My lover and Rue

SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

INFORMATICS DIVISION

SOFTWARE ENGINEERING GROUP NOTE 111

Dimensional Design

Requirements Specification for Prototype, DRAFT 2.2

Issued by
D R Gibson

21 March 1986

DISTRIBUTION: R W Witty
D R Gibson
M Bertran-Salvans
T Povey, DEC
C Evans, DEC
R&D/DD/DEC file

KEYWORDS: SEGN 111 Dimensional Design

History

1 Original draft. *Author Duncan Gibson, 17th October 1985*

1.1 Restructured to use a different style of headings, etc. Revised to include ideas and comments made by Tom Povey. *Author Duncan Gibson, 30th October 1985.*

1.2 Revised after a technical evaluation at IOSG, 30th October. See also, SEG Note 80. *Author Duncan Gibson, 8th November 1985.*

1.2.1 Revised after a further technical evaluation on 11th November. *Author Duncan Gibson, 14th November 1985.*

1.2.2 Revised after comments from Rob Witty. After providing a Dimensional Design of the document, it was also restructured. An appendix relating to design issues added. *Author Duncan Gibson, 15th November 1985.*

2 Revised after technical meetings of 18th and 20th November 1985. *Author Duncan Gibson, 26th November 1985.*

2.1 Renamed,revised and added to after meeting with Tom Povey on 9th January. *Author Duncan Gibson, 16th January 1986.*

2.2 Slight revisions after meeting of 19th March. *Author Duncan Gibson, 21st March 1986.*

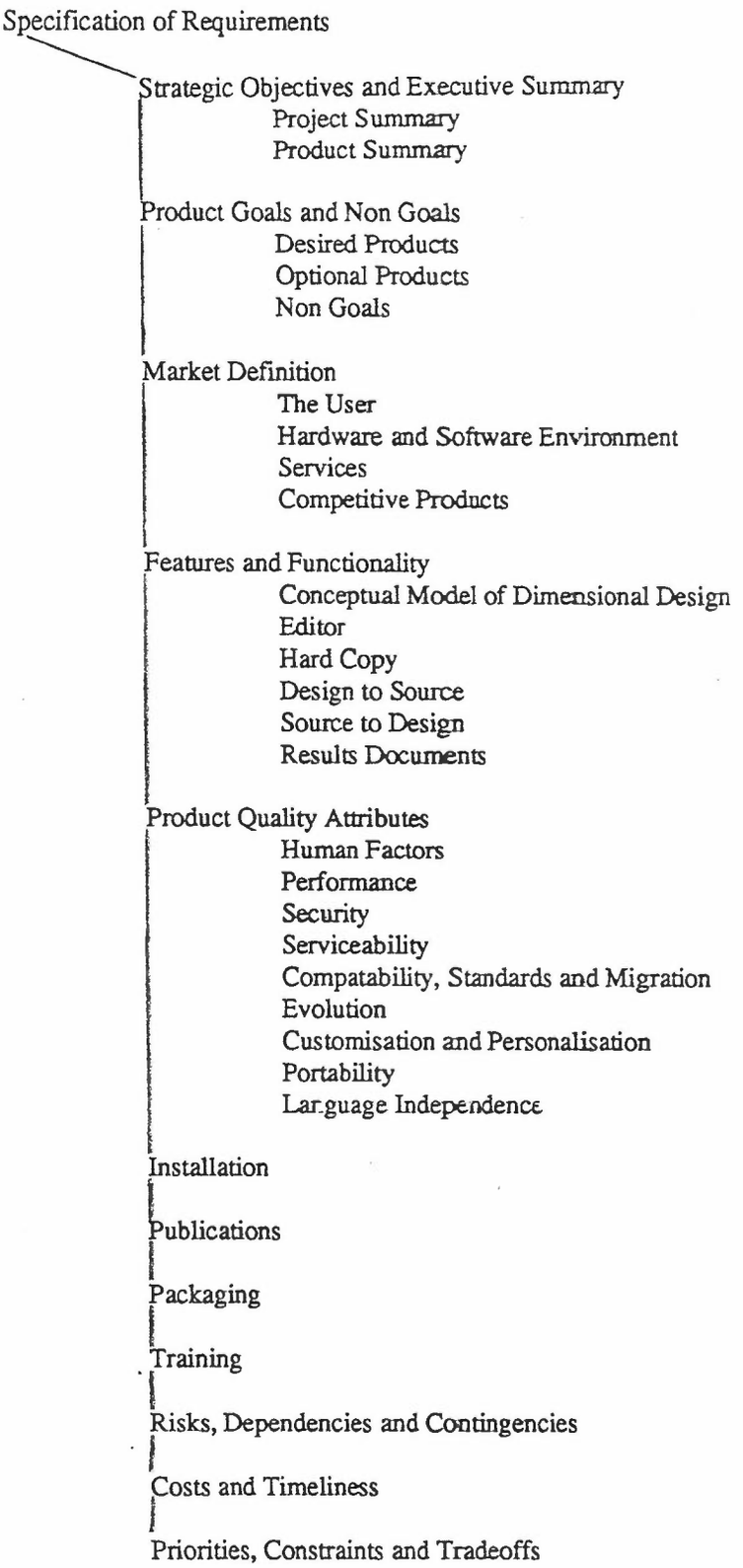

Figure 1: Dimensional Design showing the high level structure of the Requirements document.

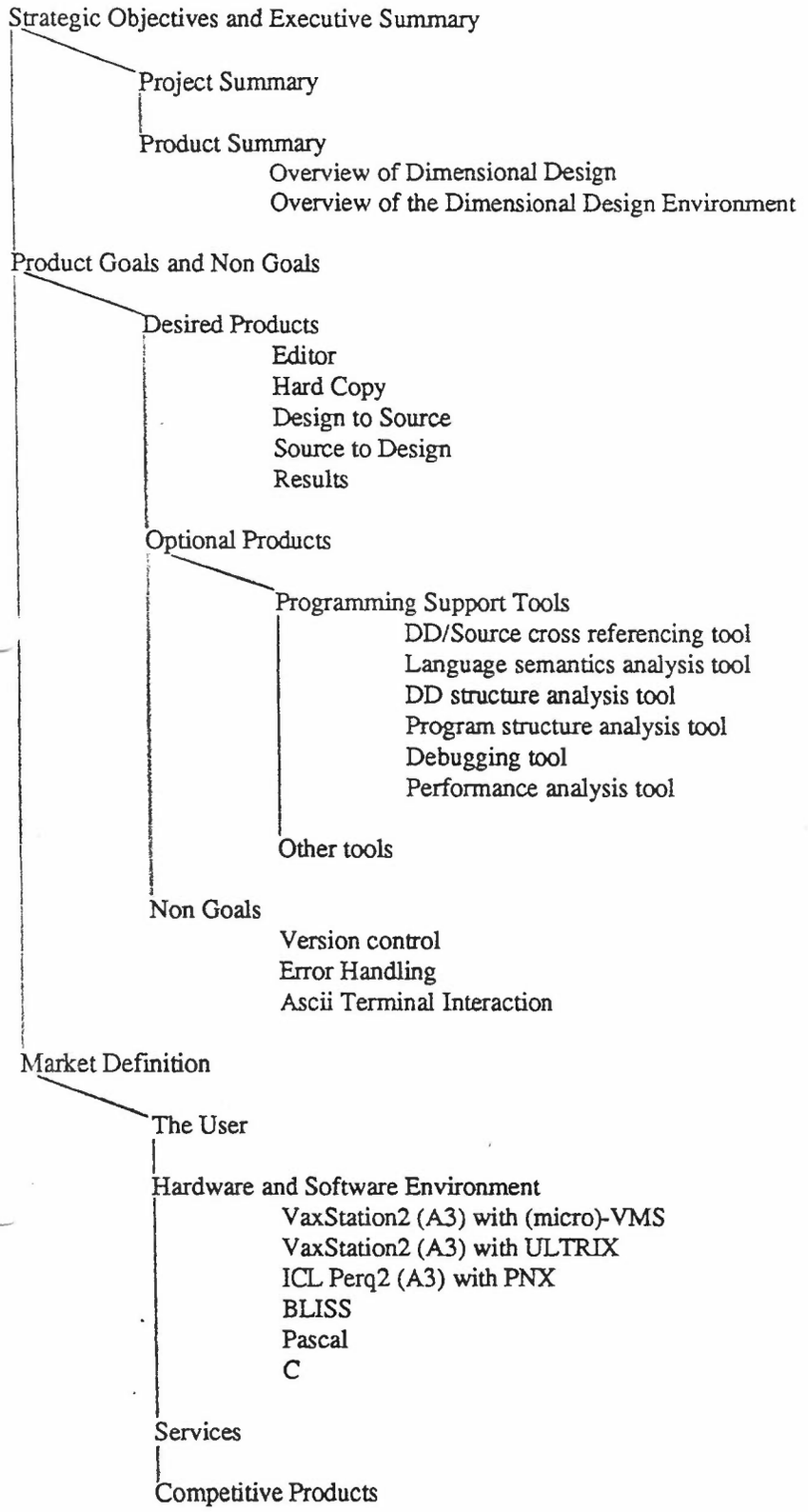

Figure 2: Dimensional Design showing the deeper structure of sections 1, 2 and 3 of the Requirements document.

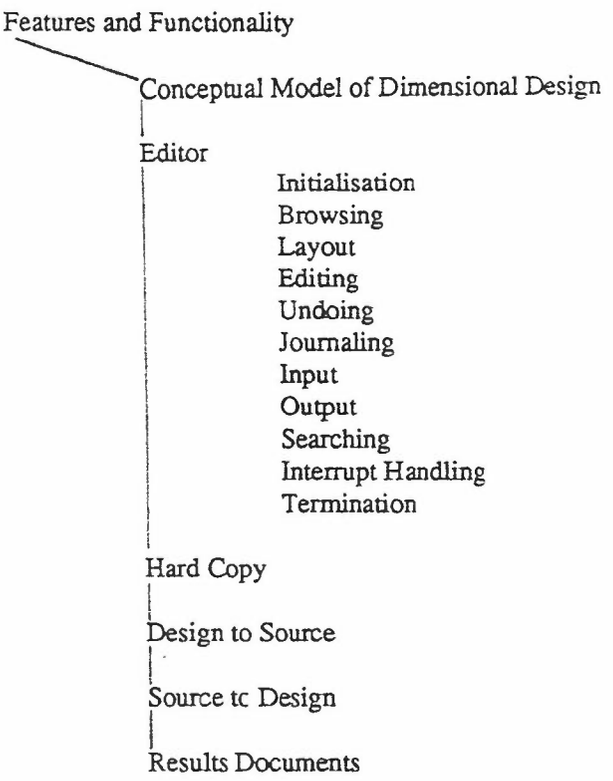

Figure 3: Dimensional Design showing the deeper structure of section 4 of the Requirements document.

Specification of Requirements
for a prototype
Dimensional Design Environment

* DRAFT 2.2 *

1. Strategic Objectives and Executive Summary

1.1. Project Summary

1.1.1. To produce the design tools, outlined in the Product Goals, which will make up a working prototype Dimensional Design Environment for in-house development work by IOSG.

1.1.2. To evaluate and study the use of Dimensional Design and compare it with other design techniques.

1.1.3. To provide a basis for a decision on whether to take the work on Dimensional Design further.

1.2. Product Summary

1.2.1. Overview of Dimensional Design

To be discussed.

1.2.2. Overview of the Dimensional Design Environment

The Dimensional Design Environment should contain a minimum set of tools to enable the design and maintenance of software using the Dimensional Design technique.

2. Product Goals and Non-Goals

2.1. Desired Products

2.1.1. *Editor:* a tree structured syntax driven editor, capable of dealing with multiple languages, which provides a Dimensional Design interface to the user. The editor must be able to take a syntactically correct "program source" as input, whereas a Dimensional Design given as input may be incomplete as far as the target syntax is concerned.

2.1.2. *Hard Copy:* a tool, or set of tools, for producing hard copy output of a Dimensional Design. A character cell printer is the minimum requirement; better quality devices are to be investigated.

2.1.3. *Design to Source:* a tool which will generate a program (or other, such as a text formatter) source file from a Dimensional Design.

2.1.4. *Source to Design:* since the editor is syntax driven, this conversion can be made by running the editor in "batch" mode.

2.1.5. *Results:* documents relating to the results of the evaluation and study of Dimensional Design.

2.2. Optional Products

2.2.1. Programming Support Tools

These are things which will aid the use of Dimensional Design as a software engineering tool. The tools can be divided into two main categories: those which will be used on program source code and are therefore *static* in nature, and those which will be used in conjunction with a running program and which are *dynamic* in nature. The set of static tools which could be provided might consist of

2.2.1.1. DD/Source cross referencing tool. This would enable a user to find the part of a Dimensional Design which corresponds to a given line of source, and vice versa.

2.2.1.2. Language semantics analysis tool. A tool which would check the semantics of generated source, for such things as undeclared variables.

2.2.1.3. Dimensional Design structure analysis tool. This would provide some statistics which may show the effectiveness of a particular Design, such as the average number of refinements per node.

2.2.1.4. Program structure analysis tool. A tool for gathering statistics about a program, such as the average number of statements per procedure, which could be compared with figures obtained for the structure of the Design.

The set of dynamic tools would be far more difficult to implement, but might contain such things as

2.2.1.5. Debugging tool

2.2.1.6. Performance analysis tool

where output or feedback is produced in terms of an annotated Dimensional Design.

2.2.2. Other Tools

To be specified.

2.3. Non-Goals

The tools produced during the course of this project should be viewed as prototypes, and a means of generating interest in Dimensional Design, as well as highlighting areas for further research. With this in mind, the following areas will not be investigated, or incorporated into the DDE tools.

2.3.1. *Version control:* exactly how different versions of a Dimensional Design, and any derived source code or hard copy, etc, are related is external to the Dimensional Design itself. A tool which will track version numbering will not be provided within the DDE.

2.3.2. *Error Handling:* when used in "batch" mode, the editor will fail at the first syntax error, if the input is not syntactically correct. However, in interactive use, the editor will notify the user that an error has been made, and then force the user to correct the error, before allowing the user to continue. However, Dimensional Designs which are not complete, but which remain syntactically correct by use of some meta-symbol which matches any missing sections for example, must be taken into account.

2.3.3. *Ascii Terminal Interaction:* The graphical tools of the DDE are not intended for interactive use from character cell terminals, such as a VT100. The tools are intended for use on a high resolution graphics workstation. This does not mean that tools cannot be run in batch mode from ordinary terminals.

3. Market Definition

3.1. The User

The Dimensional Design Environment is intended to be used by computer professionals, such as software engineers and technical secretaries, who are at ease with interactive tools. The user must be aware of some of the underlying architecture, such as the file system. The DDE is not aimed at non-technical staff who are not keyboard/mouse literate.

3.2. Hardware and Software Environment.

The Dimensional Design Environment is intended to be used on a high resolution graphics workstation which allows user interaction via a puck, or mouse. The graphical tools should run within any window management system which may exist. The DDE should also allow the use of a local (or possibly even wide) area network so that a distributed design team can share common resources. The possible environments are:

3.2.1. VaxStation2 with A3 landscape screen running (Micro)-VMS

3.2.2. VaxStation2 with A3 landscape screen running ULTRIX

3.2.3. ICL Perq2 with A3 landscape screen running PNX

The initial target machine is the VaxStation2 running (Micro)-VMS, but it is planned to use a VaxStation1 for development purposes. Both ULTRIX and PNX are proprietary versions of UNIX†

† UNIX is a Trademark of Bell Laboratories.

The DDE must be able to handle multiple languages, and so the project must aim to provide support for the following:

3.2.4. BLISS: for use within IOSG

3.2.5. Pascal: for use at RAL, and for the DDE development

3.2.6. C: for use at RAL (optional, if time permits)

The DDE, or rather the Hard Copy tool(s), must provide support for a character cell output device as a minimum requirement. Methods of using better devices, eg those with higher resolution or graphics capabilities have yet to be investigated.

3.3. Services

Not applicable.

3.4. Competitive Products

To be investigated.

4. Features and Functionality

4.1. Conceptual Model of Dimensional Design

For ease of discussion, let us consider the simplest conceptual model of a Dimensional Design as a tree where each node in the tree is either a single character or a subtree, and that it is enclosed within a cuboid. Nodes may be connected via arcs to show the structure of the Design, and its associated implementation.

Note: the actual model may differ from the conceptual one for performance or implementation reasons.

4.2. Editor

4.2.1. Initialisation

The editor must allow the user to create a new Dimensional Design, and also allow for the manipulation of existing Designs. To provide some degree of security, especially during the prototype stages, the editor may only read existing Dimensional Design files, it may not overwrite them. The user will be forced to create a new file for a new Design, or new version of a Design. The editor will not provide any form of version control or tracking however.

4.2.1.1. When it is invoked initially, the editor must establish the Dimensional Design which is being edited or viewed. This may involve providing a minimum empty framework within which to build a new design, or it may simply involve reading an existing file and taking some action on its contents. This action should occur as quickly as possible.

4.2.1.2. The editor must be able to establish a window into the Dimensional Design which will appear on the screen in some form. This action should occur as quickly as possible.

4.2.2. Browsing

The editor must allow the user to view the Dimensional Design which is being edited, and allow the user to move around within the Design. This is known as browsing, and does not produce any change in the underlying Design. All of the browsing actions should occur as quickly as possible.

4.2.2.1. The user must be able to pan through the Dimensional Design, ie change the location of the window into the Design.

4.2.2.2. The editor must allow the user to change the size of the window into the Design.

4.2.2.3. The editor may allow the user to zoom into/away from the details of the Design by using some scaling technique. This may be achieved by a reduction/expansion scheme, which is more properly dealt with in the section on Layout, or by changing the size of font, etc., displayed on the screen.

4.2.2.4. The user may be able to insert some placeholder into the Design, analogous to a bookmark, so that the user can move between such points with relative ease.

4.2.2.5. The editor should provide a means of showing the user where the current window is in the context of the overall design. This may be achieved by some sort of 2-D meter.

4.2.3. Layout

The functions outlined in the previous section simply allow the user to move around within the Design, without changing the underlying design, or changing the actual appearance of the Design. The user may wish to change the actual appearance of the Design, without changing its underlying structure, by altering the properties of the displayed Design.

4.2.3.1. The user must be allowed to change the properties of character atoms, such as the font used for display, or the character size, etc.

4.2.3.2. The user must be allowed to change the properties of the connecting arcs, such as length, thickness, etc.

4.2.3.3. The user must be allowed to change the properties of the enclosing cuboids, such as visibility, etc.

4.2.3.4. The editor must provide a means of displaying the Design in a more compact form by allowing for white space compression. This is graphical white space, as opposed to textual white space, and may be achieved by providing a set of alternative drawing algorithms, in addition to the standard drawing method.

4.2.3.5. The editor should provide a means of removing all textual information from the screen so that the user can see the structure of the Design as a skeleton of connected arcs.

4.2.3.6. The editor should allow the user to compress a node within the Design into some stub representation, and also allow for the expansion of such a stub, in order to remove screen clutter. This compression may be applied to terminal and non-terminal nodes of the Design.

4.2.3.7. The layout system must be able to handle common subtrees in some way.

4.2.4. Editing

The functions described in the previous sections will enable a user to move around within a Design, and to change its appearance on the screen. The most obvious application of an editor is to change an object!

4.2.4.1. The editor must allow the user to create a node.

4.2.4.2. The editor must allow the user to insert a node into the tree.

4.2.4.3. The editor must allow the user to delete a node from the tree.

4.2.4.4. The editor must allow the user to copy a node of the tree.

4.2.4.5. The editor must allow the user to move a subtree.

4.2.5. Undoing

The user must be allowed to undo an action. This is crucial if the user is to be allowed to experiment within the editor, without being frightened of losing several hours of work by exploring new possibilities. An undo function also provides the user with some recovery after an inadvertent button press or key stroke. As this is one of the most difficult functions to provide, the prototype editor will only have a "best attempt" at recovery.

4.2.6. Journaling

The editor may provide a record of operations, or journal, so that the following functions can be achieved:

4.2.6.1. Playback: the user or, more importantly, a more expert user can replay part or all of an editor session to see why the editor produced the results that it actually did, which may not have been what was expected intuitively. In the prototype version this can be used to aid troubleshooting.

4.2.6.2. Demonstrations: the features and capabilities of the editor can be shown to an audience for training, or exhibition purposes.

4.2.6.3. Checkpointing: the editor may save the state of the session at various points, and all journal operations are recorded relative to the last saved state.

4.2.6.4. Macro facility: if it is possible to generalise the journaling function, it may then be possible to

provide a macro system.

Clearly, if it is possible to reverse the journal of recorded events, then this would provide one method of implementing the undo action. It may also then be possible for the user to undo an undo action, ie to play an event through backward and forward, several time over if need be.

4.2.7. Input

4.2.7.1. The editor may be used in batch mode, as well as interactively, but this document will not say how the interaction will be made, what form of commands are used, or screen designs, etc.

4.2.7.2. The editor must be able to handle input of syntactically complete sources, eg program sources which will compile, but it must also handle Dimensional Designs which are incomplete as far as the same compilers are concerned, but which maintain their integrity by use of some meta-symbol for example. The editor may need to handle input which has been generated via a journal facility, if there is one.

NOTE: The format of the Dimensional Design file, and indeed the journal file, are design decisions. However, there is a requirement that these files be in a form suitable for transmission and exchange between systems and tools.

4.2.7.3. The editor should allow the user to work on more than one Dimensional Design at a time. Multiple buffers should be taken into account during the design phase, although the prototype need only provide a single buffer.

4.2.7.4. The editor must be able to handle Dimensional Designs which may span several files. For example, a software project may have been partitioned into several distinct modules, each of which has its own Dimensional Design. The overall Design may contain references to these modules. This may be achieved using an "include" mechanism. The user should be able to expand/compress such references. These references may occur more than once in the overall Design, and may give rise to a common subtree (see Layout).

4.2.8. Output

See above.

4.2.9. Searching

The editor must allow the user to search for an item or items which exhibit a particular set of properties. In this instance, an item may mean an arc, node, character atom, cuboid, some means of generating a particular subtree structure, etc. Possible properties of items are outlined elsewhere. Since the editor is syntax driven, the user must also be able to search for syntactic entities.

4.2.10. Interrupt Handling

The editor must be able to handle user level interrupts in a sensible fashion. The user may want to abort or cancel an operation using an interrupt, so the editor must take the appropriate action.

4.2.11. Termination

4.2.11.1. Under normal conditions, the editor should make a series of checks before it terminates, and ask the user for confirmation of intent depending on the results. For example, if the user modifies a Dimensional Design during the course of a session, and tries to quit without writing the changes back to file, then the editor should prompt for confirmation.

4.2.11.2. If the editor terminates abnormally for some reason, then it should provide some failsafe action, if possible, such as writing the modified Designs into files.

4.3. Hard Copy

4.3.1. The hard copy tool must be able to take a Dimensional Design file as input and produce output which will cause a printer, or plotter, to "draw" the Dimensional Design.

4.3.2. The hard copy tool must be able to produce a standard version of the Dimensional Design, with all nodes and included references expanded, etc.

4.3.3. The user should be able to specify that the hard copy tool produces a listing of the Dimensional Design in the same form as it appeared on the editor screen, ie with appropriate nodes compressed, or having visible cuboids, etc.

4.3.4. The hard copy tool should be able to print a subtree of the original Design if that is what the user selects, by some marker in the Design possibly. The obvious default action is to draw the whole tree.

4.4. Design to Source

4.4.1. This tool must take a complete Dimensional Design, ie one without any meta-symbols which may be present during the editor stage to maintain syntactic integrity, and produces output which meets the target language syntax rules. For example, a Dimensional Design of a program should give rise to program code which satisfies the syntax analysis of the compiler, even though it may not actually compile for semantic or other reasons.

NOTE: A Dimensional Design may have target language dependencies which mean that it cannot be retargetted for another language automatically!

4.4.2. The Design to Source tool is intended to be a batch system, even though it may be initiated interactively.

4.5. Source to Design

This may be achieved by using the editor in batch mode. The source file must be syntactically correct though.

4.6. Results Documents

These should be presented as a series of evaluation reports which discuss:

4.6.1. the use of Dimensional Design in the development of the DDE;

4.6.2. the results of controlled experiments into how designers use the DDE;

4.6.3. the results of the use of Dimensional Design in a project, especially when compared to similar methods in use in other projects.

5. Product Quality Attributes

5.1. Human Factors

5.1.1. The DDE should be easy to use by a professional programmer.

5.1.2. The DDE should provide some basic help facility, or even some computer based instruction.

5.1.3. The prototype design should allow for the inclusion of measurement metrics, for evaluation purposes.

5.2. Performance

5.2.1. When used interactively, the editor must be able to keep up with the user. Those tools which are to be used in batch mode may take significantly longer.

5.2.2. The editor should present some screen based feedback to the user within about 4 seconds. Obviously, this will depend on the design decisions. During use, the editor should provide feedback indicating what is happening, or what actions are available, as soon as possible.

5.3. Security

5.3.1. The VMS version of the DDE must do nothing to jeopardise the VMS file security, ie the DDE tool set only has the default user priviledges.

5.3.2. Similar limitations apply to the UNIX version of the DDE.

5.4. Serviceability

5.4.1. Availability: not applicable.

5.4.2. Reliability: not applicable.

5.4.3. Maintainability: the project itself should use Dimensional Design to ease maintenance of the software. Coding standards should be adhered to wherever possible.

5.4.4. Auditability: the prototype should provide an option which allows a skilled user to track the behaviour of the tool.

5.5. Compatability, Standards and Migration

5.5.1. The DDE must accept files generated using any existing Dimensional Design related software. A group at the University of Linkoping in Sweden produced the Dimsys software. Research work is also being carried out at the School of Telecommunications of the Polytechnic University of Catalonia in Barcelona. This does not mean that the DDE should be tied to the file format that either of these groups use: the use of a Dimsys file, or one from the Spanish system, may be achieved using some translation tool.

5.5.2. The project must investigate possible formats for DDE files, and also how various languages are represented and generated. (There is a risk of initial formats being accepted as the de facto standard which will then prove difficult to change!)

5.6. Evolution

The design of the prototype should allow for the evolution and addition of further features, as discussed in sections 2 and 4.

5.7. Customisation and Personalisation

The user should be free to change any language definition and provide additional constructs as and when required. However, such changes may need to be made for every stage of rule generation, input parsing and code generation.

5.8. Portability

As the DDE is ultimately intended to run on 2 machines, and 3 operating systems, then portability issues must be considered at every stage of the design and implementation.

5.9. Language Independence

In order to keep the DDE independent of any particular natural language, such as English, all textual input and output undertaken by the user interface must be parameterised. All messages used by the DDE should have a unique index, so that the text used by any particular language may be held in a separate message file, rather than hard coded into the system.

6. Installation

6.1. The VMS version of the prototype should use VMS Install.

6.2. The installation of the system under UNIX has yet to be investigated.

7. Publications

The project should aim to produce user guides of some sort, where the minimum set is

7.1. "A Guide to using the DDE" and

7.2. "A Quick reference card for the DDE"

8. Packaging

Not applicable.

9. Training

The users taking part in the evaluation must receive some training and support from the project team.

10. Risks, Dependencies and Contingencies

to be discussed.

11. Costs and Timeliness

to be discussed.

12. Priorities, Constraints and Tradeoffs

to be discussed.

Appendix Z: Issues

The issues which were presented as part of the requirements specification in previous drafts are being used as the basis of a high level design document, and so will appear elsewhere.