except to note that documents will be decomposed into a set of expertext *fragments* and associated semantic *links*.

Argumentation

The argumentation domain embraces the experiential and operational information generated by the administration and application of the Standards and which, we have argued, is essential for their proper understanding and development. This information may be well documented but is dispersed and held as minutes and records of meetings, notes of discussions, formal records of appeal determinations, written answers to queries and correspondence, research reports and so forth; or it may simply be anecdotal and located in an individual's experience. A major, and continuing, research problem in the project is to find an appropriate *rhetoric model* for organising and structuring this kind of information and binding it to the documentation and meta-knowledge domains. What we require is a method of organising texts which allows a user to expose key relationships within argumentation material and to make explicit the inherent processes of dialogue and negotiation. We have to date identified two candidate models: Issue Based Information Systems (IBIS) [Kunz 70][Conklin 88] and Rhetorical Structure Theory (RST) [Mann 87][Mann 89]].

The IBIS model suggests that we can characterise argumentation material as being the end-product of a process of dialogue or debate, involving a number of participants, about one or more *issues*. An issue in this sense is simply a question or unresolved problem. IBIS provides a framework for structuring and recording the elements of information generated by this process of issue resolution in a node-and-link representation. IBIS offers a relatively restricted set of node and link types. For example, nodes may be *issues*, *positions* or *arguments* where positions *respond_to* issues and arguments *support* or *object_to* a position.

RST, by contrast, is not primarily concerned with dialogue but with providing a way of analysing and structuring texts as written *monologue*. It works by splitting the text into fragments of any length and then linking these fragments to form a hierarchy. The links between fragments are commonly of a nucleus-satellite form, in which one node is ancillary to the other. An important difference between the RST and IBIS approach is that nodes in RST can be *groups* of sub-nodes. This allows links to be placed between large sections of the text, each of which may themselves contain a network of links and sub-nodes. RST provides a more sophisticated representation than IBIS, both in terms of the number of different relations and the definition of restrictions on how the relations should be applied. Typical relation types in an RST graph might be *background, concession, contrast, elaboration, motivation* or *means*.

IBIS and RST are to some extent complimentary and some hybrid model might be desirable. For example, it may be possible to merge the different types of relations to provide a single set of relations with a rich variety of types; or to use IBIS to map out the dialogue structure of the argumentation material and then use RST relations to map out the fine detail of the issues, arguments and so forth. However, it seems likely that there is no abstract way of resolving the issue of modeling the information in the argumentation domain and that it can only be resolved by pragmatic experimentation in the application context. There are no technical obstacles to this as the underlying node-and-link kernel of the prototype system admits nodes and links of any type.

Meta-knowledge

The meta-knowledge domain embraces information we wish to see represented in the system but which is either only implicit in the domain documents or which is explicit but is dispersed and un-structured. It is information which is *about* Standards but which is largely independent of any particular formulation or expression of a set of requirements. This meta-knowledge we need to formalise and represent is largely given and defined by the nature of the application domain. Building Standards, regardless of any particular format or content, are, in essence, always a response to some perceived *risk* or dysfunction in buildings or their systems and have the intention of minimising or eliminating such risks by assigning required *properties* to appropriate *elements* of buildings and their spaces. This trilogy of risk (implicitly the *intent* of the Standards), elements and properties provides the essential basis for the layers of meta-knowledge required. We propose three meta-knowledge layers for the system; a *concept network* for modeling the intent of the Standards; a stereotypical *building model* for representing building elements, their possible relationships and attributes; and *classification hierarchies* for organising elements and properties. Other researchers have similarly proposed the need for classification systems for Standards [Vanier 91], for reference models of the objects that are the subject of Standards [Cornick 91] and for a layered representation of Standards and related material [Turk 91].

The primary purpose of the meta-knowledge layers is to add functionality to browsing and navigation. The user will be able to stipulate, either by way of menus or graphical browsers, the concepts which the text fragments must contain, thus focusing on a particular area of interest. The meta-knowledge layers will also enable the system to alert the user to the existence of other fragments relevant to the original query but not explicitly referenced from the currently activated fragment set and to suggest to authors appropriate locations for placing new material.

The Prototype System

We shall now look at how some of these system design issues have been addressed and implemented in a prototype system.

System Overview

The overall structure of the prototype system is shown in figure 1. It has a modular, layered architecture, where each component is implemented in terms of the one immediately below it, but is completely independent of the ones above it. There is no direct communication between CONNEKT and user-tools like TOME, neither is there any between the user-tools themselves; all interaction is through the Expertext Shell, letting it take care of informing each tool of what the other is doing. This helps to ensure that the low-level components are as general-purpose as possible and not restricted to any particular

higher-level application or domain. As a result the components are much easier to maintain, and (especially CONNEKT) are available for use in completely different systems.

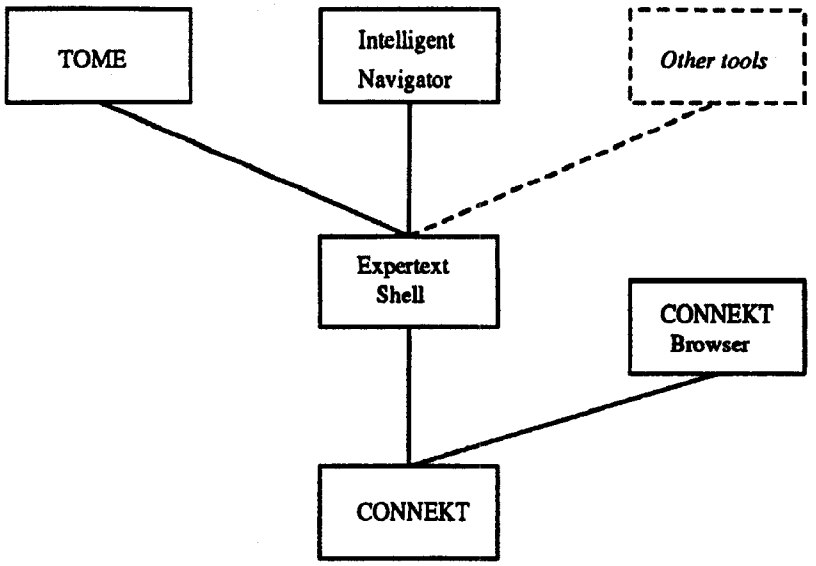

Figure 1: System Architecture

The system runs on Sun SPARCstations and is implemented entirely in Pop11 (an AI language similar in functionality and power, though not in syntax, to Common Lisp) in the POPLOG[3] programming environment. The user-interface components use the POPLOG interface to the Sun OpenWindows X Server and the OPEN LOOK[4] Intrinsics Toolkit Widget Set.

Much, but not all, of what is described below is already implemented. The majority of the work left to do is on the Expertext Shell rule interpreter and the Intelligent Navigator.

The Constrained Network Tool (CONNEKT)

Basic functionality

CONNEKT knows nothing about text, hypertext or rules. It simply provides facilities for constructing, constraining, and manipulating multiple nets of nodes and arcs. Nodes, arcs, and nets all have types, and nodes and arcs can have arbitrary information associated with them in named slots.

It is possible in CONNEKT to place an arc between two nodes that are in different nets. This allows nets for different types of application, which must conform to local constraints and configurations, to function together in a unified system. For example, a node in a hypertext net representing a standards document can be linked to a node in an IBIS net representing the issue that underpins it, and to nodes in classification structure nets which define and index its contents.

[3]POPLOG is a registered trademark of the University of Sussex
[4]OPEN LOOK is a registered trademark of AT&T

Each net type T is defined by a Net Consistency Model (NCM)[5] file, T.ncm. This lists the node and arc types that can be used in the net, along with their syntactic and semantic properties. An example NCM for the "IBIS" net type is shown in figure 2. Each node and arc specification starts with the type name, and is followed by zero or more named property fields each starting with a '+'. The *includencm* directive splices in node and arc specifications from the NCMs of other net types, so that common types can be shared.

```
includencm {general};

nodetypes
        {       issue       +ETSdtype {presentation, reference},
                position    +ETSdtype {presentation, reference},
                argument    +ETSdtype {presentation, reference},
                comment     +ETSdtype {presentation, reference}
        };

arctypes
        {       specialises       +map = +dep none,
                replaces  +map =  +dep none,
                contradicts       +map = +dep none,
                responds_to       +map position -> issue +dep backward,
                supports          +map argument -> position     +dep backward,
                questions         +map argument -> position     +dep backward,
                objects_to        +map argument -> position     +dep backward,
                suggests          +map * -> issue +dep none,
                comments_on       +map comment -> *     +dep none,
                underlies         +map issue -> TS.section      +dep none
        };
```

Figure 2: An NCM for an IBIS net

The *+map* field (the arc "mapping") defines which types of nodes can be connected by the arc type. The basic form is *n1.t1 -> n2.t2* meaning that the arc connects a node of type *t1* in a net of type *n1* to a node of type *t2* in a net of type *n2* . The net type specification is optional (and generally omitted) and defaults to the net type which the NCM defines. A `*' indicates any type, and `=' means the arc connects nodes of the same type. For example[6] the *underlies* type, which connects an *issue* node in an IBIS net to a *section* node in a Technical Standards (TS) net, has the mapping *issue -> TS.section*\ in the IBIS NCM.

The *+dep* field defines the arc "dependency". When an arc is deleted from a net, it may be appropriate to automatically delete either or both of the nodes it connects. In fact what usually happens is that a node is deleted, automatically deleting all arcs to and from it (since arcs can't exist without their nodes) and when these arcs go they take some more nodes with them. For example, in an IBIS network, when a *position* is deleted, all *supports* arcs to it will go too, and because these have "backward" dependency, meaning that when the arc is deleted the node it connects *from* is also deleted, all the connected *argument*

[5]the NCM grew out of the IBIS [Kunz 70] notion of a rhetoric model, which defines the relations allowed between each type of IBIS node.

[6]Remember that although this example is clearly in the hypertext/IBIS domain, there is no notion of such an application in the NCM; it simply defines a bunch of node and arc types.

nodes will disappear. Other values for the *+dep*, are "forward" (the *to* node is dependent on the arc), "mutual" (both connected nodes dependent, and deleted with the arc) and "none". The dependency information can be used to ensure that certain portions of the net that never become isolated, as they would be meaningless without any relations to anything else.

Only the *map* and *+dep* fields are built in to CONNEKT, but applications can define new ones quite easily (as indeed the Expertext Shell does), and access their values with the **getncmfield** procedure. A full list of CONNEKT procedures for creating, destroying and manipulating nets, NCMs, nodes, and arcs can be found in Appendix 1.

A graphical interface to CONNEKT is provided, in which each net has a separate window on the screen, and nodes and arcs are represented as boxes and arrowed lines with labels. Examples are shown in figure 3. Different node and arc types can be displayed in different colours and shapes; the browser currently defines *+color* and *nolabel* fields which can be used to specify the details in the NCM. Nodes and arcs can be created using popup menus and the mouse, and moved around the scrollable layout pane freely. Nodes created by application programs (such as TOME) go into a queue to await placement by the user.[7] Arcs between nets can be created simply by clicking and dragging the mouse pointer between windows (as is being done in the figure). Once created, they appear as links to net icons in each window (also shown in the figure).

Facilities are included for applications to highlight nodes and arcs, have certain ones (individual nodes, all arcs of a given type, and so on) visible, invisible or highlighted, and to move them around the layout pane. In our own application these are chiefly used by the Intelligent Navigator (see Intelligent Navigator below).

The Expertext Shell (ETS)

The ETS is built on top of CONNEKT, and uses it to implement hypertext documents of text fragments and links. Each document is a net, each fragment a node, and each link an arc. It also uses CONNEKT for the representation of classification structures, as described in Classification nets below.

Documents

Each document net has six associated data fields.[8]

- *title*: the full name of the document, for example 'Part N: Electrical Installations'
- *ref*: a short reference name, for example "PartN", for use by the **select** and **deselect** procedures below
- *file*: the UNIX pathname of the file containing the ASCII text of the document
- *buffer*: a list of strings containing the document text line by line

---

[7]There is at present no automatic layout algorithm, but hooks exist in the code for later addition

[8]The descriptions here and below apply only to text documents; graphical documents will have a slightly different representation.

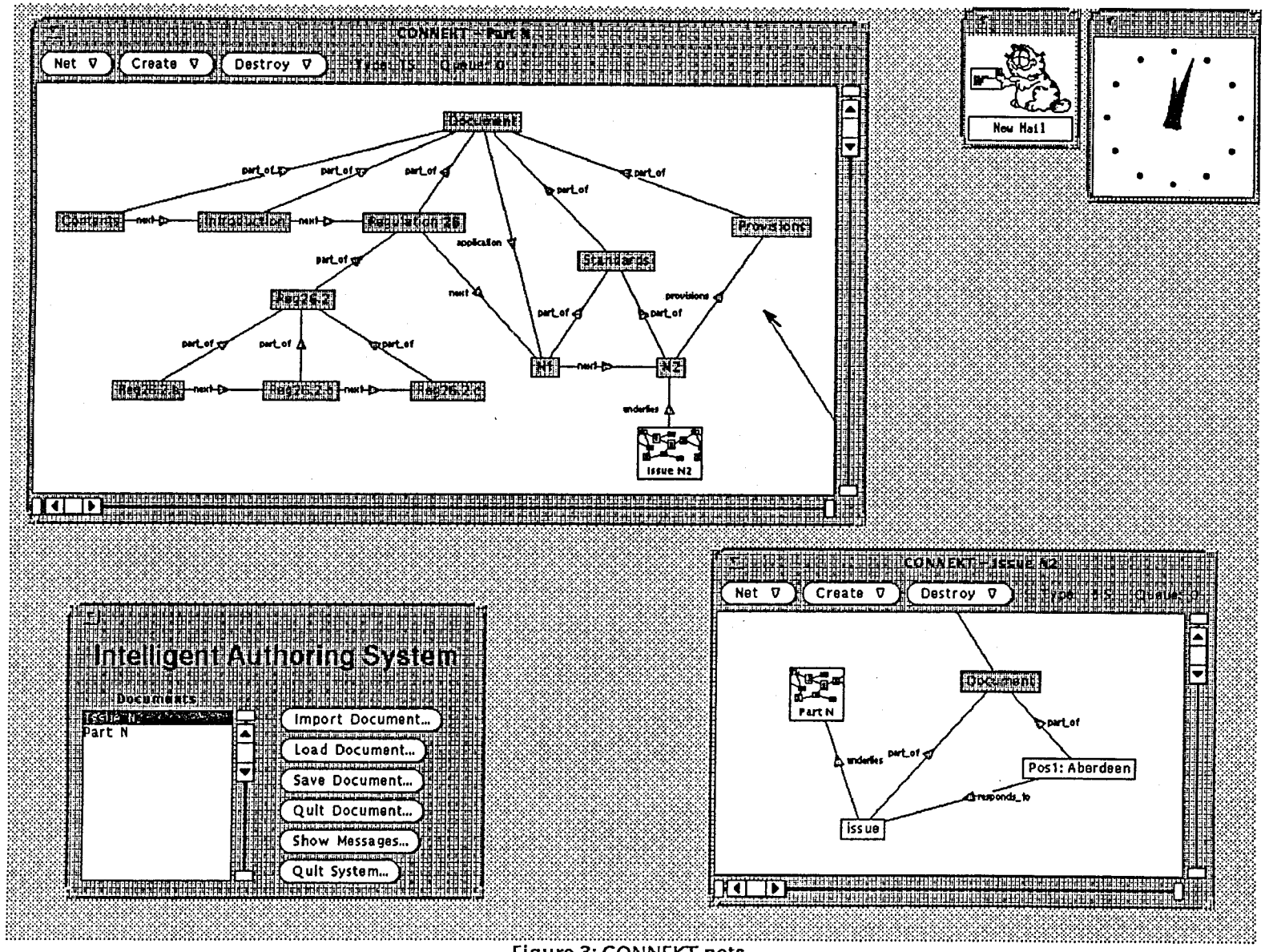

Figure 3: CONNEKT nets

- *style*: a structure defining the emboldened and italicized portions of the text

- *document-fragment*: an initial text fragment containing the entire document text; the ETS defines a *+ETSdoc* field (with no arguments) whereby one of the node types in the document NCM can be marked as the type to be used for the document fragment node.

Fragments and links

Following the Headed Record Expertext model, each fragment node is given two special slots: the *header* in which formalised knowledge about the fragment is stored, and the *record* which identifies the source knowledge about it, i.e. the text itself. Each link arc also has a *header* slot (though of course no record).

The record structure has two fields:

- *document*: a pointer to the document net (as above)

- *span*: a structure specifying the *(column, row)* co-ordinates of the start and end points of the fragment in the document

In our application the values of these fields are provided automatically by TOME (see below) when a fragment is marked up.

The header formally describes the link or fragment (including the text identified by the fragment record). Headers are of variable length and can be directly manipulated by the user (for example through TOME), to change field values, add new fields and delete fields. Currently, fragments and links have the following common header fields by default:

- *creator*: the name of the user who created the fragment of link (including both new empty fragments and those marked up from an existing document)

- *date*: the date and time of creation

- *content-type*: the semantic class of the fragment or link; this is the same as the type of the item's underlying CONNEKT node or arc, but it is duplicated for ease of access

In addition, fragment headers currently have these fields:

- *title*: the full name of the fragment, for example 'Regulation 26: Electrical Installations'

- *ref*: a short name for the fragment, for example "Regulation26", for use by the **select** and **deselect** procedures (see below)

- *author*: the user who wrote the fragment text (cf. *creator*)

- *display-type*: one of the keywords "presentation", "reference" or "button", describing the appearance and behaviour of the fragment (see TOME below); the ETS defines a *+ETSdtype* field to be used in NCMs to define the allowable display types for each content type (see figure 2 above)

- *rules*: a set of navigation/formalisation production rules local to the fragment (see Rules and the interpreter below).

Note that the links to and from the fragment are arcs represented at the CONNEKT level, not in the fragment header.

It will also be necessary to store version and status information in fragments, in order for TOME to keep track of changing documents. This includes information on the completeness of both the text itself, and the rest of the header and the fragment links, for example to indicate that the classification (see Classification nets) is as yet only partial.

ETS procedures: select and deselect

A full list of current ETS procedures for creating and maintaining expertext networks is given in Appendix 2. Two however deserve special mention here, as they are crucial to the use of the ETS rule interpreter (described in detail in the next section). These are **select** and **deselect** which each do two main things with a list of specified fragments:

- they inform the user-tools (TOME and the Intelligent Navigator) that the fragments should be made current/non-current respectively; this will be interpreted by different tools in different ways

- they make the rules local to the fragments and all their sub-fragments (defined by *part-of* links) available/unavailable for firing by the rule interpreter.

In addition, **select** adds the specified fragments to an ordered list of fragments visited so far (the *path*. The syntax for the procedures is fairly complex. The basic form is:

select *{<fgspec-1>, <fgspec-2>, ... <fgspec-n>\fB}*

(similarly for **deselect**) where each *fgspec* can be any of the following forms:

- *fgref* --- a fragment *ref*, for fragments in the current document .

- *docref* : *fgref* --- for fragments in other documents

- *fgspec* $ df --- the *document-fragment* (see Documents above) for the given fragment

- *fgspec* $ path --- the fragments selected so far, up to the given fragment

- *fgspec* -> *linkname* --- all fragments connected by the specified link type *from* the given fragment

- *fgspec* <- *linkname* --- all fragments connected by the specified link type *to* the given fragment

In the last four of these, the *fgspec* can be omitted and defaults to the current (most recently selected) fragment. Also, when called with no fragment list **deselect** operates on the current fragment.

For example,

deselect;

select {PartN:Regulation26 <- underlies $ df};

might be used in a rule to close the current fragment and open the document fragment containing the IBIS issue underlying Regulation 26 in Part N.

Classification nets

The ETS uses CONNEKT nets to represent the concept and classification structures which encode meta-knowledge about documents (as discussed in System design above). For example, classifiers (i.e. classification nodes) of type *element* and *property*, instances of which might be *dwelling, conservatory, is-glazed* and *has-access*, might be connected by arcs of type *example-of, applies-to, requires* and *excludes*, which represent the relations and interactions between the classifiers. Each ETS text fragment can then be classified simply by creating ordinary CONNEKT arcs between the fragment node and a selection of classifiers, and consequently, finding fragments defined by user-specified classifiers becomes a network navigation problem which can be handled using the same rule-based approach as used for fragment to fragment navigation (see below).

Rules and the rule interpreter

The ETS provides a forward-chaining interpreter for production rules which can be used for both document navigation and text formalisation. These rules can be either *global*, that is, stored in the ETS central rule base, or *local*, meaning that they are bound to particular fragments as detailed above. Navigation rules, which often apply to classes of fragments rather than individual fragments[9] will usually be global. Text formalisation rules will almost always be local. However, the distinction between the two types of rule is quite hazy, as we shall see. The interpreter has the usual *Working Memory* (WM) area in which partial results and input data are stored during computation.

The basic rule syntax is:

rule <rule-number> <WM-patterns>\fB;
    \fI<Pop11 code>\fR
endrule;

where <*WM-patterns*> consists of zero or more patterns that must match with current WM assertions for the rule body to be run. The fact that the body consists of arbitrary Pop11 code means that rules automatically have full access to all the CONNEKT and ETS procedures (see Appendices), including **select** and **deselect**, and that ordinary algorithmic code (for complex mathematical calculations and so on) can be freely mixed with rules.

The ETS rule interpreter uses several special conflict resolution strategies when more than one rule's WM patterns match, in addition to the usual *recency, specificity, refractoriness,*

---

[9]e.g. `if the current fragment is of type *position*, and the user clicks on the "Next" button, then select all other *position* fragments that *respond-to* the same *issue* that haven't yet been visited.'

and so on. These are as follows:

- *parochiality*: fire more local rules before less local rules and global rules
- *fragment-recency*: fire rules for most recently activated fragments first
- *fragment-order*: fire rules for fragment F1 before rules for fragment F2 if there is a *next* link from F1 to F2.

The query translator

The firing of rules is directed by the WM patterns. To specify the initial state of the WM, the ETS provides an interpreter through which users can specifying their interests in a flexible, user-friendly query language. The input query strings are then translated into internal WM assertions.

Most queries will be fairly simple (for example "next") but complex Boolean expressions are handled, as these are necessary when accessing fragments using classification structures, for example "all fragments not in Part E about fire safety and stairs".

See the section on the Intelligent Navigator for more on user-input of queries.

A worked example of navigation and formalisation rules

Appendix 3 contains the complete text of Part N of the Technical Standards for the Building Standards (Scotland) Regulations 1990, the subject of which is electrical installations. This is by far the shortest Part of the Technical Standards,[10] but it exhibits many of the interesting features (from a text formalisation point of view) that are found in the Standards, and is thus ideal as an example text here. The network for Part N is the top one in Figure 3.

Figure 4 contains a set of rules for Part N, plus a couple of global rules. Let us suppose that a user is currently reading some fragment in Part N, trying to find the requirements that apply to a particular electrical installation, and decides to let the system do the work. Suppose also that clicking on the "Application" button in the Intelligent Navigator asserts the pattern [button Application] into the otherwise empty WM, and then starts the rule interpreter.

The only rule to match will be the global rule 001, which recognises that the "Application" button has been clicked, and selects the fragment containing the application information for the current document, by following the application link from its document fragment. Rule 001 also modifies the WM to indicate the function that is to be performed.

---

[10] 5 sparse pages in the original, compared with 44 pages for Part E, `Means of Escape from Fire and Facilities for Fire-Fighting'.

```
/* fragment Regulation26.2 */
rule 101 [fn apply];
     deselect

/* fragment Regulation26.2.a */
rule 102 [fn apply];
     if ynask ('Does the installation serve a buil...?') then
          deselect {->partof, N2};
     endif
endrule;

/* fragment Regulation26.2.b */
rule 103 [fn apply];
     if ynask ('Does the installation form part of ...?') then
          deselect {->partof, N2};
     endif
endrule;

/* fragment Regulation26.2.c */
rule 104 [fn apply];
     if ynask ('Does the installation consist of a ...?') then
          deselect {->partof, N2};
     endif
endrule;

/* fragment N1 */
rule 105 [fn apply];
     select {N2, Regulation26.2}
endrule;

/* fragment N2 */
rule 106 [fn apply];
     wmdelete [fn apply]
endrule;
```

Figure 4 Rules for Part N

From the network we can see that the fragment selected by 001 is N1. The local rule 105 thus becomes active. The text that this rule formalises is:

> N1.2 The standards apply to electrical installations in, or serving, all buildings,except -installations specified in regulation 26(2).

This common form, `the standards apply, except...' can be formalised by first selecting the standards (fragment N2), and then selecting the rules for the exception cases, which are in sub-fragments of fragment Regulation26.2[11]. The *fragment-recency* policy chooses the exception rules over the standards, so rules 101---104 are now active. The *parochiality*

---

[11]Note that first selecting N2 and then conditionally deselecting it isn't the only valid method (rule 101 could select it after the exceptions have first been checked) but it does reflect the structure of the original text fairly well.

policy then chooses the most local rules 102, 103 and 104 before 101, and by *fragment-order* (see the *next* links in figure 3) rule 102 is run first, and it asks the user a question, related to the text of clause 26.2.a. If the answer is `yes', an exception has been found so the rule deselects the fragment that Regulation26.2.a is part of, i.e. Regulation26.2, thereby deselecting all of the latter's other sub-fragments, which contain rules 101, 103 and 104, as only one exception condition is needed. Rule 102 also deselects the standards in N2 because they are no longer applicable. No rules are left to fire, and processing stops.

If the answer to the question in 102 is `no', control proceeds to 103, and similarly to 104. If no exceptions are found then rule 101 fires, deselecting Regulation26.2 and leaving only fragment N2 selected.

If N2 were more complicated, there would be lots more rules to fire. However, there comes a point where any more formalisation is pointless: the text of N2 is quite clear as it is, and the user will be left looking at it. The only rule for N2, 106, simply deletes the [fn apply] pattern from the WM, effectively saying `I'm done'.

However, the global rule 002 can now come into play. There is a *provisions* link from N2 to the `Deemed to Satisfy' provisions for N2.1, and rule 002 ensures that if the user presses the "Provisions" button then the relevant fragment is presented.

Note that the questions for each of the rules 102, 103 and 104 could simply be `Is the installation described by the highlighted text?', since TOME will highlight the current reference fragment (see below). In other situations it might be possible to ask a succinct question much shorter than the original text, but the user would, if necessary, be able to see from the text exactly why it was being asked, effectively eliminating the need for the rule interpreter to actually generate answers to `Why do you want to know?' questions.

All of the rules in this example are concerned only with movement between text fragments. However, rules may also be written for navigation through the classification nets to which these can be linked (see Classification nets above). User queries can then specify particular classifiers, which are taken as starting points from which the rules work through the classification nets, taking account of complex relations between classifiers, until text fragments are eventually reached and selected.

Special benefits of the ETS rule interpreter

- The user only has to formalise the sections that need it. The emphasis is on finding and presenting relevant sections of text which are then fairly easy to understand and not worthy of further logical analysis.

- There is no need to draw a distinction between text formalisation and document navigation. Rules of each type ---if they can be assigned to one or the other ---work together seamlessly.

- Fragment-local rules are modular. They can easily be added or modified incrementally as small text sections are authored or revised.

- Fragment-local rules provide extra support for checking document integrity: when a text is revised, analysis of the fragment's existing rules (along with its links and header fields) can help to reveal the possible effects on consistency with other fragments.

- Fragment-local rules support good understanding by users of the line of reasoning

the rule interpreter is pursuing when it asks for input. Looking at the original piece of text is often much preferable to trying to understand a computer-generated "explanation".

The User Tools

To finish we will take a brief look at the functions of two user tools currently being implemented on top of the ETS.

The Text Object Markup Editor (TOME)

TOME is primarily a tool enabling the author to convert linear documents into expertext documents. It displays ETS fragments with different *display-types* (see Fragments and links above) in different ways. Each "presentation" fragment has its own window on the screen, displaying the text and a panel of control menus. The text of presentation fragments should make sense in isolation. "Reference" fragments, whose texts are generally much shorter and need the context of surrounding text to make sense, even though the text cross-references them from elsewhere, appear as regions with a light grey background within a presentation fragment. "Button" fragments, which are single words or phrases in the text acting as cross-references to presentation or reference fragments, or terms with definitions in a glossary, appear red and emboldened.

Starting from the document fragment produced when a linear ASCII document is "imported" (see the **ETSImportDocument** procedure in Appendix 2), new fragments are created by marking out a section of text of a current fragment, and selecting a display type and a content type from the menus. A *part-of* link is automatically created from the new fragment to the old. As this process is repeated, a hierarchically structured hyperdocument is built up from the original linear text. Using the CONNEKT browser, other links (for example *next*) can be made between the fragments thus created (see figure 5).

TOME provides facilities for changing the style of portions of the text, for example emboldening and underlining. It is also possible to also edit the headers and the text of fragments, and create new, empty fragments to augment a document. Finally, for the author, TOME includes a rule editor with which rules can entered and bound to fragments.

TOME windows (with their editing facilities disabled) are also used by the Intelligent Navigator for displaying fragments to the reader. A presentation fragment is selected by opening its TOME window. A selected reference fragment is highlighted within its window (which may have to be opened first).

Intelligent Navigator

The Navigator appears to users as a window containing a text field for the input of queries which define the text fragments they wish to view (see The query translator above). When a query is typed and the RETURN key pressed, the Navigator just passes the query string to the ETS to parse into WM assertions, and then starts the rules.

The window also contains a set of buttons, each of which has a common query associated with it. When the button is clicked, the query is processed and the rules started as for textual input. Currently the buttons defined are "Next", "Previous", "Application", "Provisions" and "Issue".

As well as using TOME, the Navigator uses the CONNEKT browser to display the net or nets currently being explored. Selected nodes are visible, deselected ones are not. The

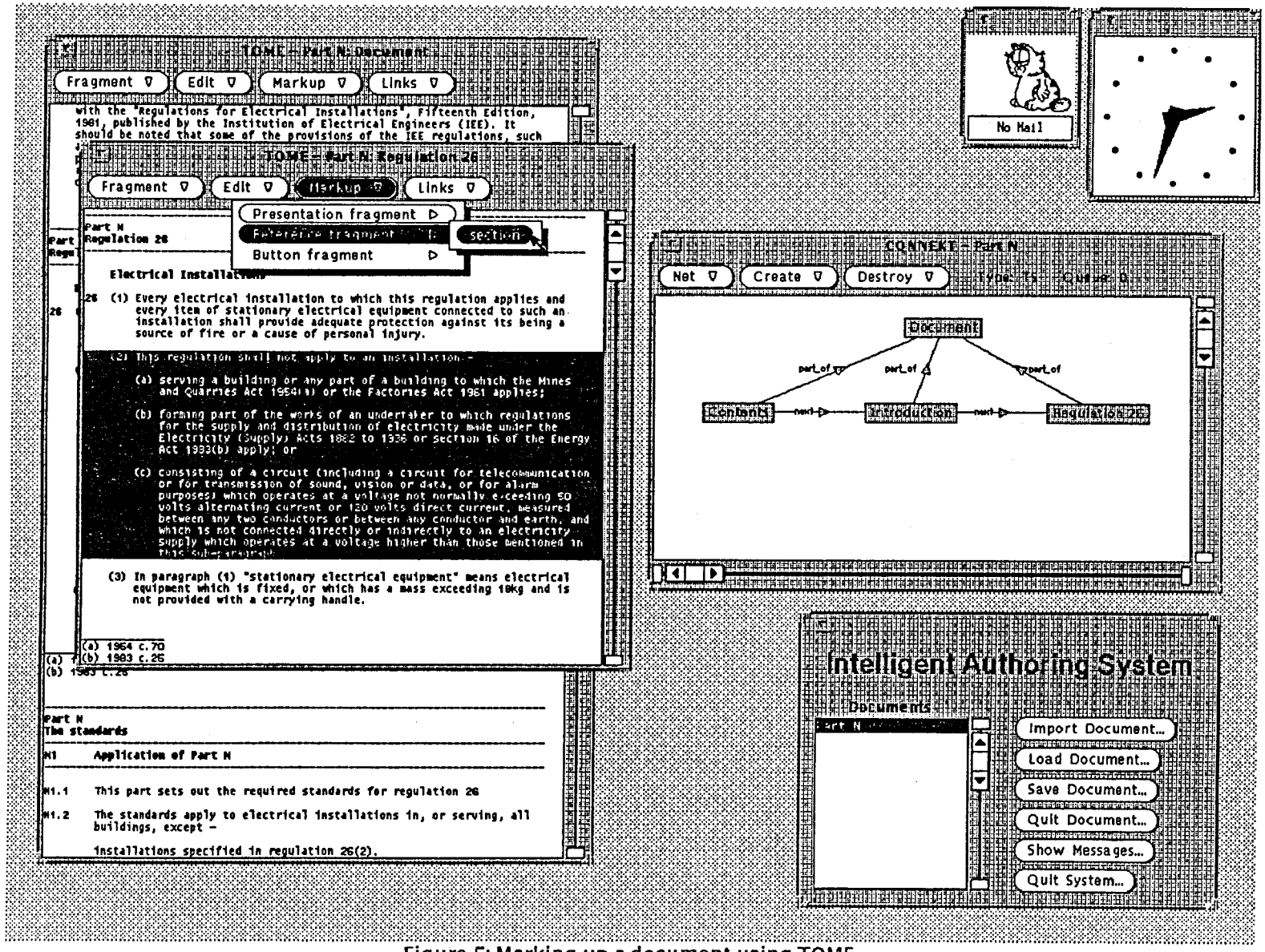

Figure 5: Marking up a document using TOME

current node is highlighted.

Summary

The paper has described the application context, system design objectives and progress on implementation of a prototype system intended to support the authoring and use of Building Standards information. Significant characteristics identified in the application context are multiple domains of information and problems associated with scale and change and it has been argued that consequent user problems centre primarily on difficulties of information organisation and retrieval. An enhancement of the hypertext paradigm, termed expertext, has been proposed as the primary representational model for the application context's domains of information. Technical aspects of a prototype system have been described. System components are implemented in a modular, layered architecture with kernel facilities for constructing, constraining and maintaining multiple nets of nodes and arcs and traversing these nets in an intelligent manner using rules that can be associated with individual nodes.

Acknowledgements

Partners in the collaborative research described in this paper are the Scottish Office's Building Directorate, the Department of Architecture and Planning of the Queen's University of Belfast, the Department of Artificial Intelligence, the University of Edinburgh and the Department of Civil Engineering of the University of Leeds. The academic partners in the project are supported by a grant awarded under the Information Technology Applications Initiative of the UK's Science and Engineering Research Council.

References

[Barlow 89] Barlow J, Beer M, Bench-Capon T, Diaper D, Dunne P E S & Rada R , "Expertext: Hypertext-Expert System Theory, Synergy and Potential Applications"; in Shadbolt N, (Ed.) *Research and Development in Expert Systems IV*; pp. 116-127; Cambridge University Press, Cambridge UK, 1989

[Conklin 88] Conklin J & Begeman M L , "IBIS : A Hypertext Tool for Exploratory Policy Discussion", *ACM Transactions on Office Information Systems*, 6(4); pp. 303-331, 1988

[Cornick 91] Cornick S, Leishman D & Thomas R , "Integrating Building Codes into Design Systems", in *Proceedings of VTT Symposium 125 Computers and Building Regulations*, Espoo, Finland, pp.210-236, 1991

[Diaper 89] Diaper D & Rada R , "Expertext : Hyperising Expert Systems and Expertising Hypertext", in *Proceedings of Hypermedia/Hypertext and Object Orientated Databases*; pp. 124-152;Unicom Seminars Ltd; Brunel University, 1989

[Diaper 90] Diaper D & Beer M , "Headed Record Expertext and Document Applications", in *Proceedings of Hypertext Update*; pp. 62-69; Unicom Seminars Ltd., London, 1990

[Garzotto 89] Garzotto F & Paolini P , "Expert Dictionaries: Knowledge-based Tools for Explanation and Maintenance of Complex Application

Environments"; in *Proceedings of the Second international Conference on Industrial & Engineering Applications of AI & Expert Systems IEA/AIE-89*; ACM Press, pp. 126-134, 1989

[Kunz 70] Kunz W. & Rittel H , *Issues as Elements of Information Systems*; Report S-78-2, Institut für Grundlagen der Planung, Universität Stuttgart, 1970

[Mann 87] Mann W C & Thompson S A , Rhetorical Structure Theory : *A Framework for the Analysis of Texts*, Isi/rs-87-185, Information Sciences Institute, University of Southern California, 1987

[Mann 89] Mann W C, Mathiesson C M I M & Thomson S A , *Rhetorical Structure Theory and Text Analysis*, Isi/rr-89-242, Information Sciences Institute, University of Southern California, 1989

[Rada 89] Rada R & Barlow J , "Expertext : Expert Systems and Hypertext", in *Proceedings of EXSYS'89 Conference*, IITT-International, Paris, 1989

[Reed 91] Reed K A , "Building Regulations and Standards in the Computer Age", in *Proceedings of VTT Symposium 125 Computers and Building Regulations*, Espoo, Finland; pp.12-16, 1991

[Stone 91] Stone D & Tweed C , "Representation Issues in the Design of an Intelligent Information System for Building Standards", in *Proceedings of VTT Symposium 125 Computers and Building Regulations*, Espoo, Finland; pp. 237-249,1991

[Turk 91] Turk Z , "Building Model Standard as a Foundation for Computer Integrated Design", in *Proceedings of VTT Symposium 125 Computers and Building Regulations*, Espoo, Finland, pp. 181-194, 1991

[Vanier 91] Vanier D J , "A Parsimonious Classification System to Extract Project-Specific Building Codes"; in *Proceedings of VTT Symposium 125 Computers and Building Regulations*, Espoo, Finland; pp. 134-145, 1991

Appendix 1: CONNEKT procedures

N.B.: In addition to carrying out the functions described, each of the following procedures informs all current application and tools using CONNEKT of what it has just done. This means when one tool calls a CONNEKT procedure all the others know and can respond accordingly to keep the entire system up-to-date, without the need for direct communication between each pair of tools.

createnet(*type, title*) —> *net*
Creates and returns a new net of the given type and title. Uses the NCM in the file $IADS/*type*.ncm.

destroynet(*net*)
Destroys the net and all its nodes and arcs.

createnode(*net, type*) —> *node*
Creates and returns a node of the given type in the given net. Raises an error if the

type is not a node type in the NCM for the net.

createarc(*from-node, to-node, type*) —> *arc*
Creates and returns an arc of the given type between the two given nodes, which may be in different nets. Raises an error if the type is not an arc type in either node's NCM, or if the mapping (NCM +*map* field) is violated.

destroynode(*node*)
Destroys the node and (using **destroyarc**) all arcs to and from it.

destroyarc(*arc*)
Destroys the arc and (using **destroynode**) none, one or both of the nodes it connects, depending on the arc dependency (NCM +*dep* field).

destroyall(*net*)
Destroys all nodes and arcs in the given net.

selectnode(*node, key*)

selectarc(*node, key*)
Informs all current applications and tools to apply a function to the node or arc; *key* is a simple word or number, which is translated by each application or tool into a real function as appropriate.

savenet(*net, filename*)
Saves a description of the net and its contents to the given disk file.

loadnet(*filename*) —> *net*
Creates and returns a net, with nodes and arcs, from the description in the given disk file.

changetype(*net, type*)
Changes the type of a net, by changing its NCM. Raises an error if the existing pattern of nodes and arcs is not compatible with the new NCM.

getncmfield(*net, type, field*)
Returns the value of an NCM field for the given node or arc type in the NCM for the given net.

slotvalue(name, item) —> value

value—> **slotvalue**(name, item)
Returns/updates the value of the named slot (for example the *header*) of the node or arc *item.*

Appendix 2: Expertext Shell procedures

ETSImportDocument(*file, type, title*) —> *doc*
Creates a new CONNEKT net with the given type and title, using **createnet.**
Creates a document structure for the net and loads text from *file* into the buffer.
Creates a *document-fragment* using **ETSCreateFragment**

ETSSaveDocument(*doc*)
Writes the document text buffer to the *file* recorded in the document structure (see

Documents above). Writes the style structure to *file*.sty. Calls CONNEKT **savenet** to record the document net in *file*.mkp.

ETSLoadDocument(*file*) —> *doc*
Creates a document structure. Loads text from *file* into text buffer. Creates style structure from *file*.sty. Creates document net from description in *file*.mkp using CONNEKT **loadnet**

ETSCreateFragment(*doc, span, content-type, display-type*) —> *fg*
Creates a fragment with the given attributes, by calling **createnode** to make a fragment node of the given *content-type* in the net of the specified document, and adding the appropriate structures to *header* and *record* slots. Fills in *creator* and *date* fields in the header.

ETSCreateLink(*from-fg, to-fg, content-type*) —> *link*
Creates a link of the given type between two fragments, using **createarc** and adding a *header* slot to the arc.

ETSDestroyFragment(*fg*)
Deletes the fragment from its document net, using **destroynode**

ETSDestroyLink(*link*)
Deletes the link from its document net, using **destroyarc**

ETSHeaderField(*name, item*) —> *value*

value —>**ETSHeaderField**(*name, item*)
Returns/updates the value of the named header field for the fragment or link *item.*

ETSParseQuery(*string*)
Converts a query string into ETS working memory patterns and asserts them (see the query translator above).

ETSStartRules()
Starts the rule interpreter with global rules and rules local to the currently selected fragment.

ETSSelectFragment(*fg*)
Calls **selectnode**(*fg*, "open") and adds the local rules for the fragment to the active rule list.

ETSDeselectFragment(*fg*)
Calls **selectnode**(*fg*, "close") and removes the local rules for the fragment from the active rule list.

select *fragment-list*

deselect *fragment-list*
Powerful syntactic interfaces to **ETSSelectFragment** and **ETSDeselectFragment**. See descriptions in ETS procedures: select & deselect.

wmadd *pattern*
Adds a pattern to the ETS working memory.

wmdelete *pattern*
Deletes a pattern from the ETS working memory.

wmmatches *pattern —> boolean*
Returns *true* if the pattern is in the current WM, *false* if it is not.

Appendix 3: Part N

PART N

ELECTRICAL INSTALLATIONS

Contents

ASTERISKS

Throughout the Technical Standards an asterisk against a standard denotes that a provision deemed to satisfy the standard or some aspect of the standard is specified at the end of the relevant Part.

ITALICS

Throughout the Technical Standards a term in italics is a defined term. The definition is listed in Part A, General.

Part N
Introduction

---

1. The intention of this Part is to ensure that electrical installations are safe in terms of the hazards likely to arise from defective installations, namely fire, electric shock and burns or other personal injury.

2. The regulation applies to installations in, or serving, buildings. An installation consists of the electircal wiring and associated components and fittings, including permanently secured and large stationary equipment, but excluding portable equipment and appliances. Exceptions are made for mine and quarry buildings and factories (as they have particular hazards and are subject to other legislation), statutory supply undertakers' works and extra-low voltage installations which are not supplied from a higher voltage (eg 240V) circuit.

3. The requirements of the regulation are deemed to be satisfied by complying with the 'Regulations for Electrical Installations', Fifteenth Edition, 1981, published by the Institution of Electrical Engineers (IEE). It should be noted that some of the provisions of the IEE regulations, such as the safeguards for livestock, the requirements for caravans, the provision of IEE completion and inspection certificates and the recommendations for periodic inspection and testing, are outwith the scope of the Building Standards Regulations.

Part N
Regulation 26

Electrical Installations

26 (1) Every electrical installation to which this regulation applies and every item of stationary electrical equipment connected to such an installation shall provide adequate protection against its being a source of fire or a cause of personal injury.

(2) This regulation shall not apply to an installation -

(a) serving a building or any part of a building to which the Mines and Quarries Act 1954(a) or the Factories Act 1961 applies;

(b) forming part of the works of an undertaker to which regulations for the supply and distribution of electricity made under the Electricity (Supply) Acts 1882 to 1936 or section 16 of the Energy Act 1983(b) apply; or

(c) consisting of a circuit (including a circuit for telecommunication or for transmission of sound, vision or data, or for alarm purposes) which operates at a voltage not normally exceeding 50 volts alternating current or 120 volts direct current, measured between any two conductors or between any conductor and earth, and which is not connected directly or indirectly to an electricity supply which operates at a voltage higher than those mentioned in this sub-paragraph.

(3) In paragraph (1) 'stationary electrical equipment' means electrical equipment which is fixed, or which has a mass exceeding 18kg and is not provided with a carrying handle.

(a) 1954 c.70
(b) 1983 c.25

Part N
The standards

N1 Application of Part N

N1.1 This part sets out the required standards for regulation 26

N1.2 The standards apply to electrical installations in, or serving, all buildings, except -

installations specified in regulation 26(2).

N2 Electrical installations

N2.1* An electrical installation must be constructed, installed and protected to minimise the risk of fire in the building or elsewhere.

In normal operation, taking into account the surroundings, it must not create the risk of burns, shock or other injury to people.

It must -

a. safely accommodate any likely maximum demand

b. incorporate suitable automatic devices for protection against overcurrent or leakage; and

c. have switches, or other means of isolating parts of the installation or equipment connected to it, as are necessary for safe working and maintenance.

Part N
Provisions deemed to satisfy the standards

ELECTRICAL INSTALLATIONS

(N2.1) The requirements of N2.1 will be met where an installation complies with the relevant requirements of the 'Regulations for Electrical Installations', Fifteenth Edition, 1981, published by the Institution of Electrical Engineers.