DIVISION OF COMPUTER SCIENCE

An Integrated Approach to the Development of Advice Systems to Support Learning and Domain Based Information Retrieval

Mike Bearne
Jill Hewitt
Sara Jones
John Sapsford-Francis

Technical Report No.180

February 1994

Technical Report No. 180 February 1994

An Integrated Approach to the Development of Advice Systems to Support Learning and Domain Based Information Retrieval

Mike Bearne, Jill Hewitt, Sara Jones and John Sapsford-Francis
comrmb, comqjah, comrsj, comqjs @herts.ac.uk

School of Information Sciences, University of Hertfordshire,
College Lane, Hatfield, Herts, AL10 9AB, UK

Tel: (0707) 284766, 284327, 284370, 284354
Fax: (0707) 284303

Abstract

We present a method for developing advice systems. The focus is on allowing users to access information about large, ill-defined domains using their existing knowledge and understanding of those domains. The method relies on iterative refinement and integration of models of the domain, the requirements and the interface. We propose the use of a Generic Advice Systems Architecture which provides the high-level structure for interfaces to such systems, without premature commitment to a particular 'look and feel'. The method is illustrated with material from an ongoing project aimed at developing an advice system to support the integration into Higher Education of students with disabilities .

Submitted for presentation at HCI '94, the annual conference of the BCS HCI Special Interest Group, to be held at the University of Glasgow, 23 - 26 August 1994.

An Integrated Approach to the Development of Advice Systems to Support Learning and Domain Based Information Retrieval

Mike Bearne, Jill Hewitt, Sara Jones and John Sapsford-Francis
comrmb, comqjah, comrsj, comqjs @herts.ac.uk

School of Information Sciences, University of Hertfordshire,
College Lane, Hatfield, Herts, AL10 9AB, UK

Tel: (0707) 284766, 284327, 284370, 284354
Fax: (0707) 284303

Abstract
We present a method for developing advice systems. The focus is on allowing users to access information about large, ill-defined domains using their existing knowledge and understanding of those domains. The method relies on iterative refinement and integration of models of the domain, the requirements and the interface. We propose the use of a Generic Advice Systems Architecture which provides the high-level structure for interfaces to such systems, without premature commitment to a particular 'look and feel'. The method is illustrated with material from an ongoing project aimed at developing an advice system to support the integration into Higher Education of students with disabilities .

Keywords: design method, domain modelling, task analysis, knowledge elicitation, advice systems, iterative prototyping

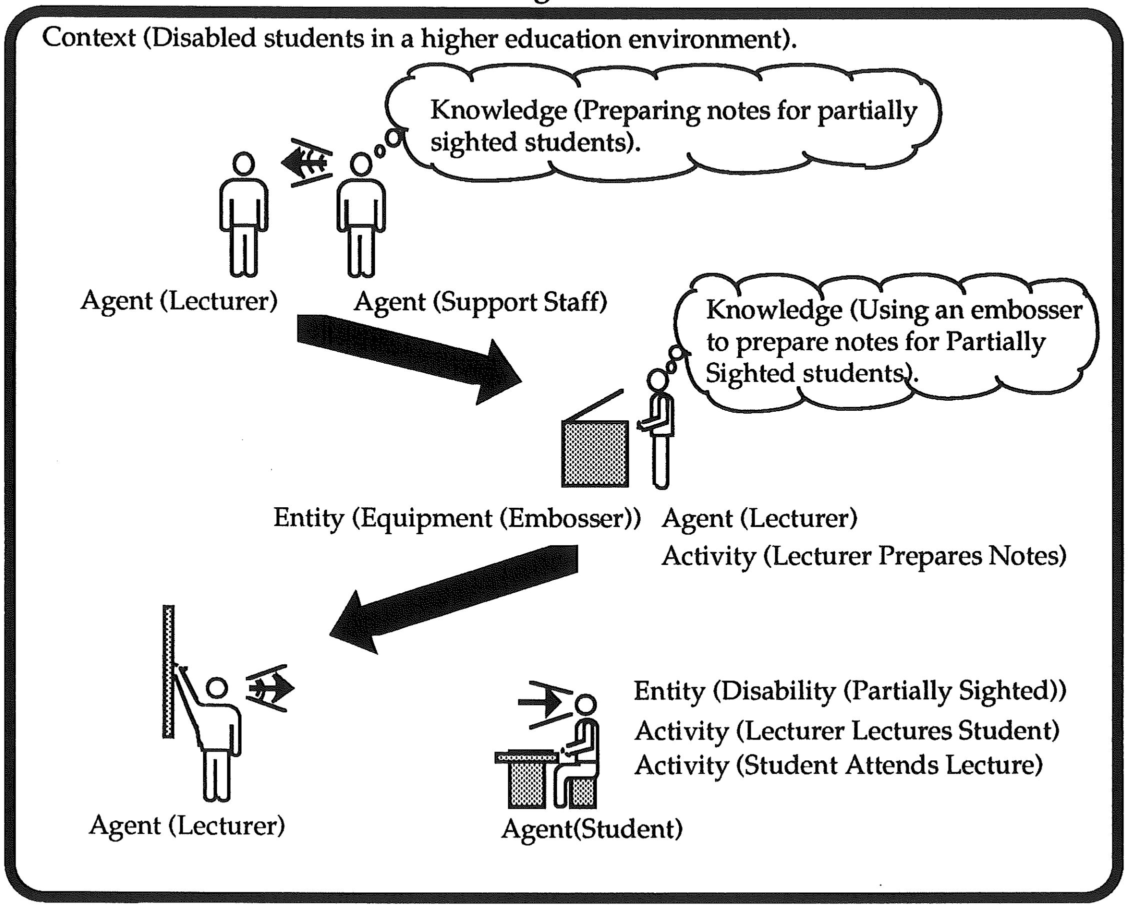
1. 0 Introduction

This paper describes and illustrates a method for developing advisory systems. One of the key issues, that the method addresses, is the use of domain modelling to provide a foundation structure for the users' interaction with the system.

Advisory systems are typified by : a large group of infrequent users, diverse sources of knowledge, lack of definitive solutions to user problems, the need to assist naive users with query structuring and the need to persuade users of the advantages of using the system in preference to existing methods.

Illustrative material is drawn from the SPIRE project. This project (funded under the Teaching and Learning Technology Programme 2) concerns the development of a multimedia advisory system that will assist with the integration of students with disabilities into Higher Education. The current practice in providing support can be described as "ask someone who knows or who has done it before". There is a very informal network of lecturers and support staff who have built up individual bodies of knowledge relating to particular aspects of the domain. Any of these people may be approached by someone with a problem, and, if they do not know the answer, they may refer the inquirer to other internal contacts, to a knowledge base of pamphlets or manufacturer's brochures or to outside organisations. The system is unstructured, and, for the individual inquirer, frustrating and serendipitous, since they often have no clear idea of who to ask or where to go for information. The domain and some of the emergent substructures within it are illustrated in Figure 1.

Figure 1


Context (Disabled students in a higher education environment).
Knowledge (Preparing notes for partially sighted students).
Agent (Lecturer)
Agent (Support Staff)
Knowledge (Using an embosser to prepare notes for Partially Sighted students).
Entity (Equipment (Embosser))
Agent (Lecturer)
Activity (Lecturer Prepares Notes)
Entity (Disability (Partially Sighted))
Activity (Lecturer Lectures Student)
Activity (Student Attends Lecture)
Agent (Lecturer)
Agent(Student)

The provision of an automated advice system would appear to provide an attractive alternative to the current situation. A system which contained a permanent, easily maintainable data store, and which supported queries from a wide range of non-expert users could considerably ease the load on the current staff 'experts' whilst increasing users' level of satisfaction with response to their queries. In addition such a system could give a wider access to information, particularly by supporting queries from potential students and members of staff with a peripheral interest in the field.

The authors' work on the MODEMA project (Hewitt et al. 1994) also addressed many of these issues. MODEMA was an 18 month pan-European project funded under the EC TIDE initiative. The project objective was to build a prototype knowledge based browsing system that provided advice on the integration of people with disabilities into the working environment. The project was successfully completed in June 1993.

This type of project presents a number of important issues:

- Usability of the system: the system will be highly interactive, usage of the system for individual users will be infrequent so learnability and accessibility will be important. The main high level task, that of supporting the integration of students with disability into higher education, is a difficult one and one that is often done badly. It is vital that any computer system introduced into this problem domain does not provide a further barrier to good practice.

- typical users of the system will be naive about the content of the domain of knowledge, they will also be naive about the best way to access this knowledge, for example they may say that they do not know what questions to ask. The system must therefore be sufficiently flexible to support exploration learning of the domain as well as the provision of useful advice on a wide variety of specific queries.

- the expertise that must be captured and represented within the system is held by a large number of "narrow domain" experts, for example experts in rehabilitation technology, experts in particular disabilities, social services advisors. It is very rare that any one individual expert possess substantial general expertise. Currently there are no systems that meet the needs that we address here.

- issues concerning the integration into Higher Education of students with disability are extremely complex, and cannot be satisfactorily dealt with by an expert system in that frequently no single solution can be offered, However it is possible to present a number of useful approaches or advice on procedures that have worked well in the past.

- there is no formal customer authority. There is no formal requirements specification. Under these circumstances the developers of the system must decide for themselves what needs they are trying to meet and how they will meet them.

This problem is not an entirely new one. All design problems are more or less ill-structured - Guindon (1990) quotes Simon's characterisation of ill-structured problems as : incomplete and ambiguous specification of goals, no predetermined solution path and the need for integration of multiple knowledge domains. However, the degree of uncertainty and the fact that not only is there no pre-determined solution path from the requirements to the finished artefact, but not even any clear idea of requirements, make the class of problem we are addressing particularly taxing.

The development approach for this class of problem requires a combination of knowledge elicitation, task modelling of user and domain tasks, and iterative design and evaluation with potential user groups to allow refinement of domain and requirements models in parallel with interface development. The target system does not exhibit the characteristics of an expert system (see for example, (Amble, 1987)), in that it does not offer a single solution to a user query, only information that may help the user to find a solution. It is closer in concept to the idea of an intelligent information retrieval system as described in (Brazier & Ruttkay, 1993), but does not reason on the behalf of a client, rather providing the client with the information needed to make informed choices.

Previous experience and current problems have lead us to develop a method that enables the development of systems such as this. This method specifically supports:

- the elicitation of expertise from many different sources
- the integration of this expertise into a coherent body that is represented in the system
- the use of early prototype systems in the validation of requirements and knowledge representation
- parallel development of requirements and domain models
- a task analysis of the current domain
- a projected task analysis of how users of this system will usefully access the knowledge
- the provision of an interface structure underpinned by this task analysis that supports naive users and helps them decide what advice they require.

In the following sections we describe this method.

2.0 Developing a System

A major goal in developing an advice system must be that people will choose to use it in preference to any existing system. A more ambitious goal is that people who do not choose to use the existing system will become users of the new system and will thus be 'drawn into' the domain. More ambitious still is the goal that the system will help to perpetuate domain experts, by providing support for users to learn about the environment.

For the SPIRE project it is envisaged that the ideal system would assist the users in building up their general knowledge of the domain as well as answer specific queries. This is particularly important in view of the fact that non-experts often start by asking the wrong questions. An example of this could be "Which rooms are accessible by blind students?", the answer to which might be "All of them", whereas the questions "Where can I get a Braille map of the University?" or "How do I present diagrams to a blind person?" would elicit more pertinent responses.

It is the aim of the project team to provide a system for each university campus by the end of the project. These will contain general information, information relevant specifically to the university and, where appropriate, site specific information (e.g. local contacts, locally available equipment). A longer term aim is to produce a generic system shell into which other universities can integrate their own specific information.

Key aspects of the development of this type of system are:

- it utilises both a requirements and a domain model. This facilitates the structuring of the advice about aspects of the domain in a way which corresponds to the structure of the domain itself as seen by experienced users.
- an early throw-away prototype is used and evaluated with potential users to help consolidate the requirements and domain models and to provide a focus for further knowledge elicitation.
- an iterative prototyping approach is followed, with the aim of maximising user satisfaction with the system
- an early emphasis is put on the development of the information management aspects of the system, thus enabling easy adaptations to the model as they are discovered through the iterative development cycle.

An overview of the proposed method is illustrated in Figure 2.

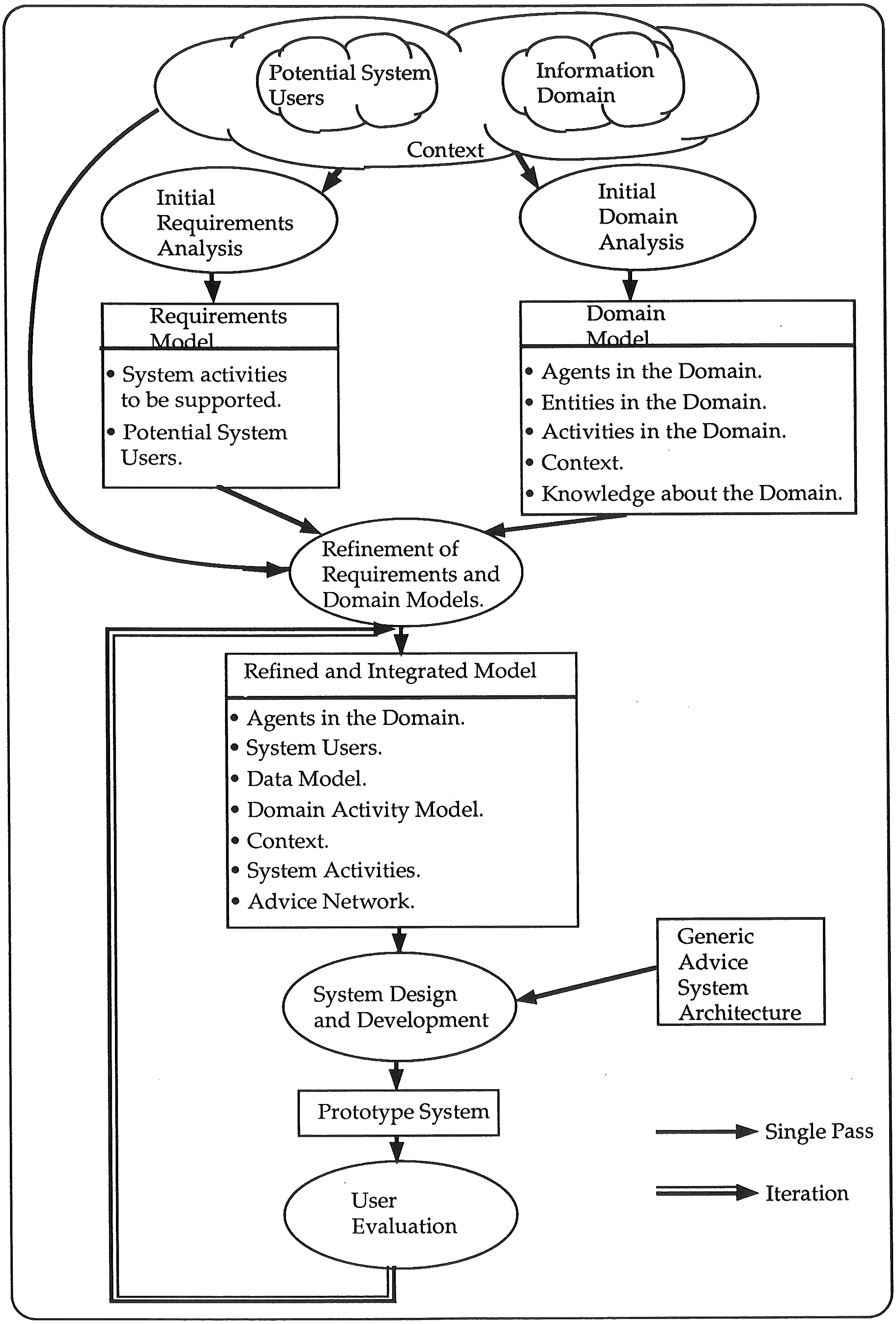
2.1 Initial Models of the Domain and System Requirements

As described above, these models are developed in parallel.

2.1.1 Initial Requirements Model
It is typical of this type of development that there is no client-tendered requirements specification, but there is likely to be some general consensus amongst potential users that such a system would help them to work more efficiently. The fact that there is no existing automated system, nor even a well-structured manual one, means that users have really very little idea of what a system might do for them, beyond one or two specific query scenarios which they may have formulated in the past.

These things make the requirements specification hard. More effort has to be put into explaining the possibilities to the users, and negotiating with them as to what can reasonably be provided with the time and money available. In addition there are political issues, for example the opinion of the existing experts on the desirability of such a system may be divided. In the development of the MODEMA system some resistance was encountered in the form of experts who did not want users to be better informed as they "could misinterpret information" (MODEMA deliverable 8,1993)

Figure 2: Advice system development method


Potential System Users
Information Domain
Context
Initial Requirements Analysis
Initial Domain Analysis
Requirements Model
• System activities to be supported.
• Potential System Users.
Domain Model
• Agents in the Domain.
• Entities in the Domain.
• Activities in the Domain.
• Context.
• Knowledge about the Domain.
Refinement of Requirements and Domain Models.
Refined and Integrated Model
• Agents in the Domain.
• System Users.
• Data Model.
• Domain Activity Model.
• Context.
• System Activities.
• Advice Network.
Generic Advice System Architecture
System Design and Development
Prototype System
User Evaluation
Single Pass
Iteration

The first attempt at requirements specification must be based on informed guess work by the developers, using their domain knowledge.

For SPIRE, our first guess at high-level system requirements were:

System users
These would be lecturers, current and potential disabled students and support staff (e.g. exams officers, accommodation officers etc.)

System activities
1. Allow users to obtain advice about how to support disabled students specific to the user's role (lecturer, students etc), and the disability of the student. Expert users might want specific information, novices might want to be guided through the system more.

2. Allow users to browse around learning about the domain.

2.1.2 Initial Domain Model
This must be derived from studying documents, talking to experts in the domain, and any users they support, attending meetings and generally observing any existing protocols. At this stage the description is in terms of natural language lists of items, there is no point in using any formalism so early in the modelling process.

For SPIRE we had access to existing support staff and to lecturers who had dealt with disabled students. We also attended meetings of the designated "Disability Network" - a group of staff with varying responsibilities for looking after disabled students. The initial model is described below.

Context
Disabled students in higher education

Agents in the domain
These are people who carry out tasks which are relevant in the context of the system

Example: Lecturers, Students, Current Disabled Students, Potential Disabled Students, Support Staff

Entities in the domain
These are artefacts in the domain about which we wish to record information

Example: Braille embosser, RNIB Braille Service, Dyslexia, Partial-Sightedness, Disabled Students Allowance, ScreenEnlarging Software, Site

Activities in the domain
These are the typical activities which are carried out in the domain, which together make up a task model of the domain

Example: students attend lectures
lecturers teach lectures
lecturers prepare handouts
partially sighted student uses screen enlarging software
lecturers prepare exam papers

2.2 Requirements and Domain Knowledge Elicitation

In addition to individual interviews, it was found that seminar activities provide a good forum for the elicitation of user requirements and domain knowledge. Two techniques which were used are described below:

2.2.1 Card sort
A card sorting exercise (Cordingley, E.S.B. 1989) was carried out to help with elicitation of the static knowledge structures of people working in the domain

List of entities were taken from initial domain model, these were written on cards, one per card, and participants were asked to group them in any way which seemed natural to them. Since people have different knowledge structures, it is normal to envisage that not all people will group entities in the same way, but the areas where there is a clear consensus are prime candidates for entity classifications in the system.

The results of this exercise will have more validity than an entity class system imposed by the designers since they reflect the opinions of the users and enable the users to view the system in their own terms.

Example: Sample categorisations of some key objects:

Unanimous categorisation:	Equipment, Disability, Money
Majority categorisation:	Policy
Minority categorisation:	Accommodation

2.2.2 Scenario-based requirements and knowledge elicitation
Scenarios were generated from the initial domain model and used in an exercise to elicit further user knowledge of the domain and further details for the user requirements specification. The use of scenarios in the pre-prototype stage of development help the designers to build more realistic first prototypes (Young & Barnard, 1987)

> **Example:** "You are a lecturer who has been told that one of the students in a forthcoming class that you are teaching is partially sighted. What information and advice might be helpful to you and in what context?"

Requirements elicitation
The question from the scenario asking the user to identify helpful information and advice helps to identify the user expectations of any system. The responses helped to identify requirements for system activities which should be added to the model at this stage:

> **Example:**
> lecturer consults university policy on dyslexia
> lecturer asks student if they are registered as dyslexic

Domain knowledge elicitation
The responses to the scenarios contained implicit information which helped to identify further knowledge of what activities are carried out in the domain.

> **Example:** Some activities which were added to the initial domain model after questioning potential users :
>
> students taking notes
> students taping lectures
> students doing group work
> lecturer setting exams

2.2.3 The Throw-Away Prototype

The construction and evaluation of an early "throw away" prototype system, that embodies our initial understanding of user requirements and user domain knowledge provides an important validation method. Presenting such prototype systems to users and to the various "narrow domain" experts in the context of particular problems also enables us to elicit further user requirements and expert knowledge. As we have stated earlier a typifying feature of advisory systems is that they concern diverse sources of knowledge where there is a lack of definitive solutions to user problems. Under these circumstances it is hard for users and for "narrow domain" experts to provide complete (or indeed adequate) views of user requirements and of relevant domain knowledge. The presentation of a prototype system:

- forms a basis for dialogue between the potential users of the system and the designers
- gives potential users a sense of the kind of thing that might be possible
- gives the users confidence that solutions to particular problems might be found.

This method of validation and further requirements analysis and knowledge elicitation is carried out in parallel with other design and development activities (eg requirements and domain knowledge elicitation) in which the designers develop their understanding of the domain and tasks which need to be supported.

It is usually desirable to give maximum access to the prototype, including as wide range of users as possible, and a "hallway and storefront" type of evaluation is extremely useful in getting users immediate reactions. There is a slight reservation, in that as it is difficult and impractical to include a lot of structured knowledge in an early prototype, some users may judge the system on its paucity of content. We have found that careful explanations and the provision of some set routes through the system helps to alleviate this problem.

2.4 The Refined Integrated Model

Following the evaluations of the initial prototype, and the elicitation exercises, it is possible to integrate findings about the nature of the domain and system requirements by amalgamating the responses to the various knowledge elicitation and requirements capture exercises (above). In this process the priorities are to emphasise generic activities and categorisations while keeping them sufficiently concrete and complete to be useful.

It is written a little more formally because we need to be precise about the way in which we are rationalising and compromising between views of the domain elicited from different sources. We are also getting closer to the implementation and we need to make it easy to make changes during iterative development. However, too much formality is undesirable because the model needs to be understood and agreed by all members of the design team and there is no need for formal proofs at this stage.

This process has some parallels with TAKD in the building of lists of generic objects and actions.

2.4.1 Agents in the domain

```
Agent ::= Lecturer|Student|Potential Disabled Student
          |Current Disabled Student| Support Personnel
```

2.4.2 System users

This is derived from list of agents in the domain:

```
User ::= Lecturer|Potential Disabled Student
          |Current Disabled Student| Support Personnel
```

The user type will be implemented as filter on information presented, allowing different users to follow slightly different dialogues and see slightly different screens - see below.

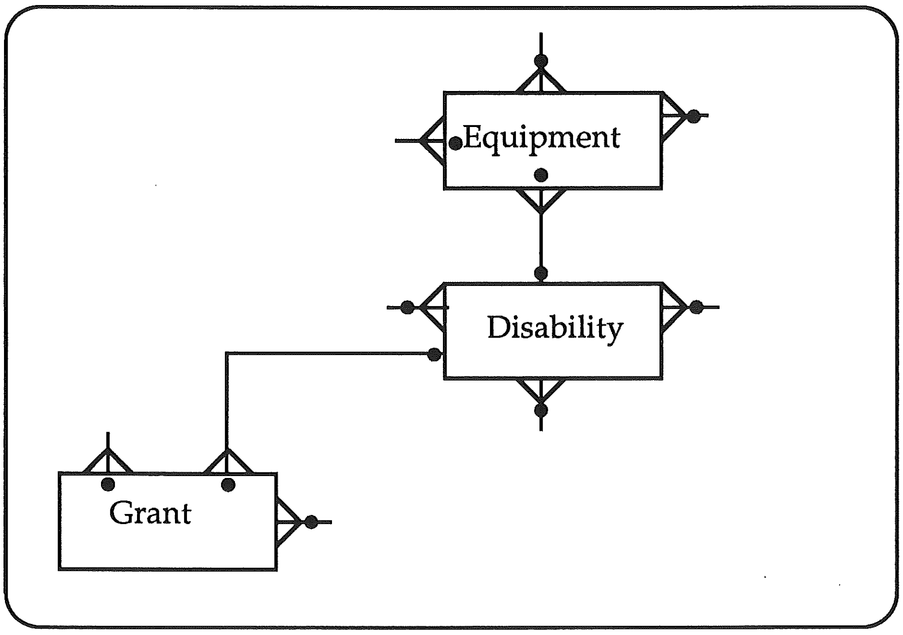
2.4.3 Data model

Derived from sets of entities in the domain as elicited in card sort, for example:

```
Equipment = {Enlarger, Braille Embosser, Robot Arm .....}

Grants = {Disabled Students Allowance, Discretionary Grant,
             European Grant,...}
```

The entity model will be normalised as it will be implemented using a relational database package for reasons of efficiency, giving fast information retrieval and a flexible structure for update and amendment. It can be expressed as an Entity-Relationship model.

Figure 3: ER data model.



2.4.4 Domain activity model

This presents hierarchies of activities as elicited in the scenario-based knowledge extraction. Although we are interested in the hierarchical relationship between activities, the recording of further information relating to plans is not necessary, since they are not the target of a user query, merely a structuring tool.

Example:

```
Lecturer teach lecture
     Lecturer prepare handouts
     Lecturer prepare visual material
     Lecturer write-on board
     Lecturer present video
```

```
Student do coursework
     Student use library
          Student use on-line search facility
          Student locate book
     Student take notes
          Student prepare essay
          Student write text
          Student draw diagrams
```

This task hierarchy is embodied as part of the dialogue stricture, as described in more detail in section 3.1

2.4.5 Context
This can be expressed as:

```
Disability X Users X Site
```

Instances of the context can be set up by the users to allow the system to refine queries to reflect only the current context.

2.4.6 System activities
The structured queries are presented as more detailed task hierarchies eg for entering information about user role, student disability; for getting advice about particular situations etc.

Example:

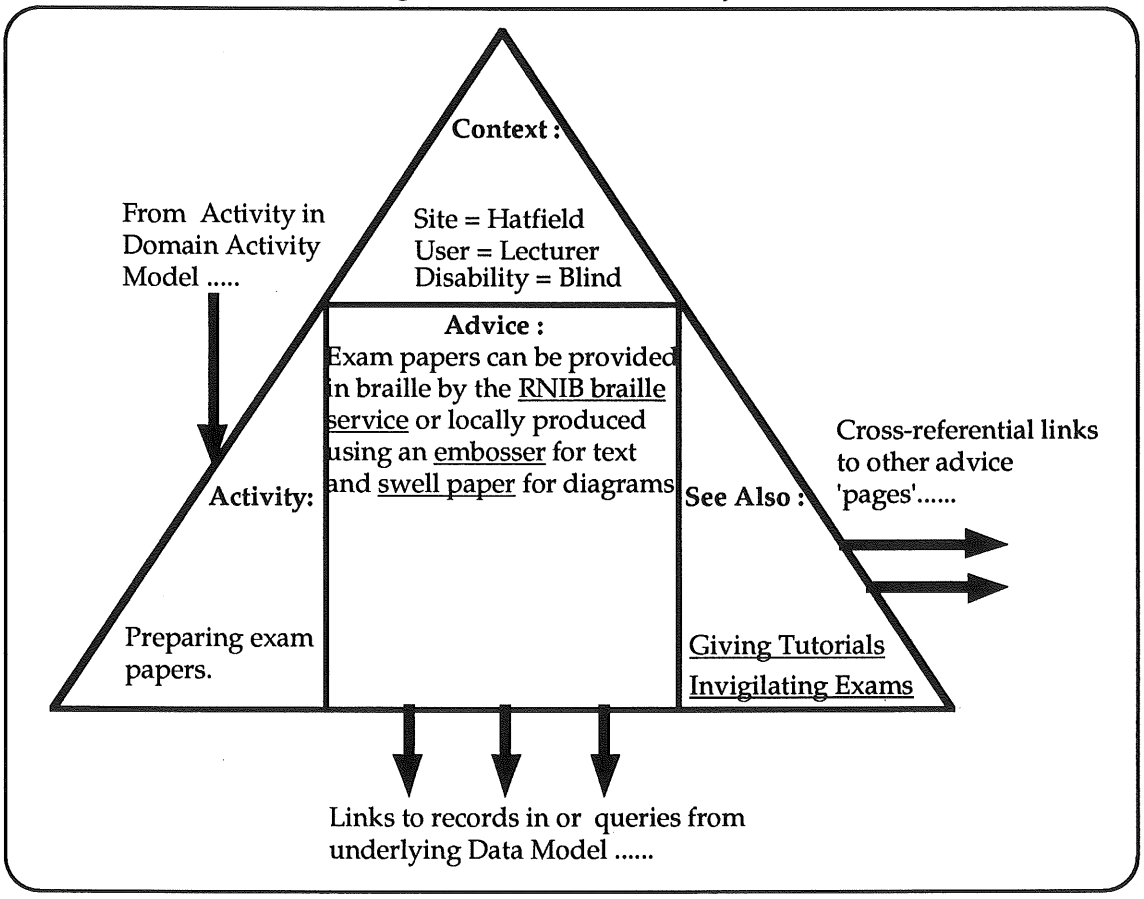
```
user enters context for a query
     user enters disability
     user enters user role
     user enters site

user finds out how to prepare an exam for a blind/partially
 sighted student
     user consults University policy on exams for partially-
      sighted students
     user finds out how to prepare exam papers for partially-
      sighted students
          user finds out how to prepare textual documents
          user finds out how to prepare graphics documents
```

The database querying level is not represented here as this will be modelled as direct queries on the relational data model.

3.4.7 Advice Network
The links between structured queries and the data model are provided pieces of advice which present an 'expert's response' to the users query which embodies text and hypertext links to other pieces of advice and to the data model. This layer is necessary to avoid the need to link everything to everything else, and it imposes a bit more apparent structure into the interface to avoid the 'lost in hyperspace'(Meister,89) phenomenon.

The pieces of advice are structured objects, as represented in figure 4. At the centre is the advice text, with hypertext links underlined. These provide direct access to the database objects. A 'see also' section highlights links to other pieces of advice which may be relevant to the user's query. Each piece of advice is relevant only in a particular context and is related to an activity in the Domain Activity Model. The separation of the database links from the links to other advice entities is another strategy to help the user avoid getting lost in the system.

Figure 4: An Advice Object


Context :
Site = Hatfield
User = Lecturer
Disability = Blind
From Activity in Domain Activity Model .....
Advice :
Exam papers can be provided in braille by the RNIB braille service or locally produced using an embosser for text and swell paper for diagrams
Activity:
Preparing exam papers.
See Also :
Giving Tutorials
Invigilating Exams
Cross-referential links to other advice 'pages'......
Links to records in or queries from underlying Data Model ......

2.5 Generic Advice System Architecture

An integrated view of the system architecture is shown in Figure 5. The system model can be viewed as three inter-relating planes. At the top is the Domain Activity Model which helps to provide a focus for the user to structure his/her queries. Beneath this is the Advice Network, the structure of this is determined by the context entered by the user, since advice is context specific. At the bottom is the relational data model.

The diagram shows system users as a subset of the agents in the domain, and traces a query through the planes of the model to reach the target data, we must assume that the context has been set. At the top level, the user is navigating a hierarchy of tasks to find the sub-task in which he/she is interested. This is then linked to a piece of advice

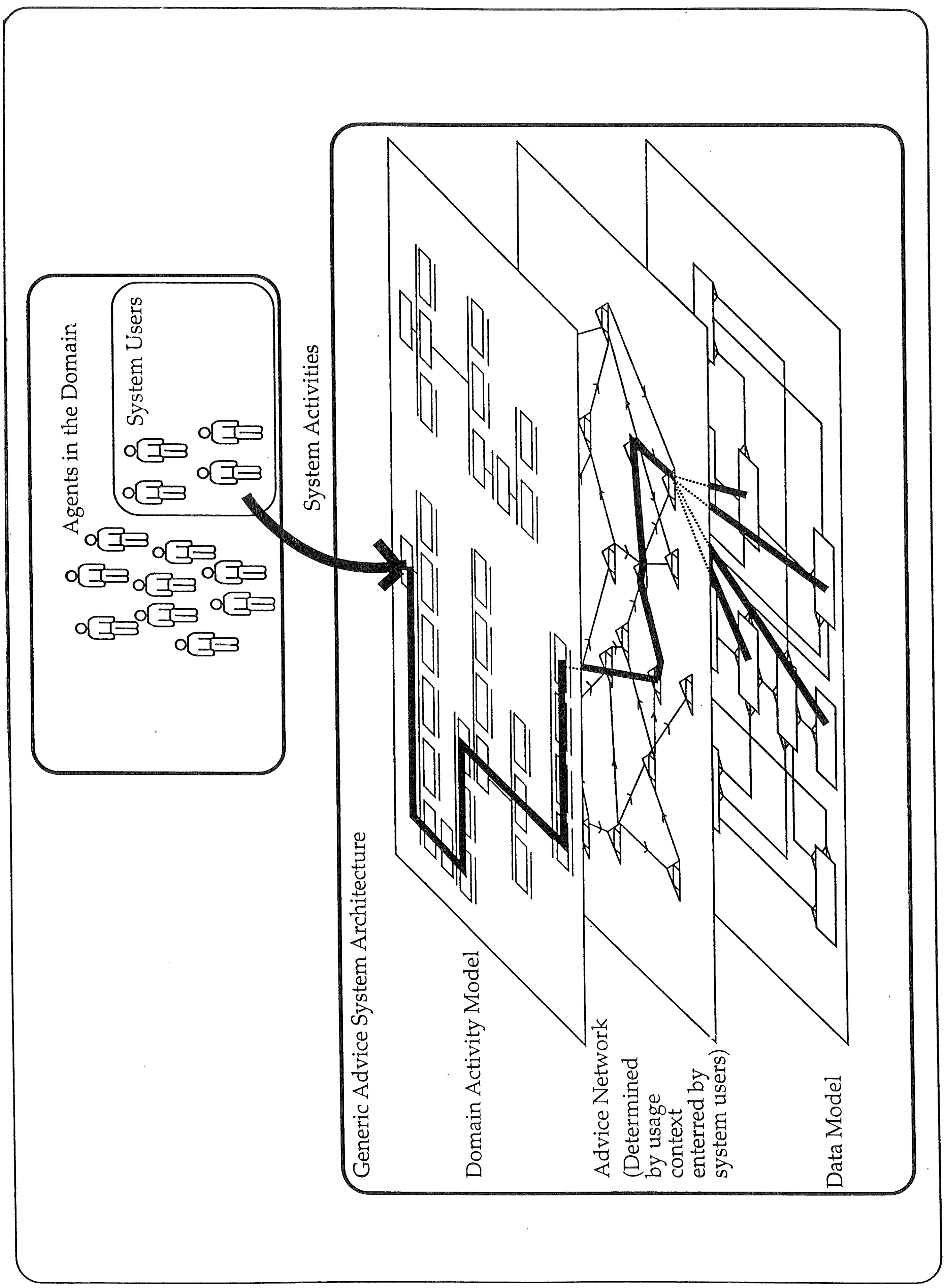

Agents in the Domain
System Users
System Activities
Generic Advice System Architecture
Domain Activity Model
Advice Network (Determined by usage context entered by system users)
Data Model

Fig. 5. Generic Advice System Architecture

relevant for the context, and the user looks through three more pieces of advice before selecting four links to the underlying database.

For direct database access, the query would be shown only on the bottom plane.

2.6 System Design and Implementation

Although the first throw-away prototype can be implemented with almost any prototyping tool, it is important that certain factors are taken into account when choosing the platform for subsequent incremental prototypes and the finished system.

- all the data in the system will need to be regularly updated whilst the system is being developed, since the elicitation and refinement process is being carried out in parallel with interface development

- an interface which allows domain experts to update and maintain the system will need to be developed in parallel with the other system developments, and ideally will be in place early in the project cycle to make it easier to enter data whilst the development is in progress.

- if multimedia artefacts (video, sound, images) are to be used, due thought should be given to storage aspects and to the presentation platform which will be required for the finished system. In some cases there may be a requirement to develop a 'cut down' version which will run on older, lower specification machines without the multimedia element.

For the SPIRE system, the initial prototype was built using Hypercard, with video clips implemented using Quicktime tools, however subsequent versions are being based on a cross-platform relational database package which supports a graphical user interface. All three layers of the system model can be implemented as sets of database tables, making data entry and subsequent modifications a relatively simple task.

2.7 Refinement through Iterative Incremental Prototyping

Results of user evaluations will feed back into enhancements to the system model and the interface model (to be described below). Since they are formally specified, it should be quite easy to update them.

3.0 A Model of Human-Computer Dialogue

Once the three layers of the Generic Advice System Architecture have been filled out with material from the refined models of requirements and domain knowledge, the high-level structure of the interface has been defined.

It is intended that users should begin a session with the system by entering information defining a context of interest, for example, that of a lecturer teaching a blind student on the Hatfield campus. On the basis of the information entered, particular layers of domain activities and advice are selected. These then form the basis for the user's interactions with the system by determining the paths which the user may follow through the system. The user in our example would first be presented with hierarchies of activities carried out by lecturers and asked to select an activity of particular interest. He or she would then be presented with the appropriate piece of advice from the advice layer. From that advice, it would be possible either to access information from the data layer or further pieces of advice on related topics.

The task of interface design is thus confined mainly to defining a look and feel to the system. It would be possible to implement the interface using a range of technologies. Advice systems of the kind envisaged could in theory be implemented using text-based command line interfaces. However, both SPIRE and MODEMA have employed multimedia interface technologies for the presentation of information from the data layer, as it has been felt that it is more effective to present such information using a combination of text, graphics and video. It is intended that sound might also be added in future prototypes of the SPIRE system.

We are currently considering the way in which the complex task of designing multimedia interfaces can be supported by modelling dialogue in those interfaces using the CSP notation [Hoare 85]. CSP can model all systems or dialogues which can be represented using either state transition diagrams or Petri nets. It is quite readable for non-mathematicians, even in its textual form, and tools such as Alexander's SPI [Alexander 87] have been developed to support graphical animation of specifications written in a subset of CSP. Previous work [Jacob and Jones 94] has suggested the way in which a small part of the CSP notation can be used to model multimedia interfaces. The rest of this section describes the way in which a fragment of the dialogue between the lecturer of a blind student and the data layer of our advice system may be specified.

Symbols used include: $\hat{=}$ which means 'is defined as'; -> which can be read as 'then'; $\sqcap$ and $\square$ which denote choice between two alternative paths of behaviour, internally and externally to the process in which the choice is made; $||$ which means, roughly 'during the same time period as'; and `SKIP` which means that a particular part of the dialogue has finished.

We assume that information about the context of interest has already been entered, and consider a user who has already navigated to a particular point in the activities layer and selected the activity 'preparing exams' as being of particular interest. For the sake of illustration, we assume that the user has three interests in this respect: she wants to find out what the policy of the University of Hertfordshire is in this case; she wants to find out how the exam papers might be produced; and she is also concerned about the way in which the exam will be marked. We model this simplified view of the user as follows:

```
Lecturer_of_Blind_Students_at_Hatfield ≙

      Lecturer_of_Blind_Students_at_Hatfield_Consulting_UH_Policy ||

      Lecturer_of_Blind_Students_at_Hatfield_Preparing_Exam_Papers ||

      Lecturer_of_Blind_Students_at_Hatfield_Marking_Exams
```

The user begins the dialogue by asking for advice about the activity selected:

```
Lecturer_of_Blind_Students_at_Hatfield_Preparing_Exam_Papers ≙

      open_advice_preparing_exam_papers_for_blind_students_at_hatfield ->

      Seeking_Advice_Preparing_Documents_for_Blind_Students_at_Hatfield
```

She is then presented with a piece of text presenting the advice shown in figure 4. At this point, she is able to access information from the data layer relating to various components of the advice (including the RNIB Braille service, and embossers and swellpaper machines held at Hatfield), or to request advice on another interest of hers (that of marking exams completed by blind students at Hatfield) not specifically covered:

```
Seeking_Advice_Preparing_Documents_for_Blind_Students_at_Hatfield ≙

      (open_info_RNIB_braille_service ->

        Seeking_Advice_Preparing_Documents_for_Blind_Students_at_Hatfield

      ⊓

       open_info_embosser_at_hatfield ->

       (begin_embosser_video_play ->

        Seeking_Advice_Preparing_Documents_for_Blind_Students_at_Hatfield

        ⊓

        Seeking_Advice_Preparing_Documents_for_Blind_Students_at_Hatfield)

      ⊓

       open_info_swellpaper_at_hatfield ->

        Seeking_Advice_Preparing_Documents_for_Blind_Students_at_Hatfield

      ⊓

       open_advice_marking_exams_for_blind_students_at_hatfield)
```

Information from the data layer is presented by a system module referred to below as the `Information_Displayer`. This module handles the presentation of information in a range of media: text, graphic stills and videos for the purposes of this example. The information displayer co-ordinates presentation of information in different media so that, for example, text and graphics relating to the same entity in the data layer are presented at the same time and the timing of a video presentation can be controlled. It can be modelled as three separate processes controlling the presentation of the three media of interest, which run in parallel with another process synchronising with the user's input:

```
Information_Displayer ≙ Info_Text_Displayer || Info_Graphics_Displayer ||

                                Info_Video_Displayer

                                ||

                                (open_info_RNIB_braille_service ->

                                 RNIB_Braille_Service_Info_Display

                                □

                                 open_info_embosser_at_hatfield ->

                                 Embosser_Info_Display

                                □

                                 open_info_swellpaper_at_hatfield ->

                                 Swellpaper_Info_Display

                                □

                                 open_info_braille_decoders_at_hatfield ->

                                 .......

                                 )
```

Where the relevant information is displayed in a purely textual form, the presentation is simple:

```
RNIB_Braille_Service_Info_Display ≙

      display_text_info_RNIB_braille_service -> SKIP
```

Where several components of information in different media are to be presented, the process is a little more complicated as presentation of the different media has to be synchronised. Here, for example, textual information about the embossing machine at Hatfield is presented at the same time as a graphic still showing what the machine looks like. There is also a process which supports the running of a video showing how the embosser will be used: this process is set in motion by the user's request to 'begin_embosser_video_play' which was modelled as an optional part of the dialogue under 'Seeking_Advice_Preparing_Documents_for_Blind _Students_at_Hatfield' above:

```
Embosser_Info_Display ≙ Embosser_Text_Display || Embosser_Graphic_Display ||

                        Embosser_Video_Display

Embosser_Text_Display ≙ begin_info_embosser ->

                        display_text_info_embosser -> SKIP

Embosser_Graphic_Display ≙ begin_info_embosser ->

                           display_graphic_info_embosser -> SKIP

Embosser_Video_Display ≙ begin_embosser_video_play ->

                         end_embosser_video_play -> SKIP
```

One of the main benefits of modelling dialogue with a multimedia system in this way is that it should allow us to determine in a precise way, using mathematical reasoning, whether the design proposed conforms to particular specifications, for example, those imposed by the hardware on which the system is to be implemented. For example, many machines are not powerful enough to run more than one video at the same time. If our dialogue design has been specified in CSP, we should be able to show that it does not permit situations in which this would be the case. Work on determining the extent to which complete dialogues of this kind can be specified in CSP, and the ease with which proofs of salient properties can be constructed is ongoing.

An appropriate representation for the relationship between the CSP specification and a model of screen states is currently being considered.

4.0 Conclusions

We have presented a method for developing advice systems which allows users to access information about a domain of interest using their existing knowledge and understanding. It supports exploratory learning which will increase the users' expertise in the domain.

This method has evolved out of our experience of developing advisory systems. A number of problematic issues, associated with advisory systems have been addressed.

The Refinement of Requirements and Domain Models uses a combined approach to build upon the initial requirements and domain models. This combined approach might include such activities as interviewing, card sorting exercises, scenario-based requirements and knowledge elicitation and the presentation of a "throw away" prototype. The resulting Refined and Integrated Model includes the information necessary for addressing usability issues. This information will have come from both the requirements analysis and the domain analysis. It enables us to design an interface structure that capitalises on users' domain knowledge.

The structured nature of the Refined and Integrated Model means that it is easy to incorporate enhancements suggested during the process of validation and verification by users of the system. Models of the Domain, the Interface and the Requirements for the system can all be evolved in parallel.

The Generic Interface Architecture provides a flexible structure within which a wide range of interactions can be pursued. It provides a high level structure for the interaction in terms of a Domain Activity Model, and an Advice Network. This supports the seeking of general advice, exploratory learning in the domain and obtaining specific information.

This architecture can be applied to the development of a range of systems of this kind, for example a similar structure was used in MODEMA. We have also considered applications such as careers advice, rehabilitation programmes, and a reference model for creating multimedia systems. The architecture leaves the interaction style, "look and feel" of the interface open, so that it can be implemented on a range of platforms providing access to users with special needs.

Work on the SPIRE project is continuing and we aim through this work to refine the method and provide more formal modelling techniques allowing for further integration between components of the proposed models.

5.0 References

Alexander, H. 1987, "Formally-Based Tools and Techniques for Human-Computer Interaction", Ellis Horwood
Amble, T. 1987, Logic Programming and Knowledge Engineering, Addison Wesley.
Brazier F.M.T. & Ruttkay, Zs. 1993, "Modelling Collective User Satisfaction", in Advances in Human Factors/Ergonomics, 19A, Human-Computer Interaction, Smith & Salvendy, eds. Elsevier.
Cordingley, E.S.B. 1989, Knowledge Elicitation Techniques for Knowledge Based Systems. in Knowledge Elicitation: Principles techniques and applications. Diaper, D. (Ed.) Ellis-Horwood 1989.
Guindon, Raymonde, 1990, "Designing the Design Process", Human Computer Interaction, V5 Nos. 2 & 3 1990

Hewitt, J. Sapsford-Francis, J. and Halford, P. 1994 "MODEMA - A Multimedia Public Information System" in Multimedia Technologies and Future Applications, Pentech Press Limited
Hoare C.A.R. 1985, "Communicating Sequential Processes", Prentice Hall, 1985.
Meister, D, 1989, "Lost in Computer Space", Int. J. of H.C.I. Vol 1 No. 2
MODEMA Deliverable 8, 1993, Document Number: MODEMA/WP6/DD08/FU
Jacob L. and Jones S. 1994 "Formal Dialogue Specification for Hypertext and Multimedia Systems", Technical report number 175, School of Information Sciences, University of Hertfordshire. Submitted for presentation at EWHCI 94.
Young R.M. & Barnard P. 1987 "The use of scenarios in human-computer interaction research: turbocharging the tortoise of cumulative science", Proc CHI '87