User Interfaces in Space Science Instrumentation

Alec John McCalden

Mullard Space Science Laboratory
Department of Space & Climate Physics
University College London

A thesis submitted to the University of London for the degree of Doctor of Philosophy

July 2006

Statement of Originality

I confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Alec John McCalden

Abstract

This thesis examines user interaction with instrumentation in the specific context of space science. It gathers together existing practice in machine interfaces with a look at potential future usage and recommends a new approach to space science projects with the intention of maximising their science return.

It first takes a historical perspective on user interfaces and ways of defining and measuring the science return of a space instrument. Choices of research methodology are considered. Implementation details such as the concepts of usability, mental models, affordance and presentation of information are described, and examples of existing interfaces in space science are given.

A set of parameters for use in analysing and synthesizing a user interface is derived by using a set of case studies of diverse failures and from previous work. A general space science user analysis is made by looking at typical practice, and an interview plus persona technique is used to group users with interface designs. An examination is made of designs in the field of astronomical instrumentation interfaces, showing the evolution of current concepts and including ideas capable of sustaining progress in the future.

The parameters developed earlier are then tested against several established interfaces in the space science context to give a degree of confidence in their use. The concept of a simulator that is used to guide the development of an instrument over the whole lifecycle is described, and the idea is proposed that better instrumentation would result from more efficient use of the resources available.

The previous ideas in this thesis are then brought together to describe a proposed new approach to a typical development programme, with an emphasis on user interaction. The conclusion shows that there is significant room for improvement in the science return from space instrumentation by attention to the user interface.

Contents

List of Figures

Chapter One: Science, Machines, Users and Methodology

Chapter Two: Exploring Interfaces

Chapter Three: Human - Computer Interface Case Studies

Chapter Four: Interface, Goals and User Analysis

Chapter Five: Usability Criteria Verification

Chapter Six: A Common User Interface?

Chapter Seven: Use of an Instrument Simulator

Chapter Eight: A Possible New Approach

Appendix C: Full Persona

Appendix D: Interface Assessment Tables

List of Tables

Chapter Two: Exploring Interfaces

Chapter Three: Human - Computer Interface Case Studies

Chapter Four: Interface, Goals and User Analysis

Chapter Five: Usability Criteria Verification

Chapter Six: A Common User Interface?

Chapter Eight: A Possible New Approach

Appendix B: Interview Details

Appendix D: Interface Assessment Tables

Acknowledgements

Writing this thesis has inevitably required aid, support and encouragement from those around me, and I express without reservation a sincere thanks to all those who have given an opinion, said a few words of encouragement, been critical for my own good, allowed me to interview them, put precious time into reading my efforts, or simply tolerated my excuses again for not finding time to be with them. There are of course several people who have been much closer to the whole work, and I must specifically thank my supervisor Professor Alan Smith for his continuing support. I thank also Professor Mark Cropper and Dr. Graziella Branduardi-Raymont for their time, helpful comments and support as academic panel members. Other thanks go to Professor Angela Sasse for an intellectual push in the right direction, and to Dr. Tony Newton and Gillian McCalden for taking time to proof read the evolving tome so thoroughly. Matt Whyndham and Matt Whillock also were generous with their time to give me valuable information and feedback. Most importantly I must not forget my parents, whose tolerance of my early curiosity in wondering how things worked has results in front of you today.

The work has been made possible by a grant from EPSRC, made available through MSSL, for which I am very grateful.

Glossary of Selected Terms

The definitions of selected terms used in this thesis are given here. They are compiled with help from the Oxford English Dictionary, and with regard to the context in which they are used.

affordance: the interface property of an object.

CDS: Coronal Diagnostic Spectrometer.

check box: an interface toolbox item that allows selection or deselection of a property.

code stub: a short, temporary section of software that has interfaces but does very little.

command line interface: system control performed by an operator using a keyboard to type strings of text in response to a screen prompt.

CPM-GOMS: Critical Path Method - Goals Objects Methods and Selection rules.

design model: the mental model with which the designer of a system works.

EIS: Extreme-ultra-violet Imaging Spectrometer.

EPSRC: Engineering and Physical Sciences Research Council.

ergonomics: the scientific study of man in his working environment.

ESO: European Southern Observatory.

Fitts' Law: empirical law relating user reaction time to separation and size of objects in a user interface.

front loading: system engineering term describing the move of activities to early in a schedule.

glass cockpit: common term referring to, for example, an aircraft cockpit with all-electronic displays and controls. The term is used to make a distinction from mechanical indicators and controls.

GOMS: Goals Objects Methods and Selection rules. A numerical method of analysing a keystroke-based interface.

group box: an interface toolbox item visually grouping other items together.

GSFC: Goddard Space Flight Center (NASA).

haptic: the sense of touch or tactile sensation.

HCI / CHI / MMI: synonymous terms meaning Human Computer Interface, Computer Human Interface, and Man Machine Interface.

heuristic: loosely defined, as in a rule or guideline.

Hick's Law: empirical law relating the number of choices against the selection time for a user interface.

interface: the common point of interaction between a computer and its user.

left shift: synonymous with 'front loading'.

mental model: the mental visualisation by a user of the system with which they are interacting.

MSSL: Mullard Space Science Laboratory, University College London.

MVC: Model-View-Controller, a software architecture for separating data from the user interfaces to the data.

NGOMSL: Natural Goals Objects Methods and Selection rules Language.

object oriented: a software method using encapsulation (enclosure) of software routines to form well-defined interfaces to those routines. Frequently used for the definition of display screen objects.

popup menu: single screen object which expands into a menu selection when clicked.

push button: button on screen which performs a function when clicked. May be momentary or latching action.

radio button: latching push button, one of a set of two or more. Clicking one de-activates the others.

SCRAM: emergency procedure to inhibit nuclear fission by inserting control rods. Generally accepted as the acronym from Safety Cut Rope Axe Man, but variations exist.

SEA: Scientist's Expert Assistant.

separator: a line providing visual separation between screen objects.

SGM: Science Goal Monitor.

short term memory: human memory capable of recalling items for up to about 30s.

slider: screen control capable of producing a range of values. Usually drawn as a straight line rather than curved.

SOAR: SOuthern Astrophysical Research telescope.

SOHO: SOlar Heliospheric Observatory.

TMI: Three Mile Island.

system image: the implementation in a working system of the mental image from the designer.

telemetry: data flow over a communication channel.

text input field: area on a screen reserved for user entry of text.

toolbox: software-vendor supplied collection of software routines for producing common screen objects.

usability: ease of use; capability of use.

user: a person with an interest in operating a system, either directly or through a third party.

user feedback: output from a system as observed by the user.

user model: the mental model with which the user works.

utility: the quality of being useful; fitness for purpose.

white space: any unused space on a display screen; it is not required to be white.

WIMP: Window, Icon, Menu, Pointing device. The current default for a desktop computer installation. Variations on the acronym exist.

XMM: X-ray Multi Mirror.

Chapter One: Science, Machines, Users and Methodology

1 Introduction

Ever since mankind upgraded from counting pebbles in the sand to beads on a string, there has been the drive to understand better how to control the machines we build. In the field of astronomy, simple unaided star gazing moved to the use of telescopes. Continuing demand put those telescopes in remote places at high altitude, first on *terra firma* and then beyond, turning ideas of remote control of those machines from convenience to necessity. It is one argument of this thesis that the means of communication with astronomical machines has not kept pace with the available technology and the developments in the cognitive sciences. This leads to reduced utilisation of the telescope compared to an ideal, and the implied waste of scarce resources. The communications referred to here are not those using the impressive radio networks that we operate allowing data transfer over vast distances, but rather those over the final few centimetres between user and machine. The difficulty is to produce a mechanism that accurately translates the goals of many classes of user, over an entire instrument lifetime, into a complete system that fulfils as many of those goals as possible. The overall goal is to optimise the science return of the system.

The aim of this thesis can be summarised by one hypothesis, to be proven or otherwise:

The science return of current space instruments is constrained by the user interface and could be improved by a new approach.

where:

science return	refers to the quality and quantity of the science data returned to the user
current space instruments	sets the scene at the time of writing
constrained by the user interface	implies current user interfaces are not optimum
be improved	requires a system of metrics
a new approach	implies fresh ideas, at least in the context of space science

2 Background

We start by looking at an historical perspective. One does not have to go back further than 1990 or thereabouts to find an era before the availability of the current proliferation of electronic displays and their indirect, multiple mode controls in many fields of electronics-based systems. The transition into a new way of working was caused by the growing availability of readily affordable computers and software packages such as the Apple Macintosh OS, the Borland C compiler and Microsoft Windows. These allowed easily reconfigurable display screens and as many virtual controls as the designer wanted. Hitherto, each system variable would have its own physical control, and the placement of that control would be dictated by three main factors:

- any electrical constraints caused for example by the requirement of short connections between it and the system circuitry
- any mechanical constraints due to structural requirements
- any placement constraints due to operational requirements

Placement would be influenced by ergonomic arguments, such as grouping controls of a similar function together, but frequently the debate between circuit engineer, mechanical designer and front panel graphics designer would have to be settled by compromise. If this was too much in favour of the graphics designer, the equipment would have inferior performance or be expensive to fabricate; if the solution was too much in favour of ease of manufacturing it would not sell well, being difficult to operate or looking unattractive. Commercial criteria are used as an example here as it should be clear that there would have been a real incentive to take into consideration the ergonomic points of the design. The space science world was rather more biased towards functionality and this worked acceptably as system experts typically operated the equipment. There was a steep learning curve, but this was expected and allowed for.

In the commercial world, as digital techniques and the use of embedded software have become more ubiquitous, the requirement for a close physical association of a control and the mechanism that it controls has all but disappeared. This has led to imaginative use of display space and interaction with controls, some of which may change function depending on the task required.

The multiple function controls on the Sony Ericsson mobile phone in Fig. 1 illustrate this point. The number keys can represent at least four characters each, whilst the function of the 'yes' and 'no' keys depends on the contents of the display panel above them. In the space science world, where there appears to be little perceived need - and therefore no budget - for specialists to specify the contents of display screens, the results have tended to fall towards the functional approach as perceived by the programmer putting the system together. This is fine, except that it does not take into account the needs of other users of the system. This is a large topic for a later chapter, but it can be illustrated now by considering the differing needs of a system programmer and of a first time user. The system programmer knows the details of the system extremely well, wants maximum speed of interaction with it, and is very happy with controlling it with a command-line interface and looking at dense fields of scrolling text for expected results. A first time user may simply give up at this point, gesticulate in frustration and demand the manual and sufficient time to read and extract the needed information. Then he (or she) will try out an action, make a mistake, and go back to the manual again. Now suppose the display he's looking at has simple graphical elements that help explain their function and prevent hazardous values being set. He will have a much better chance of achieving the task at hand. This simple graphical display is unlikely to suit the systems programmer though, who is likely to find it slows him up. The two people are both valid and required users, but need careful consideration of their individual roles. One method of operating the instrument *may* be sufficient, but that fact needs to be explicitly shown. A methodology of realising this analysis is shown in Chapter Four.

Fig. 1 Sony Ericsson phone

3 Terminology

Ergonomics is the European term for what is known as human factors engineering in the USA, and is variously defined as the science of people-machine relationships, or of people and their environment. One term in particular has become so pervasive in this field as a major subset of ergonomics that it makes sense to introduce it now. The term is Human Computer Interface, widely abbreviated to HCI and sometimes CHI, and the former

abbreviation will be used here. In the context of this work, *human* is self explanatory, *computer* is any machine or instrument being operated, and *interface* is the communication medium between them.

Another term frequently used is Man Machine Interface, or MMI. It implies the same area of interest.

4 Structure of this thesis

Chapter One sets the scene for this thesis, explaining why a consideration of HCI techniques is important, and setting the space science context in which the work has been carried out. It introduces the idea of people-first design, and discusses the different research methods which have been considered.

Chapter Two examines in some depth existing techniques for the design of interfaces in general, and illustrates recent practice in space science with screenshots covering several instruments. Details of exploratory programming work are also described.

Chapter Three takes five well documented examples of system failures in various fields, and looks at them again in the light of bringing out key human interface issues.

Chapter Four tackles the analysis of goals and users for a space instrument, looking at the requirements over the whole lifecycle and the range of users involved. A comprehensive set of usability criteria is developed from the results of the previous chapter and elsewhere.

Chapter Five tests the usability criteria developed earlier against six established interfaces to establish a measure of confidence in their use.

Chapter Six looks at what might be involved in standardising user interfaces for space science applications, describes work elsewhere on this topic, and analyses what can be learnt from this in the design of new systems.

Chapter Seven explores the typical organisation of an instrument design and development programme, and a new approach over the whole lifecycle including the use of a concurrent simulator is suggested.

Chapter Eight takes the outcome of the previous chapters and shows how a work flow might be developed to take all this information into account.

Chapter Nine has a set of conclusions and ideas for further work.

A full bibliography follows this, with specific references and additional material.

Appendix A has the results of running the programming work described in Chapter Two, and a listing of the contents of the CD on the back cover.

Appendix B contains more details of the interview results described in Chapter Four.

Appendix C contains an illustration of a full persona build also from Chapter Four.

Appendix D contains templates for forms with which to assess an existing space science system, taken from the work in Chapter Five. The check list from Chapter Eight is included as well.

Use in the text of *him* and *he* equally implies *her* and *she.*

5 Science Return

This work sets out to show that the science return from an instrument could be enhanced by attention to the user - machine interface. We need to attempt a definition of 'science return' in order to show the scope of the subject. This could be described as the 'quantity of science information' that is delivered of an acceptable or minimum quality. Given the limited time that is available on any given instrument, this translates to the 'rate of science information delivered'. An instrument that for example takes less time to operate, or tends to allow fewer incorrect configurations, or makes best use of the allocated time for a particular user, would seem to be likely to have an enhanced science return compared to an arbitrary norm. The term 'usability' is commonly used to describe parameters such as the three examples here. The real difficulty is finding a mechanism for grading levels of usability and this is covered more in Chapter Six.

One way of measuring science return might be to count the number of science papers that quote results from a given instrument. This has the attraction of a simple numerical approach. However, a basic counting technique takes no account of the quality of each paper.

It would be possible to quote the number of citations of each paper, giving a more proportional measure of the usefulness of each. It is likely that the typical rate of citations would mean that a period of years would be necessary before any result could be described as statistically significant. In that time it is likely that fresh ideas may have made the original approach invalid anyway; any HCI assessment needs to take place over the period of a few weeks at the most in order to be useful in a typical instrument development programme.

Another difficulty of using citations is the influence of other factors such as simple popularity of one research topic over another, where difference in popularity between two instruments would completely swamp the effects due to the merit of one particular instrument design.

The data analysis phase of writing a paper can vary greatly in time from maybe a few days to many months, also skewing the rate of production of papers and hence their citation rate. The rate of production of papers and their citations may possibly have *some* part in a scoring framework that gives different weightings to different methods of assessment of science return, but the method is seen as too uncertain to be tackled in this thesis. Matrices for determining the impact of an individual paper are in fact used in other contexts and the term “impact factor”, used to measure the quality of a scientific journal, is closely related to the use of citations.

Yet another way of describing science return might be to consider how effectively the instrument has met its design specifications and how appropriate those specifications were. Most if not all projects exist in a field of limited resources and overspending on one will have a knock-on effect on others and may reduce the science return from the overall science programme. The goal of meeting specifications that are themselves appropriate is just as relevant to maximising the science return when considered over the mission lifetime, and Chapter Five analyses the actual implementation of several tools for various space instruments.

The quality of the science data is just as important in assessing science return. To take three examples, the instrument must be calibrated accurately, the mechanisms must work reliably with predictable positions, and the entire communication system must ensure minimal missing data. In turn this imposes high requirements on all the instrument systems and a need for accurate reporting and control of those systems. This could partially be achieved by more built-in automation. For example, we all take it for granted - if we even realise that it is happening - that a computer hard disk drive re-calibrates its track seeking mechanism every few minutes to cope with changes in temperature and mechanical wear. The operation is totally transparent to the user, does not rely upon intervention from the operating system, and is a major contribution to the perceived high reliability of hard drives. The same philosophy of an automatic feedback mechanism could be employed to keep astronomical instruments calibrated.

In a resource-constrained world, one further way of looking at science return is to consider how efficiently instrument components are produced given a fixed timescale and size of workforce. Re-inventing components that are identical to those employed elsewhere is clearly inefficient, although care must be taken not to under-estimate the work required to effect apparently small changes to existing parts.

This thesis mostly uses the first concept, that of science return being dependent on usability, as the main intention is to explore various human computer interface aspects. The issue of

specifications is tackled in Chapters Seven and Eight by suggesting a new methodology, and the concept of re-usability through standardisation is explored in Chapter Six.

6 System Engineering Issues

It is appropriate to touch briefly on the formal constructs that hold a complex engineering project such as a space instrument together in a programmatic sense. Without some level of formal control and documentation, there is a likelihood that the science return will be diminished as user needs raised early in a project fail to be carried through to the finished instrument. Cancellation of the whole programme and loss of the funds spent is one possible outcome of an inadequate methodology.

From the point of view of project organisation, a key first stage is the understanding and documenting of the needs of the users. Some method of selection and prioritisation is then applied to this list of needs, as some will be more important than others and overlaps are likely to exist. Some may well end up on a wish list of functions that may be implemented only if subsequently shown technically or financially viable. Out of this list of needs comes a set of formal requirements, each of which is a clear statement of one facet of the instrument project. A complete set of requirements in systems terminology is referred to as a specification; it defines the instrument at that point in time.

The two major space agencies of ESA and NASA each have their own view of how this process should flow. The ESA approach is based on an initiative called the European Cooperation for Space Standardization (ECSS), and the relevant document (ESA 2005) defines two levels of specification:

Functional Specification (FS)	the “baseline for investigating and comparing candidate concepts”.
Technical Specification (TS)	the “baseline of the business agreement to develop or purchase the selected solution”

The FS is defined first and used to refine and make decisions about which options to take. Once the project programme is decided, the result is defined in a TS. The latter is a precise, definitive statement about the instrument design.

The NASA approach is contained in their System Engineering Handbook (NASA 1995). Projects are divided into six phases:

Pre-phase A	Advanced studies
Phase A	Preliminary analysis
Phase B	Definition
Phase C	Design
Phase D	Development
Phase E	Operations

A phase is not started until the previous one is finished.

Numerous books cover this area of requirements and specifications. One example is by Suzanne & James Robertson (Robertson 2006), and usefully also refers to the persona process described in Chapter Four of this thesis.

7 Selection of Research Methodology

This work involves a mixture of quantitative and qualitative methods. Some of the work is subjective, or otherwise observer dependent, and has the target of evolving into a numerical method which allows comparison of one piece of information against another. The topic of method choice is so large that the discussion here can only scratch the surface; however, the author feels it important to at least be aware of the main aspects of a qualitative approach in order to be able to plan the gathering and analysis of experimental data that are expected to be low in numerical content in the raw state.

Why attempt research that is not quantitative? After all, surely everything can be reduced to numbers?

Guba and Lincoln (Guba 1994) suggest an approach to this question using the four paradigms of positivism, post-positivism, critical theory, and constructivism. Myers' work (Myers 1997) has been used as a starting point in this thesis; he tackles the issue by following Orlikowski and Baroudi (Orlikowski 1991), and Chua (Chua 1986), and adopting three different philosophies for research work: Positivist, Interpretive, and Critical. This thesis uses a combination of these methods.

7.1 Definitions

Positivist: an objective, numerate approach independent of the observer and the instruments. It is characterised by evidence of formal propositions, numerical measurements, testing of hypotheses and drawing conclusions.

Interpretive: Assumes that all research must be done through social constructs such as language, consciousness and shared meanings. The background is one of the understanding and interpretation of texts (hermeneutics), and of the study of objects and events as they appear in human experience (phenomenology). Interpretive research tends to allow the definitions of variables to emerge as the work progresses, rather than pre-define them.

Critical: Assumes that events are historically linked, and are produced by people. Although these people may act to change a situation, their ability to do so is frequently limited by social and cultural bounds. The main application of critical research seems to be in the study of contemporary society and the process of emancipation in that society.

A first reaction to these definitions in the context of this work must be that a so-called positivist approach is the concept to follow. The ideas of numerical analysis and objectivity are familiar to many researchers, and popular in the sphere of space science. The questions to ask are what do the other ways of thinking bring to the problem, and what is missing so far. One obvious missing item is 'people'. If one is concerned with the interaction of people and machines, then a study of machines alone cannot produce the desired result. The concept of 'critical' research at first seemed dismissible; good for historians but hardly high technology. But individual emancipation is important in this context! One aspect of this study is to show how people might be put in full control of instrumentation, rather than have their wishes be brushed aside every so often by a piece of inadequately designed technology. Similarly, for the idea of 'interpretive' research, the study of events and objects and the concepts of language are important in the context of instrumentation control.

Therefore, although at first it seems useful to put labels on different approaches to research, the cross disciplinary approach implied in this thesis means that all three philosophies are relevant. Their attributes are a useful reminder of points to consider, but at this point that seems all.

7.2 Research Strategy

Myers (Myers 1997) takes the view that the researcher needs to choose one philosophy of the three. He then deals with the choice of research method as a strategy of inquiry and suggests four variants:

Action Research - There are differing definitions of action research, but one from Rapoport (Rapoport 1970) emphasises collaboration, ethics in carrying out the work, and contribution to knowledge both in the immediate context and in a science community context.

Case Study Research - A case study method implies experimental work on a phenomenon within the context of the real world.

Ethnography - Ethnographic research uses the technique of the researcher immersing themselves in the situation under investigation by spending a significant amount of time working closely with the subjects of the research.

Grounded Theory - A grounded theory method emphasises the need to have a continuous interaction between data collection and analysis, and takes the name from the idea of developing a theory that is firmly grounded in data systematically gathered and analysed. It is often described as using an inductive approach. The two original proponents of the concept, Glaser and Strauss (Glaser 1967), have since diverged significantly in their interpretation of the exact methodologies to be used, giving a sharp reminder of the difficulties in defining approaches to qualitative research. The adopted approach is described in section 7.5 below.

7.3 Analysis of Data

The common aspect of (virtually) all qualitative work is that the data comes from textual analysis, either written or spoken, rather than coming direct from instrumentation as in the case of quantitative work. It is likely to be essentially non-numeric. Sources may be documents and interviews in the situation of a case study, or direct observation of participants engaged in the activity under examination in the situation of an ethnographic approach.

In quantitative work it is perceived to be easy to separate the two processes of data collection and data analysis - analysis happens after the collection. This statement is in fact only partially true, as since the days of early school science experiments we all have been encouraged to plot data as it is taken to enable trends and errors to be spotted immediately, and corrective action taken if necessary. For much qualitative work, the emphasis may be the other way around - the questions put to participants significantly determine the data to be revealed. The answer to one question is then very likely to affect the next question. Myers highlights three approaches (Myers 1997) of dealing with this situation:

Hermeneutics - In this context, hermeneutics is defined as the interpretation of textual data. This can frequently seem confusing, contradictory or incomplete and the task of interpretation is to bring out coherent information. The hermeneutic method looks at the whole collection of data from the one source, as well as the individual parts of that data, in order to interpret it.

Semiotics - An an analysis method, semiotics is concerned with the meaning of signs and symbols in language. Such words and signs can be equated with essential concepts of the theory under test, and the frequency of occurrence is a measure of the relative importance of that concept. One form of semiotics deals with content analysis where the text is searched for structures and patterns and inferences made. Another form is conversation analysis where meaning is brought out from the context of the exchange. A third form is known as discourse analysis, which includes aspects of content and conversation and includes a sequence of verbal moves in which turns of phrases, use of metaphor and allegory, are all used.

Narrative and Metaphor - In promoting understanding between designers and users of a system, story telling (narrative) and comparison (metaphor) can be useful in putting one context in the realm of understanding of a group used to a different context.
Anthropomorphism - giving a machine human characteristics - is used particularly with computers. The method can assist analysis by enhancing understanding.

7.4 Key words

The following sets of key words have been found useful as an alternative method of thinking about the aspects of quantitative and qualitative research:

Quantitative - numerical, precise, defined, algebraic maths base, natural sciences, objective, prediction and control, independent of observer, counting, narrow, verification, confirmatory, narrow focus.

Qualitative - wide ranging, non-numeric, statistical maths base, social sciences, subjective, explanation and understanding, observer dependent to some extent, meaning, broad, discovery, explanatory, holistic.

7.5 Adopted Approach

The approach adopted by this work borrows from several areas of this description of research methods and is described here in order of the presentation of the methods above. A philosophy of critical research, and a strategy based on case studies, are used in Chapter Three to bring out details from real life incidents by looking at the formal reports written by those analysing the failures in the systems involved. Those reports are investigated for any factors inside the scope of this thesis, such as poor or ambiguous presentation of information, avoidable distractions, or over-confidence in the configuration of the system. These factors are then used to build a list of questions that one might ask in order to test the integrity of a system. These are then combined in Chapter Four with information from other

sources to construct a set of usability criteria which can be used to guide the design of a new system.

The goals and user analysis section in Chapter Four uses the philosophy of critical research again, with a strategy based on ethnography. The author is aware of the typical roles within the teams that are responsible for the programme that creates a space instrument because of past work in that field, and by being based at the Mullard Space Science Laboratory (MSSL) of UCL. This laboratory has one of the larger space research instrumentation programmes in the UK, with a longer history than most other groups in the UK, and so is an appropriate environment in which to be immersed. It also makes available potential interviewees with whose help the user analysis of Chapter Four gives a method of interface selection. Awareness of some of the example interfaces shown in Chapter Two is also directly due to this environment.

A philosophy split between interpretive and critical is used as the research method of Chapter Six on common user interfaces, which is derived mostly by extensive reading of conference papers which have been focused on astronomical instrumentation of all types. Certainly many new ideas have come to light in following cross references, without quite knowing where it was going to lead. The methods above only loosely describe the strategy taken in the final instance, which evolved into old-fashioned detective work.

The collection of interview data in Chapter Four uses a semiotics method, concentrating on the conversation analysis approach. The method is very much one of only gently attempting to influence the course of the interview, allowing the interviewee to bring out subjects that are important to them, and rigorously avoiding leading questions.

In analysing the information collected, narrative is an important technique in taking the variables extracted in the process shown in Chapter Four and then synthesizing personas from that information. It allows the presentation of non-numeric information in a form that promotes understanding between users and designers, and the subsequent generation of an instrument interface by a rational method.

Ideally the next step after gathering this data would be to attempt to prove its validity by using the deductions to construct a space instrument, launch it and use it for astronomy. Such an exercise might take at the very least five years from inception to launch even if a launch opportunity were present immediately, which is a scale of work out of the scope of this thesis. By using evidence from past missions, Chapter Five shows a method of building confidence in the interface criteria developed in Chapter Four. Much of the remainder of the work will rely upon a suitable future opportunity to show its validity.

Chapter Two: Exploring Interfaces

In Chapter One the theme of user interfaces to space science instruments was introduced as a central part of this work. This chapter aims to set the scene with a review of the more important aspects in the field of human-computer interfaces. Following that is a set of examples of interfaces from the field of astronomy over roughly the last decade, illustrating the significantly different approaches taken by individual groups. The chapter concludes with details of exploratory programming work targeted at developing an insight into some of the issues raised.

1 HCI Review

1.1 Why use a machine?

An inquiry into user interfaces needs to look a little at the historical perspective of the operation of machines. One can presume the purpose of a machine is to increase the rate of work of a person or group of people, or to increase the quality of their work. The question a finance provider will immediately ask is '...and by how much?' Thus we run straight into the idea of 'utility' - a measure of how useful or profitable a machine is. Other things being equal, a machine that was perceived to be easy to operate would be preferred to one that was not. The machine that is easy to use would be likely to have a greater throughput than the other. Things are not always equal, of course, and in our hypothetical situation trade-offs would have been needed between ease of use and cost, for example. If the 'better' machine cost a lot more, then despite its increased throughput the cost of the items produced might still be more than the other machine, and it would be unprofitable to adopt it.

How does this relate to space science? The fixed costs of a space mission are high, typically hundreds of millions of pounds. These are the costs of the instrument, the platform, the launcher, the ground support and other items. If we can show that the costs of any extra work required to make an instrument easier to use are small compared to these mission fixed costs, and if also we can reasonably expect a greater throughput as a result, then it seems a sensible path to follow. The work becomes cost effective and gives value for money. In the discussion on instrument lifecycle in Chapter Seven, we see that by forcing us to think of ease-of-use very early in an instrument programme, and the resultant movement of critical decisions to earlier in the programme than has been typical, it may actually cost less to take this route than to ignore it. This is frequently referred to as a 'left shift' or 'front loading' strategy in the context of systems design.

Ease of use is more commonly referred to as ‘usability’, and we’ll use the two terms synonymously.

What makes a machine easier to use - what increases its usability? In the introduction we brought in the concept of ergonomics, of which HCI is a large subset. What determines where to place a control, or what colour to make it, or how a user should be able to be aware of the results of the action they have just taken? A physical control knob could equally as well be a virtual control on a display screen - the arguments are the same. Essentially the action here is communication between user and machine, and since the concept of communication with an inanimate object is nonsense (assuming the validity of the Turing Test (Turing 1950)), the communication is actually from the designer or design team of the machine, as embodied in and restricted by the interface with which the user interacts. The interface is the point of contact and the detail of the implementation is likely to be hidden from the user.

1.2 Mental Models

Much has been written about the concept of the inner mental representation that a person builds on using any machine where part or all of the operation is hidden, and this representation is generally referred to as the *mental model* of a system. Philip Johnson-Laird is usually credited with first use of the idea (Johnson-Laird 1983) and it has become a popular theme in the literature. Virtually all proponents refer to its use with computer systems under the specific banner of HCI, but its valid use may be broader than that. Ever since people have used machines where part of the operation is not immediately obvious, there is an opportunity to imagine what is happening inside the box without knowing exactly what is going on. The mental model only needs to be as accurate as necessary to allow users to obtain good enough results for the job at hand. The ‘box’ might be a car engine, or a radio receiver, or a digital watch - users build an internal picture of where to locate virtual items inside it. For example, the watch may allow setting of time, alarm, date and stopwatch by cycling around the values with repetitive pressing of a button. Users may well build a mental representation of the watch operation by imagining a paper-like list of values, and this has the mental advantage of being able to visualise which value comes next in the sequence.

A user sees a system under their control somewhat differently than that envisaged by the designers of that system. The user will tend to build an abstract idea of what different areas exist, how they are interconnected and what they do. Understanding of these mental models is important in understanding how users always manage to discover operations that were not originally intended. Users also tend to bring experience of other systems with them,

introducing unpredictable factors into the situation. Donald Norman (Norman 1988) proposed that the three concepts of a design model, a user model, and a system image as in Fig. 1 were a good way to think of how to operate a machine.

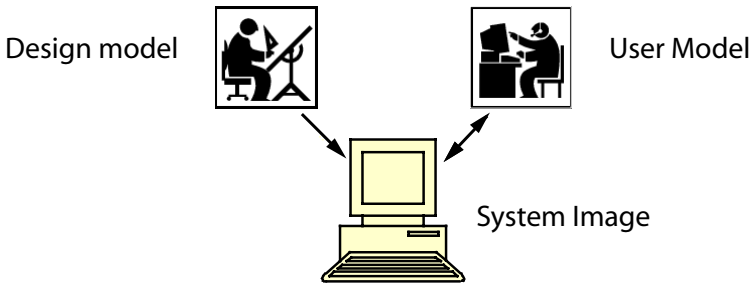


Fig. 1 Operation of a Machine

The design model is how the designer (or design team) regards what they are trying to create. Their role is to capture the functions that are required for the system and to implement those functions more or less as intended to create something that works. That implementation we call the 'system image'. In the world of commercial software, the designer has little to do with the operation of the package, at least until the next release. The only communication with a potential user is through the system image, and hence a great deal of effort is expended in commercial software in trying to ensure that a user can operate the software without being able to communicate with the designer. Clearly for commercial viability this has to be so and indeed the best commercial software barely needs any sort of manual. One of the central arguments of this discussion similarly is that space science instruments typically have little direct contact between designer and user - essentially it is all between the system image and the user, with some reference to the manual.

When the user tries to understand the system operation they will not see it in the same manner as the designer, as the type of work is different. The fact that the designer has to count an offset on a mechanism optical coder is of no interest to a user - they want to know which way the instrument is pointing. The system image conveys this information, and to a variable degree of usefulness depending on the implementation skill. The user model is that developed by the user through interaction with the system image. The better the system image communicates the designer's intention, the more accurate the user model will be, and the more successful will be the user interaction with the system.

Jakob Nielsen took the idea of mental model classification a step further with seven main types (Nielsen 1990), and suggested a short notation that might allow combined groups to be written in a compact manner:

U - user

D - designer

C - computer

R - researcher

T - task

W - world

M - manuals

The classification is largely self explanatory, being ways of looking at the situation from different angles.

Norman put down six points about the limitations of user mental models (Norman 1983a):

1. Mental models are incomplete representations of a system.
2. People's abilities to "run" their mental models are severely limited.
3. Mental models are unstable: People forget the details of the system...especially when (the system has) not been used for some period.
4. Mental models do not have firm boundaries: similar devices and operations get confused for one another.
5. Mental models are "unscientific": People maintain "superstitious" behaviour patterns even when they know they are unneeded because they cost little in physical effort and save mental effort.
6. Mental models are "parsimonious". People are willing to trade off extra physical actions for reduced mental complexity.

This was followed by DiSessa who described user mental models as incomplete, have limited operation, forgettable, confusable, have an ad hoc approach, and may minimise mental activity (DiSessa 1986).

The conclusions of these points are that in order for a system image to create a good user mental model, that image should be as simple as possible. Users relate to something familiar, such as the desktop metaphor common on personal computers now. The metaphor of folders, an open space and a waste bin is generally so acceptable that it is taken for granted, but actually implies good attention to detail to ensure that the illusion holds and the workings are kept invisible.

Other related work includes the following:

- Bruce Tognazzini summarised the effects of ambiguity in mental models (Tognazzini 1992): *"Ambiguity in an interface will always be discovered and used in a negative manner. The (user) model should reflect user tasks, rather than the design of underlying software or hardware".*

To paraphrase Tognazzini, use of an appropriate and everyday metaphor, such as a desktop, is important. Unusual ones, such as cartoon character icons to represent in this context critical nodes of a space flight system, may discourage inexperienced users. The metaphor needs to be realistic to create the desired mental model for the user, and implies some assessment of the user.

- When a system is first introduced to a user, learning by simply listening to an instructor is unlikely to build an effective model. Actually viewing the system is better, but the best approach is by actually operating it. (Tognazzini 1992).
- Tests by David Kieras showed that the use of detailed diagrams of the system may help by acting as a surrogate model to aid learning (Kieras 1984).
- In assessing how well someone has understood a task they are being asked to carry out, performance alone may not be a good measure of the validity of a user's model. People can obtain the correct result by the wrong method. One needs also to listen to verbal feedback at the same time (Payne 1990).
- Tognazzini suggested that the user model could be discussed in terms of four main topics, related to the actual human activity of using a machine (Tognazzini 1992):
    1. Visual and general sensory model - the information as presented to your eyes, ears and other senses.
    2. Kinesthetic model - the memory of what physical actions you had to take during previous operations of the system.
    3. Feedback and feel - how the system responded whilst you were interacting with the system.
    4. Resultant action - information resulting from your actions.

1.2.1 Mapping and Alignment:

A simple natural alignment of the control and the controlled object means that the operator has less mental work to do, and is using a simpler mental model - summed up by Norman as a ‘natural mapping’ (Norman 1988). A classic example often quoted is a cooker hob with four rings, where one can find all variations used for the placement of the controls. Fig. 2 represents four variants of a hob in plan view.

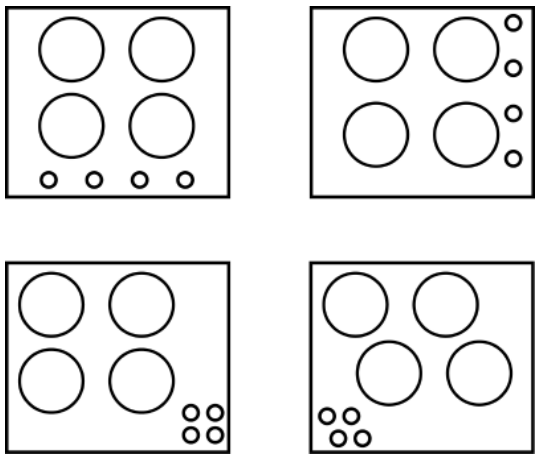


Fig. 2 Cooker Hob Layout

There is no method by which the top two layouts can be described as unambiguous. Four controls in a row cannot build a mental model of four systems in a square. Cooker manufacturers are forced to resort to labels of varying levels of clarity. The top right layout incidentally is hazardous for left handed operation as the user is encouraged to reach across rings that are hot or alight. It is likely that manufacturers cite the reduced space (and reduced fabrication cost) the top layouts take compared to the lower two, but when well designed the difference in space and cost should not be large. The perception of the manufacturer may be that design for minimum manufacturing cost is more important than design for optimum user interaction, which may be true until the time when the user is presented with a choice of hob design. The lower two designs show at a glance which control maps to which ring and any labelling would serve only to add clutter to the view.

Norman also brought this point out succinctly (Norman 1988):

> *“Whenever labels are necessary, consider another design”.*

People will frequently blame themselves when their mental model breaks down and errors happen, when the reality is that poor design is the real problem.

Mental models help you figure out what would happen in novel situations (Norman 1988), but open up an element of risk. Frequently a user model may be far from complete and whilst it allows managing of typical day to day situations it may encourage a false confidence of that user’s knowledge of the system. Incomplete user models ideally should correct themselves over time as a well designed HCI makes the error evident, but this may not

always happen. The example in Chapter Three of Three Mile Island shows the danger of false interpretation of the status of a stuck valve.

1.2.2 Heuristics

Several authors have formulated lists of heuristics, or rules of thumb, to aid in the design of general interactive systems. Three of the best known are Norman's "Seven layers of interaction" (Norman 2002); Ben Shneiderman's "Eight golden rules of interface design" (Shneiderman 1998); and Nielsen's "Ten heuristics" (Nielsen 1994). A recent addition is Tognazzini's "First principles of interface design" (Tognazzini 2003).

Two examples of these heuristics are:

- How easily can one tell what state the system is in? (Norman 2002)
- Look at the user's productivity, not the computer's. (Tognazzini 2003)

One piece of work by Bastien & Scapia is particularly valuable, as it is based on experimental evaluation of interface criteria (Bastien 1993). This work is used extensively in Chapter Four. The above lists of heuristics are very useful, but are not used further in this thesis as it is unclear as to whether they have an explicit experimental basis.

1.3 Artifacts and Widgets

This chapter began by talking about general purpose machines and implying simple manual controls. It's worth taking a look at some of the default toolbox of virtual controls and indicators that are available on most computer software development systems. This is the Window, Icon, Menu and Pointing device (WIMP) approach, intended to work with a display screen, pointer and keyboard. The Xerox Palo Alto Research Center is generally credited with the idea of using these four elements to form a graphical user interface (GUI), and with the first implementation in a device called the Alto in 1972. This was followed by the Xerox Star, and then commercialised by Apple Computer with the Lisa, and in particular by the Macintosh in 1984. Prior to these devices, the only option for a display was a single screen with a text-based interface driven from a keyboard.

The window feature allows multiple documents to be visible at one time, from multiple applications if required. Icons are a way to represent a document or a programme as an object in the mind of the user (a mental model) and allow movement, selection or execution with the pointing device. The menu allows discovery of what commands are available from a particular programme and eliminates the possibility of typing a non-existent command. The mental task becomes one of recognition of a menu item, rather than recall of a command line, and is easier and faster. The pointing device, commonly implemented as a mouse,

originally allowed selection (one click), dragging (click and hold), and execution (two clicks in rapid succession) when used with windows, menus and icons. The mouse currently implements a few more functions dependent on the platform and software. The original Xerox mouse was designed with three buttons; most of the industry followed this or adopted a two button mouse, with Apple adopting a one button design for the Macintosh. Many designs have now included a scroll wheel for fast vertical scrolling of screen documents, and some designs exist with more than three buttons as in Contour Design's ShuttlePro, shown in Fig. 3.

Other pointing devices exist and have their advantages and drawbacks. The trackerball is essentially an inverted mouse where the user's hand rests directly on a rotating ball, which drives the screen cursor. Graphics tablets allow movement of an electronic pen over an active surface to position the cursor, with many models translating pressure at the pen tip into instructions, for example, to vary the thickness of a line drawn on the screen. The other end of the pen may have a transducer as well and often be set up to act as an eraser. A touch screen takes this idea further, with a transparent active surface over the machine's display that responds to a special pen or even to a passive object such as a fingertip.

Fig. 3 ShuttlePro

Fig. 4 PowerMate

Fig. 4 shows another example of a specialised input device. This is the Griffin Powermate, a large rotary control about 50mm in diameter and particularly useful in audio-visual applications. Then there are a whole series of joysticks, pointing devices often used for playing games and based on two axis movement of a stick. The response of the system may vary as well. A mouse is normally configured to move a cursor relative to its current position, but may alternatively be set for example to give an absolute position reference as it is moved over a pre-defined area. A joystick might control the position of an object, or might be configured to control the acceleration of that object. Devices exist that are mounted on the head and pick up movements of the operator's eyes, and may have particular applications where the operator

is disabled. The whole field of pointing and general input devices is the subject of continuing research and development.

The on-screen items can typically be implemented by the software programmer as a straightforward function call, and give sophisticated appearance and behaviour for little effort compared to that needed to create such examples from scratch. Their availability does not preclude use of other custom controls and indicators; it does reduce the amount of work by the programmer considerably, and also has the benefit of achieving a level of standardisation in presenting concepts to users. Each operating system (Microsoft Windows, Apple MacOS, Sun Solaris, etc) has its own way of implementing the basic toolbox, and these implementations may change from one version to the next, but typically there is sufficient agreement between vendors on the basic requirements for interfaces to be rebuildable for any operating system with a graphical approach. Fig. 5 is an example of commercial software from the Omni Group, selected to give a representative sample of toolbox components, and annotated to show Apple Computer's names for the various parts.

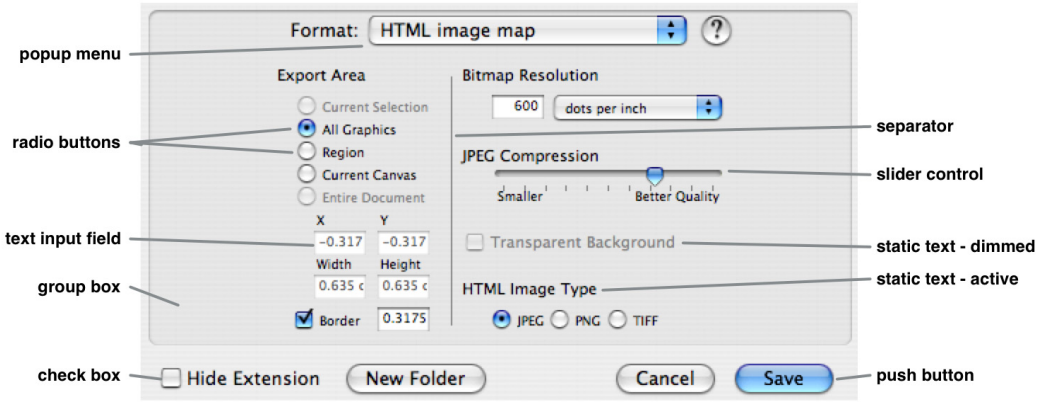


Fig. 5 Toolbox Components

This is a selection of the most common items available. The total selection does not need to be large to give a more than adequate means of constructing on-screen controls and indicators. For example, Apple's 'Interface Builder' (Apple Computer 2005b) application, offered as sufficient to build any complexity of human interface, shows about twenty different items. The precise number depends on how exactly the classification is made. By varying the dimensions, colour, labelling and position a wide differentiation can be achieved.

The current range of programming languages based on object-oriented techniques such as Java and C++ allows the programmer easily to separate the programme functional coding from that for the user interface, enabling the user interface to be built with just simple code

stubs for the application. The opportunity can be taken to develop a user interface early in a development cycle. Object orientated languages have the concept that the program is composed of individual objects each capable of actions, receiving messages and sending messages. Older procedural languages such as Fortran 77 and C have the concept of a program which is a simple list of instructions to the processor. Both have advantages and disadvantages.

1.4 User Feedback and Affordance

Any type of control is much more likely to be operated accurately if feedback is provided to the operator. For example, if one presses a doorbell push button and cannot hear even a muted response from inside the building, there immediately is a dilemma. Should it be pressed again because first time was not done firmly enough, but at the risk of annoying the occupants if it really is working? If one does nothing, there is a risk that the bell was faulty and the occupant never knows someone called. Pragmatically, most people would seek another method - and knock hard on the door.

The same consideration is true for virtual controls on a screen. If a user clicks on a control without feedback, there is no way of knowing whether the action has been registered. The programme may have crashed, or the control may not have been accurately targeted. So, as part of the programme calls that define screen objects in modern operating systems, there are features that appear to make a button depress with a three dimensional effect to give visual feedback to the operator. It's simple enough to do; the button is redrawn with a one or two pixel shift, typically to the right and downwards, but it gives a convincing effect, as illustrated by the dotted line adjacent to the 'OK' button in Fig. 6. Changes in shading may be used to enhance the effect. The inverse happens when the action is released. Feedback may be visual, as in this example, or audible with a momentary or sustained sound. It may be haptic (mechanical), as experienced with most reasonable quality keyboards. A steadily increasing pressure on a key results in initially little movement and then a sudden collapse of resistance as the key makes a well-defined movement and performs its function. The operator has immediate feedback that they have exerted sufficient effort to achieve the required task. Joystick controls may use haptic feedback for increased effect in game playing. Modern 'glass cockpit' aircraft use similar controls with simulated mechanical resistance replacing the legacy systems of direct mechanism feedback via the hydraulic fluid.

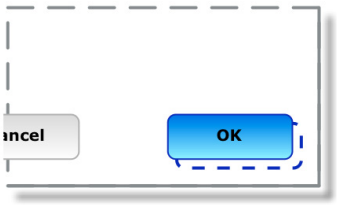


Fig. 6 3-D Effect

The push button gives a example of the concept of affordance, introduced by James Gibson (Gibson 1977) and popularised by Norman (Norman 1988). The image of the button suggests that it should be depressed by clicking as it corresponds with an idea with which most users are familiar. The *affordance* of the button corresponds with its intended action. If the intended action of the button was (illogically) to slide sideways, the affordance would clash with the intended action, and operation of the control would cause confusion and inefficiency. The desired implementation can be summarised by saying that an object or environment should, through clear unambiguous design, positively afford its intended function and negatively afford any unintended use. A negative affordance, for example, might be the common technique of reducing the contrast of an invalid screen button and making it appear grey. Modern operating system toolkits tend to make these features trivial to implement, although it is still easy for a programmer to forget to implement, as in the example, the correct affordance of 'greying out' an invalid control.

1.5 Existing Standards

As mentioned earlier, adopting standard ways of working within an operating system has the benefit to the user of presenting a way of working that is consistent and therefore minimises the mental workload needed. Nielsen suggested interface standards enhance productivity by raising throughput and reducing errors (Nielsen 1993). Alan Cooper accepts this but cautions that it is too easy to assume adherence to a given standard is a guarantee of a good interface (Cooper 2003). He makes the point that standards are only as stable as the time until their next re-issue, and rather should be treated as guidelines or rules of thumb to be followed unless there is a clear alternative.

The major operating system vendors have each published their own sets of guidelines on user interface design aimed at general purpose application development. Examples can be found from sources including Apple (Apple Computer 2005a), Microsoft (Microsoft 2004), Sun (for Java) (Sun Developer Network 2005) and The Gnome Project (Benson 2004). Each takes the concept of supplying a software developer with interface guidelines optimised for their own operating system, providing recommendations for layout, programming structures, menu operation and interaction with specific system software. Apple's document also has a useful first section which, whilst mentioning technology specific to the operating system, has general purpose guidelines applicable across multiple vendors.

There are several European standards (and hence British standards) in this field as well: ISO 13407 as one example "...provides guidance on human-centred design activities throughout the life cycle of computer-based interactive systems", and is aimed more at project managers than specialists (BSI 1999a). Another example, ISO 9241, provides (in

several parts) ergonomic requirements for visual display terminals, including quite fine detail on user interaction recommendations. Parts 11 and 13 are most relevant in the context here (BSI 1998), (BSI 1999b). In the US, IEEE standard 1295 has recommendations for the X-Windows display software package, and IEEE 9945-2 & 1003-1 cover the POSIX interface for the Unix system. Many of the standards include illustrated examples of good practice.

The elements above are examples of the contents of typical display screens; the section below summarises frequently advocated suggestions and constraints on how these screens might be laid out.

1.6 Presentation of Information

Information on a display area such as a computer screen should be laid out with careful consideration of parameters such as available space, colour, labelling, grids and white space. The various standards mentioned above contribute to this discussion to a varying extent. Another author, Edward Tufte (Tufte 1990), (Tufte 1997) and (Tufte 1997), describes numerous principles of the presentation of information, with many examples. His work is particularly appropriate when considering the understanding of data.

There are two empirically derived laws on how to place objects inside a display area:

1.6.1 Fitts' Law

When a pointer is moved across a screen and performs an action on a target, more time is taken for a distant target than a close one, and similarly for a small target than a large one. The distance argument is explained by both examples having roughly the same maximum speed of travel; the size argument is that the small target takes longer as more precise positioning is required to hit it.

If:

t = time (ms)

D = distance from start to target

S = target size along the line of motion

a, b, are constants derived from experimental human performance parameters

then Fitts' law states:

$$t = a + b \cdot \log_2((D/S) + 1)$$

The ratio of the distance and target size is the important parameter, making the statement applicable for any given display size. The statement generally only applies for human movements that are small enough to be accomplished in one continuous motion.

Raskin uses values of a = 50 and b = 150 for typical calculations (Raskin 2000). Fig. 7 shows the relationship for varying typical target sizes, using these values of a and b.

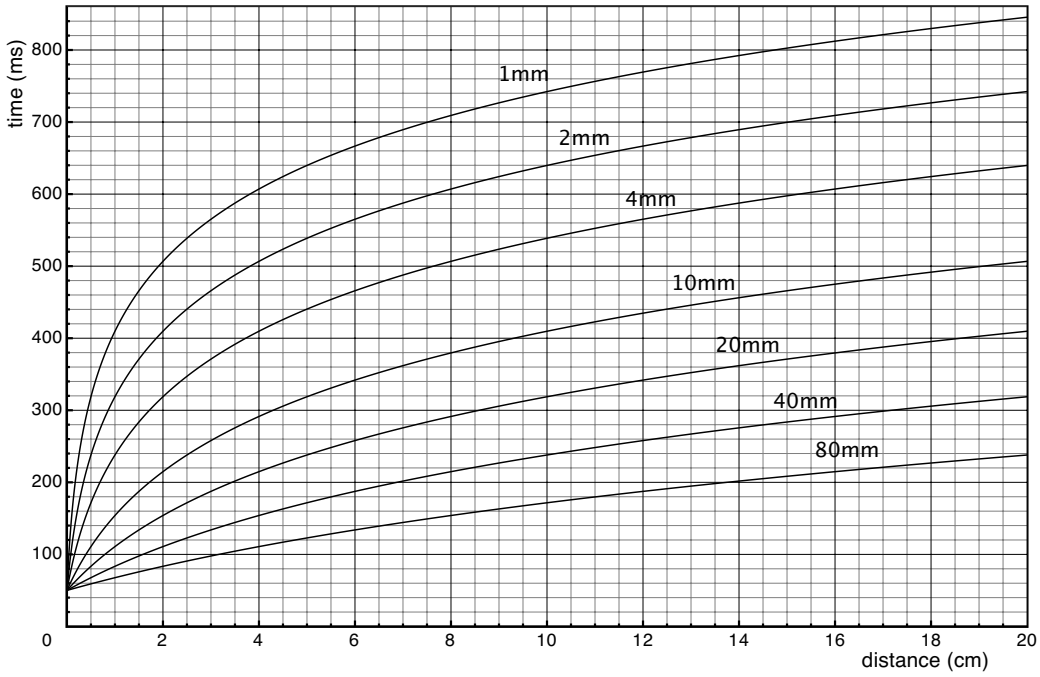


Fig. 7 Fitts' Law

Paul Fitts developed this law in 1954 as a model of human motor response and it is widely accepted as relevant to HCI design (Fitts 1954). The adoption of the mouse in preference to other pointing devices as part of the WIMP concept by Xerox PARC is credited to work by Card (Dixon 2002) following Fitts' ideas. The logical consequences of his law suggest that large objects are easier to select than small ones and explain (Raskin 2000) Apple's adoption and retention of a single menubar at the top of a display screen rather the menu per window approach adopted by other operating system providers. As the pointer is constrained to the edge of the screen, effectively the menubar is of infinite height and is therefore easy to select.

1.6.2 Hick's Law

If the user is presented with a choice of one among n objects as the target of an action, and if the probability of taking one alternative is equal to the others, then Hick's law (Hick 1952) states that the time t taken to decide is:

$$t = a + b \cdot \log_2(n + 1)$$

where a and b are constants that depend on the level of complexity of the interface and how familiar the user is with it.

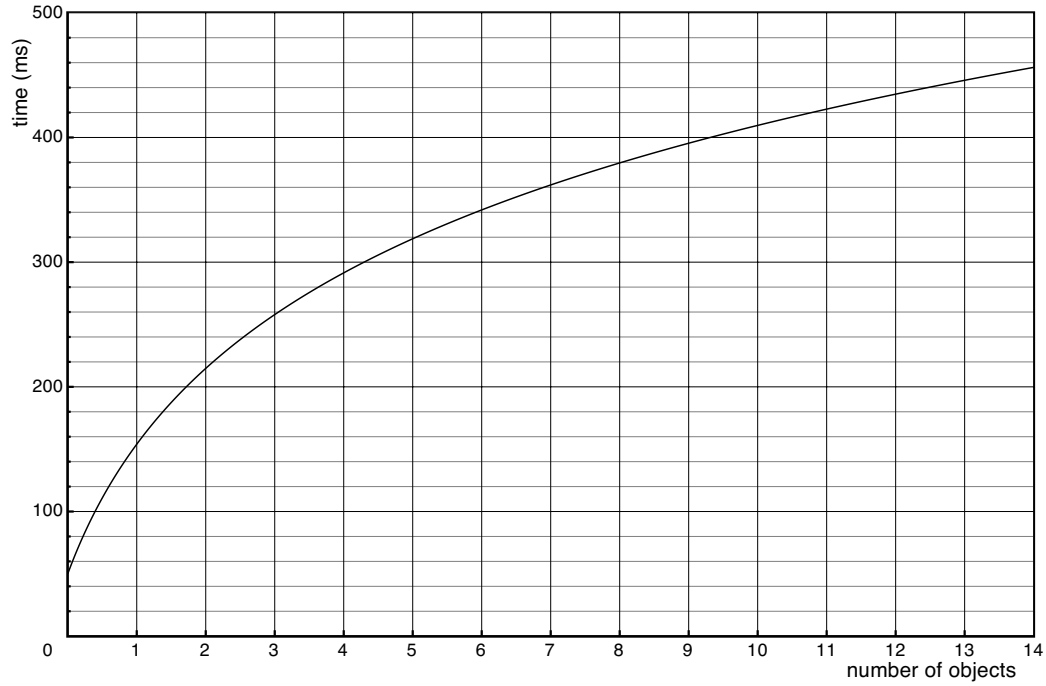


Fig. 8 Hick's Law

Clearly it is very similar in form to Fitts' law. Raskin suggests that in the absence of better information the same values of a = 50 and b = 150 are valid for estimating user performance. The relationship is probably most useful as a relative guide between different approaches to an interface and a way of quantifying the evident fact that complex decisions take longer than simple ones. Fig. 8 shows the relationship for a = 50 and b = 150.

Raskin draws the useful conclusion from this that giving the user many choices simultaneously is usually faster than organising the choices into two or more hierarchical groups (Raskin 2000). For example, choosing from one menu of eight items is faster than choosing from two menus of four items each, even without taking into account the time required to access the second menu.

1.6.3 GOMS Model

Card, Moran and Newell (Card 1983) proposed a numerical method of analysing a keystroke-based interface based on goals, objects, methods and selection rules (GOMS). It

gave a prediction of how long an experienced worker would take to perform a particular operation with a given interface design. The idea was taken up by other researchers in the field with more detailed approaches, such as Kieras's concept of the natural GOMS language (NGOMSL) (Kieras 1994), and the automated critical path method (CPM-GOMS) described by Bonnie John (John 2002), which allow for learning times by an inexperienced operator. For these more advanced methods, impressive absolute accuracies have been achieved of better than 5% in estimating and then measuring operator times to do a particular task. However, often what is required is a method of making comparative measurements between two different approaches and for this the simpler GOMS method would appear to be adequate.

The following table (Table 1) shows the mnemonics proposed by Card *et al* for the simple GOMS analysis (Card 1983) (p264). They are based on the idea of sequential processes comprising three basic user actions of keying (K), pointing (P) and homing (H), which are known as gestures, plus thinking time (M) plus machine response time (R). A graphical input device (GID) is, for example, a mouse.

Mnemonic	Name	Nominal Time (s)	Time Range (s)	Description
K	Keying	0.2	0.08 - 1.2	Time to operate keyboard (55wpm typist)
P	Pointing	1.1	0.8 - 1.5	Time to point to a position on the display. Excludes clicking afterwards
H	Homing	0.4		Time to move operator's hand between keyboard and GID or vice versa
M	Mentally preparing	1.35		Time to mentally prepare for the next step
R	Responding	Hardware dependent		Time to wait for computer to respond to any input

Table 1 Simple GOMS Analysis

Also part of the analysis is a set of rules (not reproduced here) for sizing the mental operator M. They allow for the fact that, for example, entering a four digit number with four keystrokes really only requires one unit of mental preparation time in total rather than one per keystroke.

This degree of analysis may be more than is needed for some applications. However, if one is producing a screen layout where guidance is felt necessary in order to produce an interface that is quick to use, then one of these GOMS models may be appropriate to follow.

1.6.4 Use of Colour

Good use of colour can greatly enhance the interpretation of a display and vice-versa. A colour value can be defined in terms of hue, saturation and brightness. It can be thought of in a physics context as a visible spectral line or lines of a particular equivalent wavelength, width and intensity that gives or give a certain visual response.

Tufte's analysis of this topic is useful, including many examples printed with high precision (Tufte 1990) pp 80 - 95. To re-state a selection of basic points: "… fundamental uses of colour in information design (are): to label (colour as noun), to measure (colour as quantity), to represent or imitate reality (colour as representation) and to enliven or decorate (colour as beauty)". Tufte paraphrases Imhof's (Imhof 1982) first rule for cartography on the use of colour saturation as an important constructive idea: "(saturated) colour spots against a light grey or muted field highlight and italicise data, and also help to weave an overall harmony". Many people are able to discriminate between 20,000 colours, but "For encoding abstract information, however, more than 20 or 30 colours frequently produce not diminishing but negative terms". For distinguishing adjacent data values, he states that a scale of colours using progressive brightness values of the same hue may be more effective than a scale of different hues, particularly if thin lines of the same hue are used as separators or contours. Labelling these contours may help improve usability, sparingly if in print or in a dedicated area on a display screen.

As an example of a colour scheme to adopt, Tufte advocates using a palette of colours found in nature, such as the lighter blues, yellows and greys, for general colour schemes. This gives a coherent theme to the information, and still allows the use of saturated spot colours for highlighting specific data. One issue to be aware of is perceptual colour shifts caused to a small area of colour by an adjacent large area of contrasting colour, as this can cause mis-interpretation of data. Colour blindness is another issue to be aware of, as about 8% of the population (particularly males) have some problem resolving colour differences (IEE 2004). Distinguishing between red and green is the most common problem, but can range all the way to the rare situation of no colour perception at all. People with mild symptoms are frequently unaware of a problem. The design of displays clearly needs to take colour deficiencies into account, as there will be frequent problems otherwise. Normally for this reason colour would only be used as a secondary distinction between multiple states. The primary distinction might for example be position, labelling, or shape.

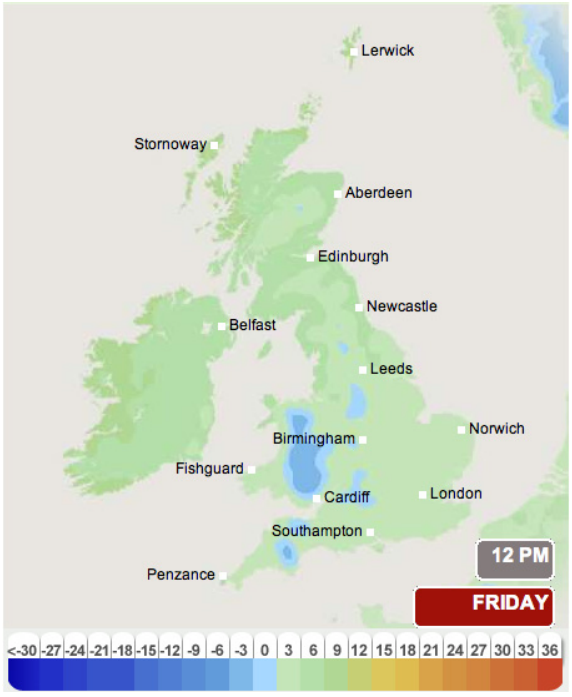


Fig. 9 BBC Colour Scale

Fig. 10 Monochrome Scale

The BBC web weather map in Fig. 9 from their website (BBC 2006) illustrates a progressive brightness colour scale. The scale is centred around zero, as the monochrome scale in Fig. 10 clarifies if this page is being viewed in colour. The values below zero and including zero are shown as a range of blues of progressively varying brightness. The positive values smoothly change brightness as well but mix the concept with common expectation and allow a hue shift from green through to red as well.

The scale incidentally appears confusing, as the colour blocks themselves are labelled with the temperature rather than labelling the boundaries between them. As temperature is a continuously varying property, a label of a specific temperature should apply to the contour between different colours, and not the colour area itself.

1.6.5 General Design Points

It is generally accepted that part of the functioning of the human brain involves the use of short term memory, where information may be retained from only a few seconds up to about thirty seconds (Atkinson 1968). With many computer interfaces, the user is required to read and possibly interpret information from one display page and enter it somewhere else. If that process takes any more than a few seconds, there is a significant risk that the information will have been forgotten or mis-remembered. If the machine already has the information, then it is weak programming if that information is not made available at all places where it is needed.

This aspect is relevant for machine control and the idea of a reactive interface. If a user operates a control and the instrument takes a long time - say more than ten seconds - to respond, the user may have forgotten a significant part of the reason for changing that

control. However, if the response is immediate or within say about two seconds, all the reasoning is still there in the operator's mind and if further operation is needed it can take this into account. It would be reasonable to expect these times to vary with different subjects and environment, and for the loss of information to be progressive over time. This thinking is used in Chapter Seven to suggest a *responsive* simulator, where accuracy of simulation is traded against fast reaction time, with the target of achieving worst-case responses of less than a few seconds.

Optimisation of a specific interface design is likely to involve trade offs of one parameter against another. For example, a design may initially appear to be cluttered with too much information presented to the user. One possibly useful analogy here is to compare clutter with noise in a communication channel - where the channel in this case is the path between interface and user. Too much clutter results in a poor signal to noise ratio, and the wanted information is lost amongst the unwanted. However, on attempting to rationalise the situation in this noisy design, it might become apparent that removing any of the display objects will result in a loss of usability. The decision could be to leave it as it is, or to reduce the size of some of the objects, or to increase the required screen size. Each case will rely upon a balanced judgement of the whole system problem at the time.

There are many other aspects of psychology, some of which are embedded in the user interface criteria developed in Chapter Four. General concepts include the ideas of forgiveness and not letting the user feel stupid, mentioned by most writers on the subject, and summarised by the idea of ensuring the user feels in control of the system. Forgiveness implies that if the user changes something and then realizes the error, the system should offer a trivial means of restoring itself to the previous state. This is usually implemented as an 'undo' command, and in the best software can track back as far as practical over the operating session. Similarly, ensuring the operator is protected from feeling stupid implies in particular removing ambiguity from the interface, with the aim that the operator is able to build confidence in its use.

2 Examples of Interfaces for Astronomical Instruments

The interfaces for astronomical instruments have a wide range of styles of implementation. The following are some examples over the period 1995 to 2006 as a means of illustration of the different styles taken by various design teams. Those teams were on some occasions directly part of the instrument science team and on other occasions the recipients of an industrial contract. The examples are roughly in chronological order and show both ground-based and space instruments. Some examples show the engineering interface to the

instrument, whilst others show the science planning software interface. In many observing programmes, operators actually control the instrument whilst observing scientists prepare plans beforehand which are then submitted for possible inclusion in the programme. Different interfaces are typically used for the two functions.

The sample of images here show only a fraction of the systems and their many screens of data that are in use. They are presented here simply to give an indication of the issues involved. They are also only snapshots that give little idea of the important actual dynamics of the tools. Ease of use is not just ability to understand a page of information quickly; if for example the system takes ten minutes to respond to a button press, or regularly crashes, or keeps forcing the user to fill in the same data over and over again in different places, then user frustration will occur and errors will happen.

The images here also demonstrate to varying degrees the level of specialist knowledge put into their design, although to assign estimates here would only be guesswork. For example, the thoroughness with which user interface criteria such as are discussed in Chapter Four have been implemented will make a difference in usability. The degree to which the various groups of users have been catered for in the design of these interfaces requires more knowledge of the instrument in order to judge, but is just as important. The interface may have been available at the start of instrument development, or may have only been working properly near the end, and may or may not have been subject to regular user testing during its development. The interface may have had a dedicated developer, or it may have been built by a programmer who at the same time was very involved in, for example, testing a difficult network connection. All these factors are discussed at length later in this thesis.

Table 2 summarises the examples considered.

Instrument	Description	Interface		
		Engin-eering	Science Data	Science Plan
SOHO CDS	Part of space-based observatory, 12 instruments	Fig. 11	Fig. 12	Fig. 13
Liverpool Telescope	Ground-based telescope, multiple builds	Fig. 14	-	Fig. 15
Hubble Space Telescope	Earth orbit telescope, 5 instruments	-	-	Fig. 16, Fig. 17
Gemini	Ground telescope, 2 installations	-	-	Fig. 18, Fig. 19
SOAR	Ground telescope	Fig. 20	-	-
XMM-Newton	Earth orbit telescope, 3 instruments	-	-	Fig. 21, Fig. 22
Solar-B EIS	Part of Earth orbit mission, 3 instruments	Fig. 25, Fig. 26	-	Fig. 23, Fig. 24

Table 2 Interface Examples

Grateful acknowledgements are given to the originators of the images used here, as referenced in the text.

2.1 SOHO Coronal Diagnostic Spectrometer (CDS)

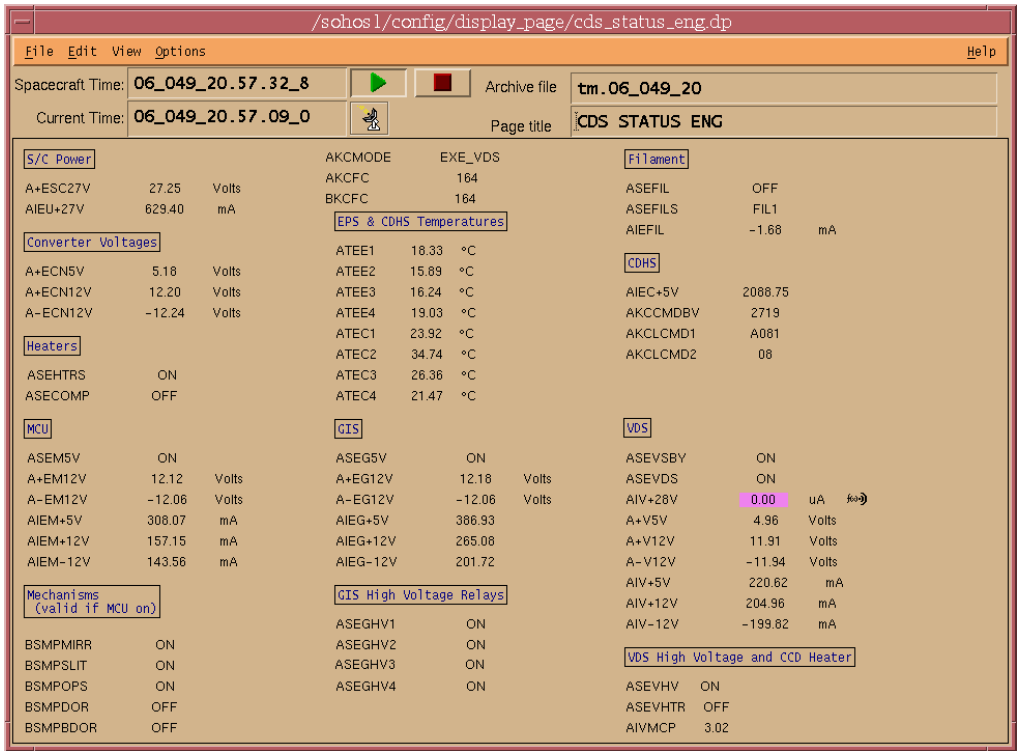


Fig. 11 CDS Engineering

This was part of the Solar Heliospheric Observatory (SOHO) mission launched in 1995. The CDS engineering command and telemetry system ran under Unix and was fabricated as an industrial contract agreed in 1993. The telemetry interface consisted of a set of pages as in Fig. 11 detailing each of the major sub-systems. Users were able to create their own pages across the whole instrument database, although this feature was rarely used. The system was employed for both ground testing and flight operation. Parameters that were outside preset limits were highlighted with colour to indicate a warning or a dangerous condition, as can be seen with one parameter bottom right in the illustration. The displays had some text features and virtually no graphics, and so parameter names were the 8 character maximum used in the instrument database. This required learning, or lookup in a paper file. Raw parameter values were translated to a calibrated parameter such as temperature or voltage if appropriate. Control of recording and replay of log files was also available from the display page, with recorded parameters played back to the display in the same manner as live parameters.

Commanding was carried out using a simple text command line and required familiarity with over one hundred mnemonics and their varied parameters. Potentially hazardous commands displayed a dialogue box for additional confirmation of that command.

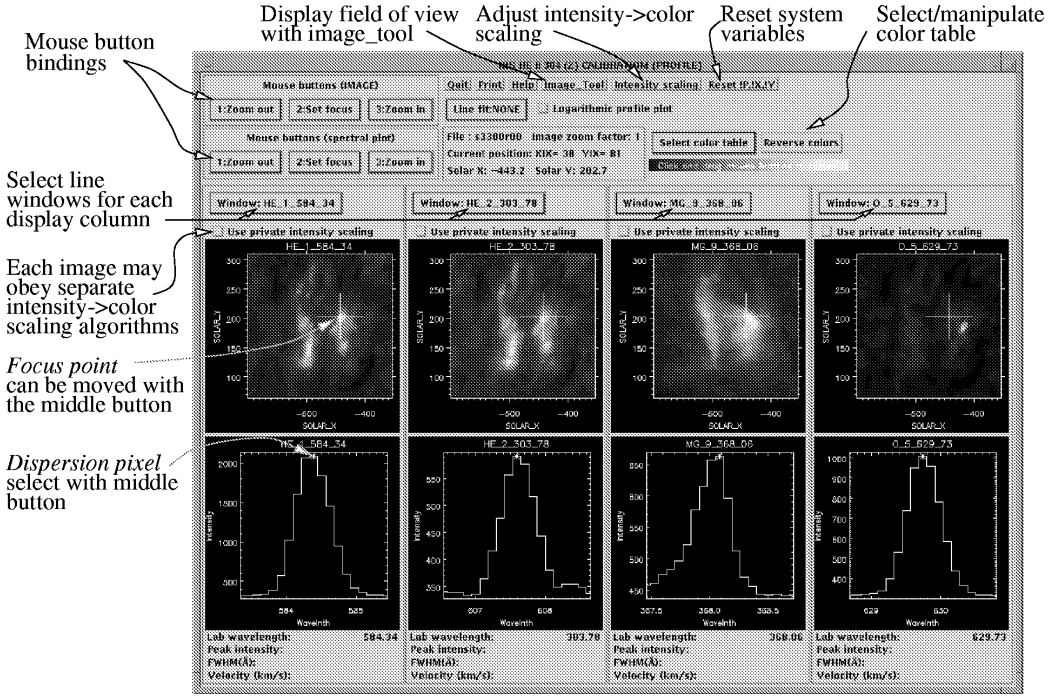


Fig. 12 CDS Quicklook Page

CDS operators could make use of “Quicklook” software routines to check the basic integrity of the downloaded science data in near real time. This was a software package written in IDL (RSI Inc. 2006) by the science team and produced diagrams such as Fig. 12. This is an annotated version from the software manual (Brekke 1997).

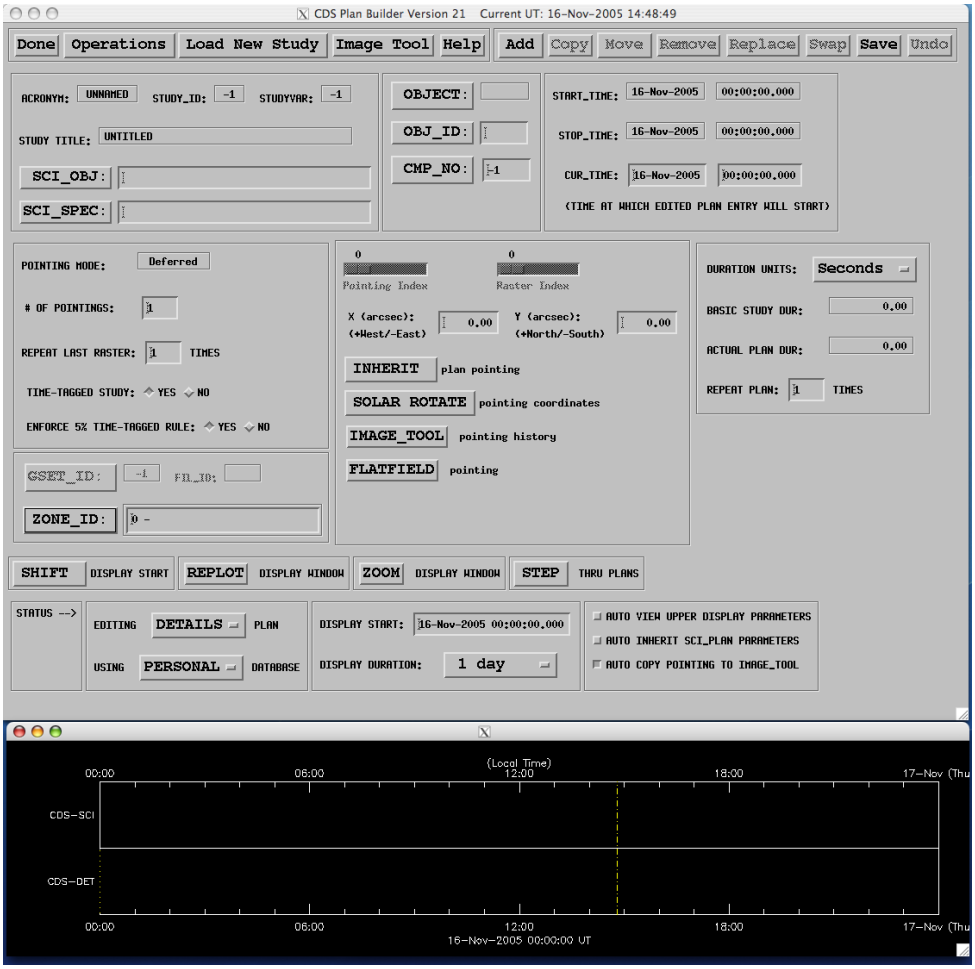


Fig. 13 CDS Science Planning Tool

Observation sequences were defined with the use of the CDS Planning Tool, dating from about 1995. This comprised three separate but interlinked tools written using IDL by the science team and currently (early 2006) forms part of the regularly-updated “SolarSoft” suite covering solar instruments (SolarSoft 2006). The output from the planning tool was used to build part of the control file that an operator would send to the instrument, and a typical screen view is shown in Fig. 13. This shows the parameters to be entered and, at the bottom, the timeline generated from the various observation sequences.

2.2 Liverpool Telescope

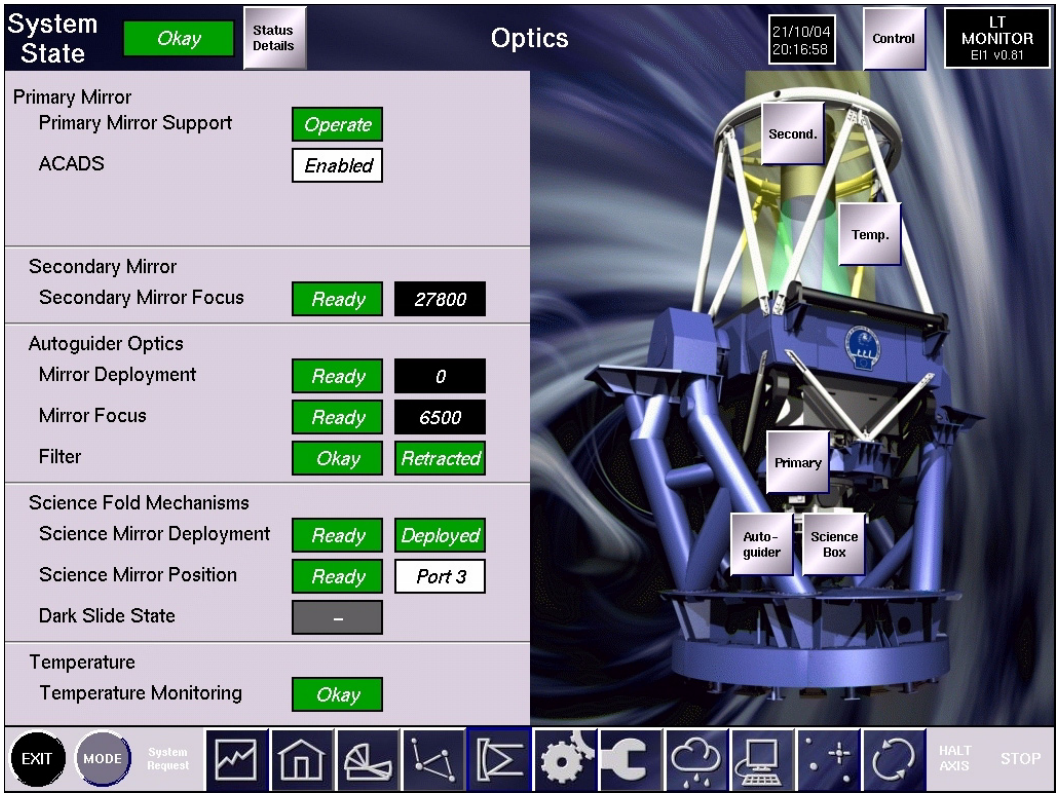


Fig. 14 Liverpool Telescope ECI

The Liverpool Telescope is a fully robotic telescope at La Palma, Canary Islands. It was designed to a standard concept as one of a series by Telescope Technologies Ltd., who supplied the images here, and is therefore unusual in a subject area where most technology is created as a one-off. The design of the remote control system dates from about 2000 and is still current in early 2006.

There are separate systems for engineering and science. One example screen from the Engineering Control Interface (ECI) is shown in Fig. 14. Each screen has a diagram illustrating the particular area or sub-system that is referred to, alongside the telemetry reported from that area. A row of controls along the bottom of the screen allow different sub-systems to be selected, and a control button at top right allows commands to be entered. The screens are generally clear and self explanatory, and have a consistent philosophy behind their design.

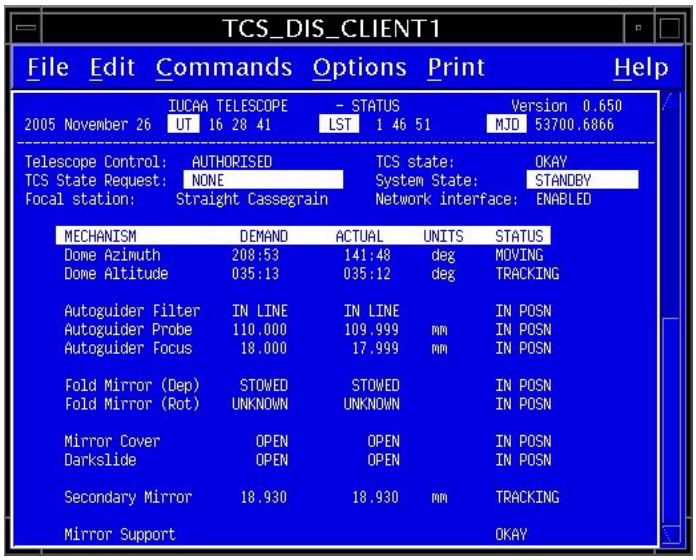


Fig. 15 Liverpool Telescope TCS

The Telescope Control System (TCS) is the interface intended for use by science users (Fig. 15). In contrast to the ECI, it is a completely text based display with pull-down menus giving access to commands. The design is still consistent between the various sub-system displays, but shows a very different approach to the ECI.

2.3 Hubble Space Telescope

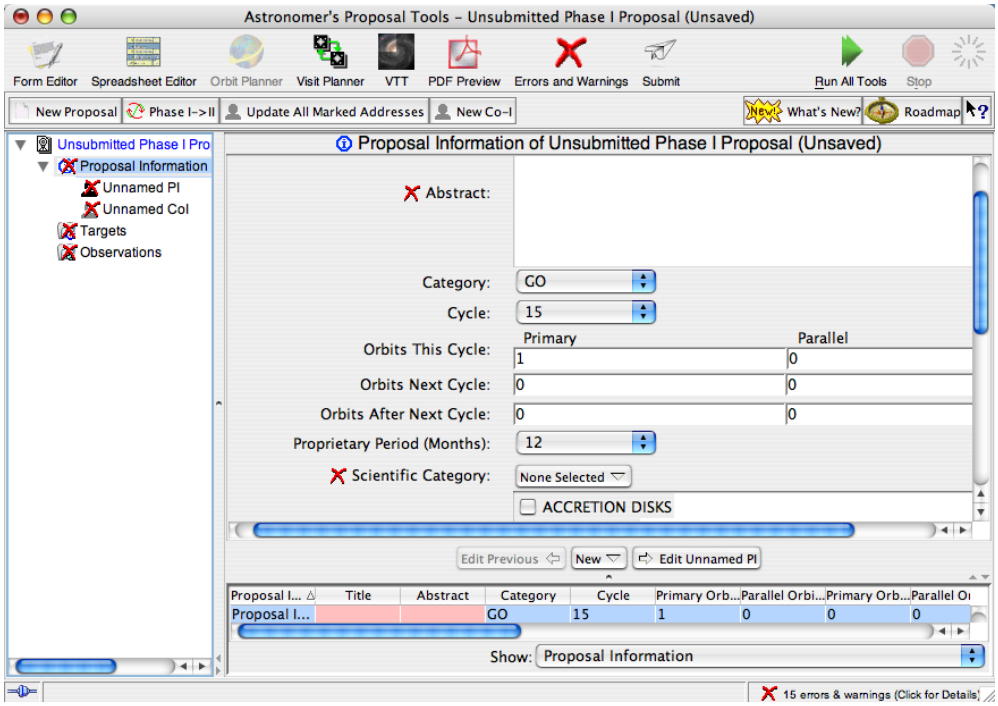


Fig. 16 Hubble APT

The Hubble Space Telescope (HST) has a comprehensive set of science tools integrated together in one utility called the Astronomer's Proposal Tool (APT) (Space Telescope Science Institute 2006). The current tool dates from about 2003 but with a legacy going back at least fifteen years. The current tool is written in Java by the NASA science support teams and includes forms to fill in the observation proposal, an orbit planner and a visual target

tuner (VTT) to visualise the expected results using existing databases. Comprehensive help is also included. Fig. 16 shows the initial screen for filling in a proposal. The bar at the top contains icons for accessing the various component tools, whilst the left-hand tree structure allows access to the different parts of the proposal.

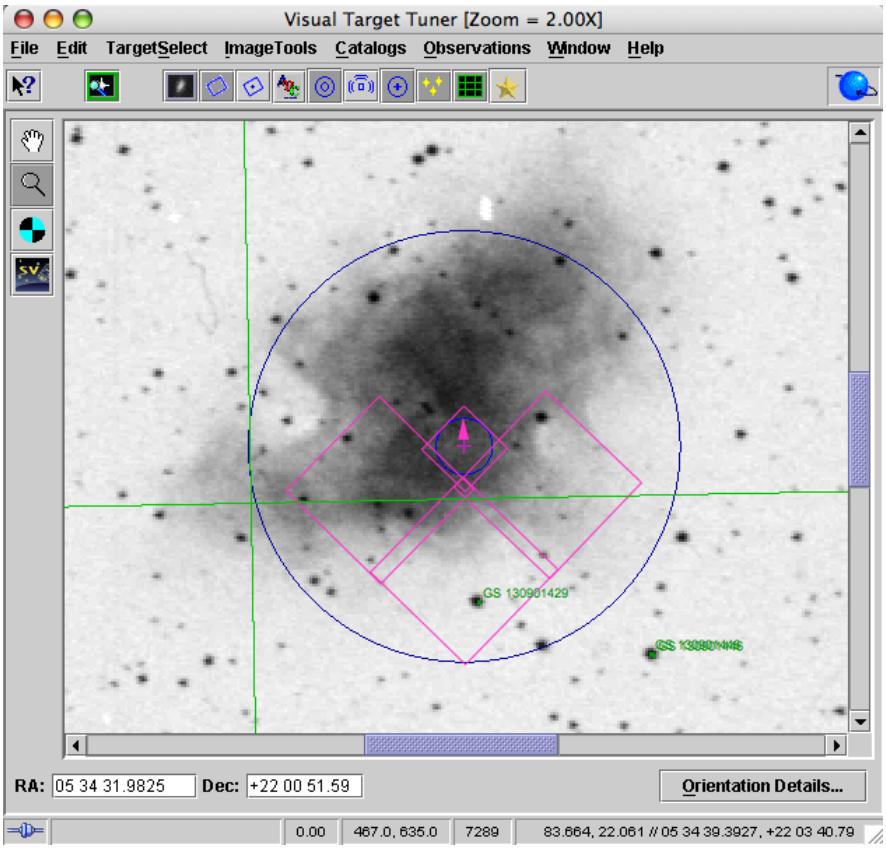


Fig. 17 Hubble VTT

One of the component tools, the VTT, is worth looking at more closely to examine its capability. For example, it has interfaces to on-line astronomical catalogues with which to plan an observation. An image from such a catalogue is shown in Fig. 17. This also shows how the instrument image centre and overall field of view can be plotted over the retrieved image, catalogue names added to selected objects and a reference grid displayed. There is a drop-down menu system for all the commands of the tool, with the more common ones presented as clickable icons under the menu bar. The image can be zoomed in and out, it can be re-oriented, and the cursor position is continuously read out at the bottom of the display. The results of the visual set-up can then be imported back into the proposal.

Chapter Six has more details on the work behind the APT. The source code forming the software modules is freely available for other uses.

2.4 Gemini

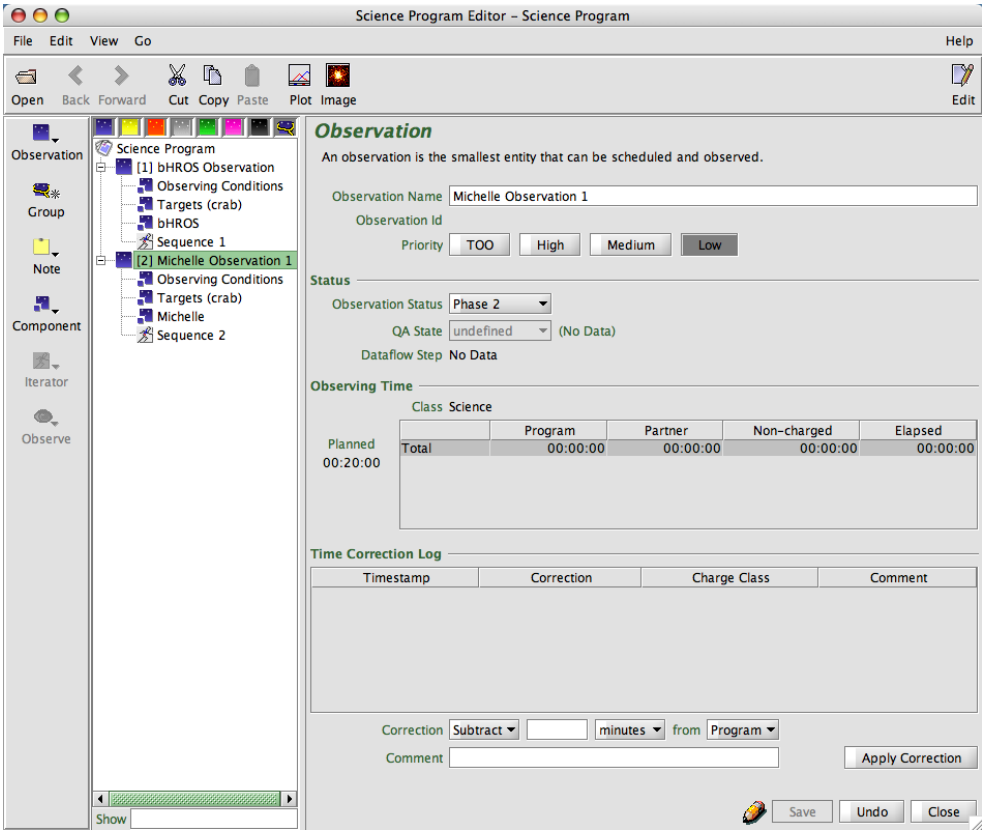


Fig. 18 Gemini Planning Tool

The Gemini observing tool (OT) shared a common heritage with the Hubble APT in the use of some common software modules, but the two paths then diverged. Chapter Six has more details. Fig. 18 shows the phase 2 Science Program Editor, for use when the phase 1 proposal has been accepted. It has a similar tree structure to the APT with the addition of a vertical selection bar on the left.

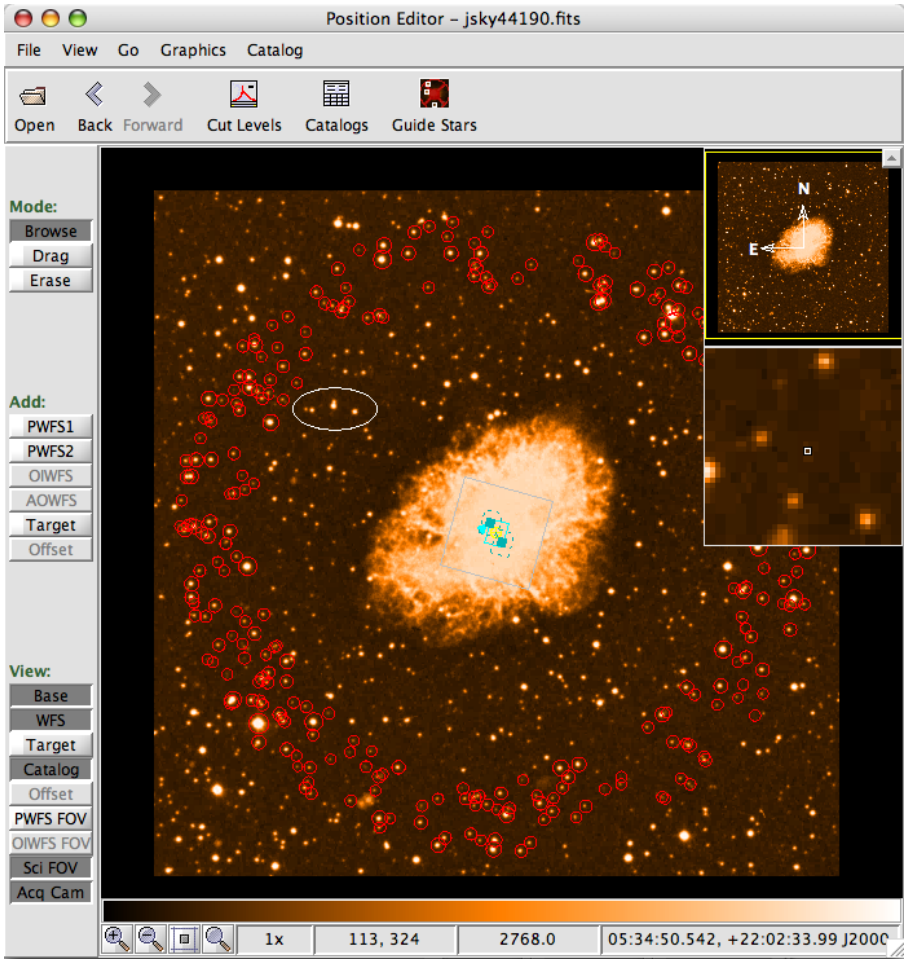


Fig. 19 Gemini Position Editor

The Position Editor (Fig. 19) is an embedded tool within the Science Program Editor and like the VTT is used for previewing images. It uses drop down menus to select from multiple functions and allows an image to be pulled from one of many on-line catalogues. Markers can be superimposed showing such points as the centre of view, items of interest (the small ellipse drawn towards top left), and telescope guide stars (highlighted in a large ring towards the edge of the image). Two optional windows top right give an overall view with compass points, and a zoomed view at the cursor position. The whole image can be zoomed in or out as well. A ‘cut levels’ embedded tool, not shown, allows the intensity range and contrast of the image to be controlled in order to emphasise subtle features. From the images, the Position Editor and the VTT clearly share a common heritage.

2.5 SOAR

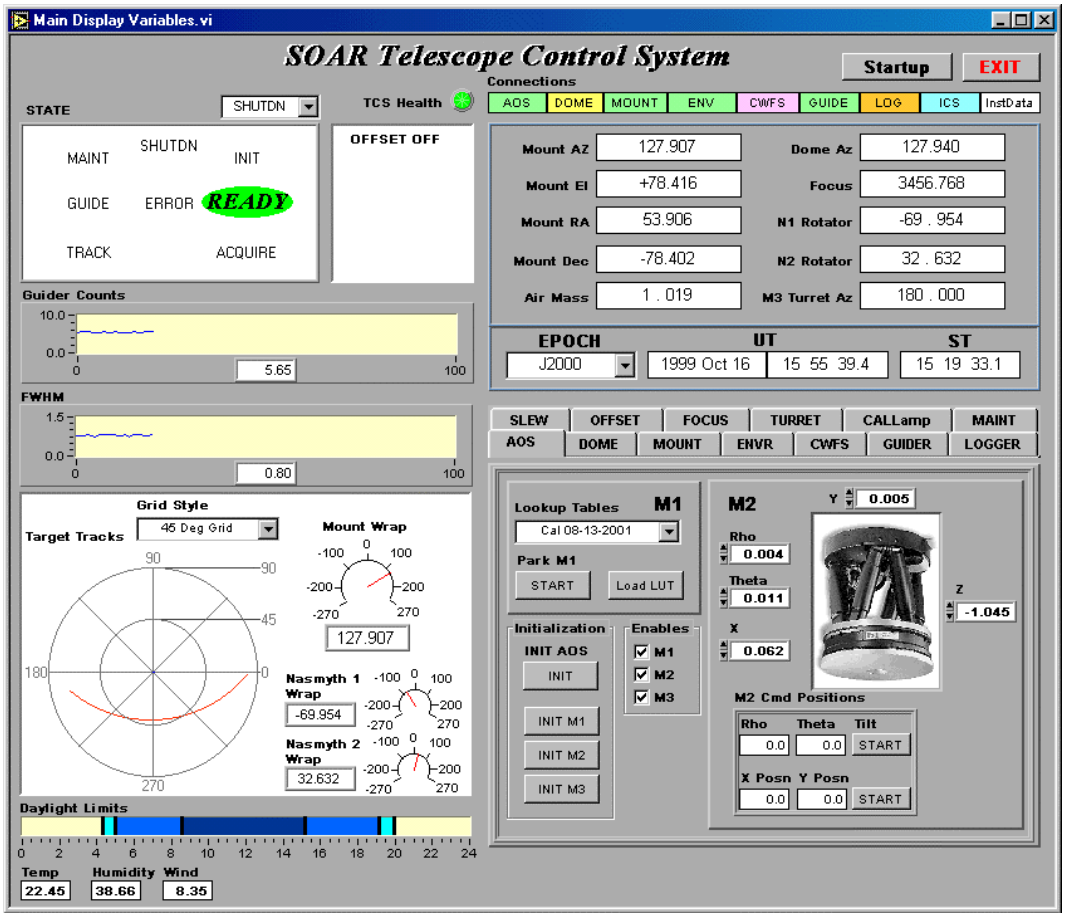


Fig. 20 SOAR Operator Panel

The control system for the SOuthern Astrophysical Research telescope (SOAR) dates from about 1999 and is constructed using the LabVIEW software package from National Instruments. The main operator's panel is shown in Fig. 20 from Ashe *et al* (Ashe 2000) and combines data display and control. About 75% of the area is devoted to displays that the operator should always have visible, whilst the lower right area has multiple tabs for the operator to select the action to perform. The panel uses a combination of custom and built-in LabVIEW controls. The image holds a lot of detail whilst still being clear to understand at a glance, with good use of mid colour background, contrast, spot colour highlights, group boxes and images of the system.

2.6 XMM-Newton

The X-ray Multi-Mirror mission (XMM-Newton) planning software allows observations to be proposed for the XMM instruments individually. The software was written under an industrial contract for ESA. Unlike the other examples here, it is designed to run in a web browser window with the information processing carried out at the mission server. Fig. 21 shows an extract from the manual (Bretfellner 2004) for a typical web page for defining instrument parameters for an observation.

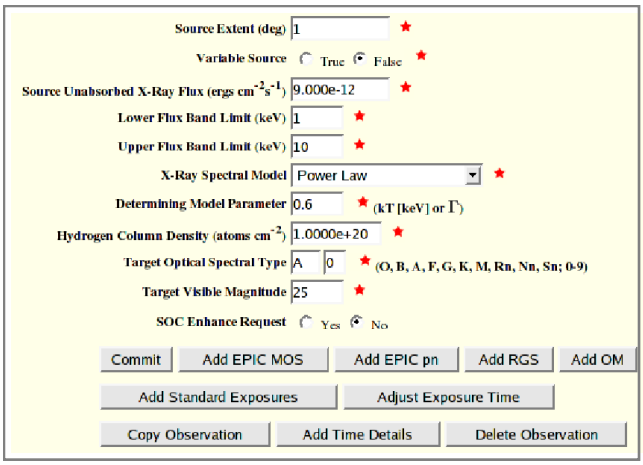


Fig. 21 XMM Planning Tool

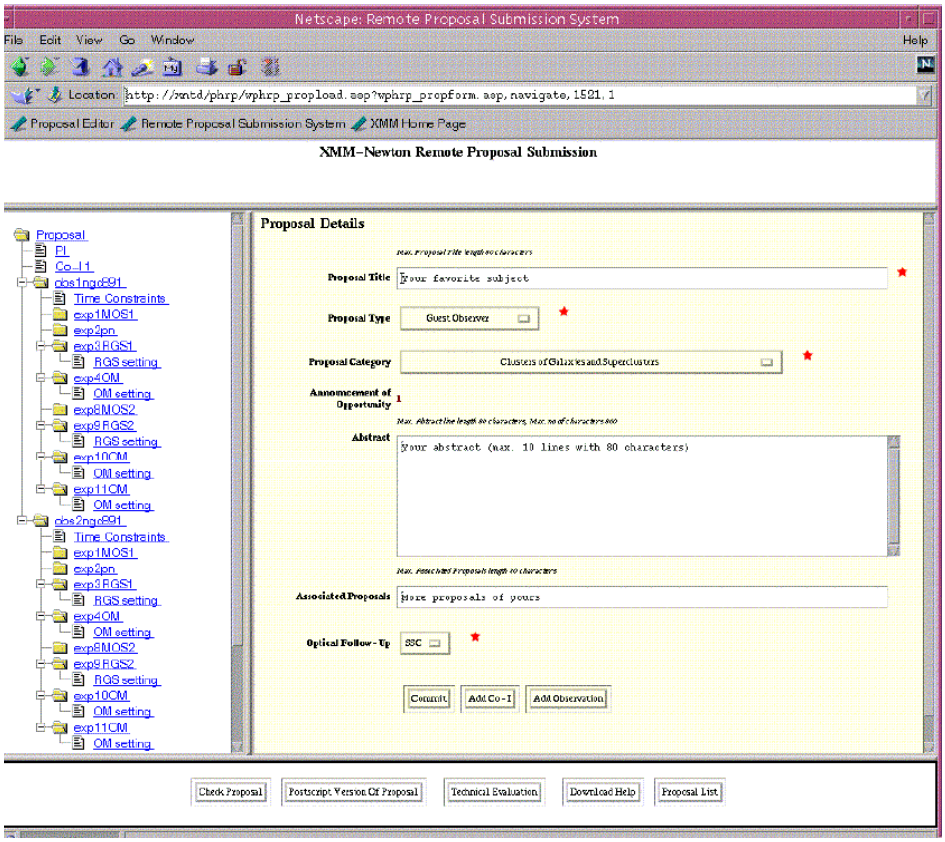


Fig. 22 XMM Tool Overall View

A tree diagram with folders and files provides navigation around the planning tool. An example from the manual is shown in Fig. 22, illustrating the browser presentation, the

navigation tree and the page for specifying the proposal details. At the bottom of the page are controls that are always available allowing the proposal to be checked or technically evaluated, a printed version to be generated, help to be displayed, and other proposals from the appropriate principal investigator to be displayed.

2.7 Solar-B EIS

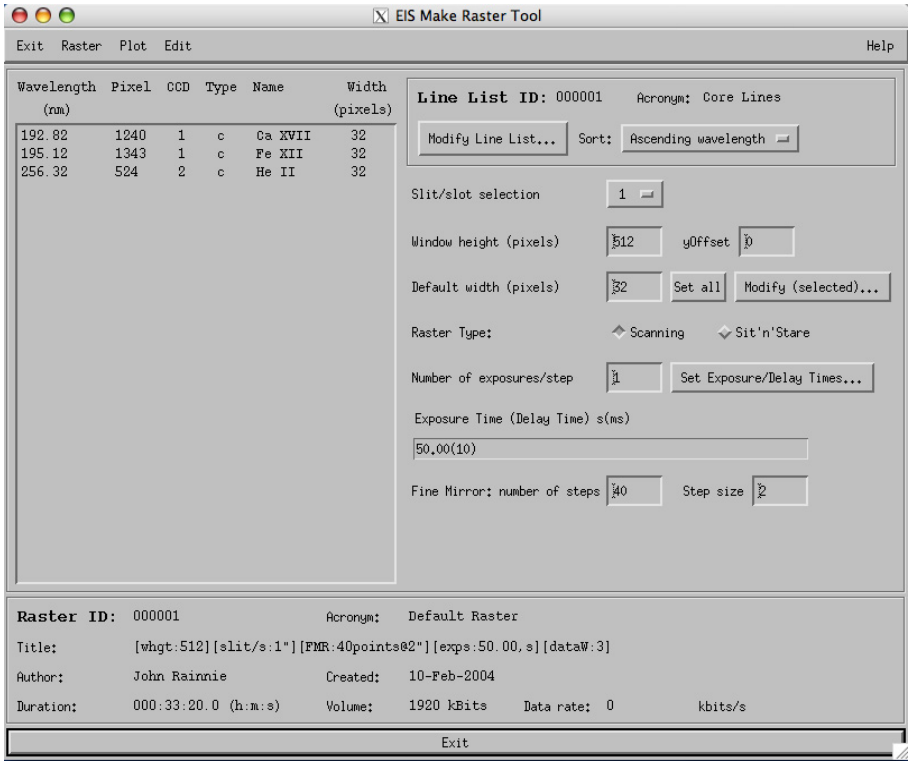


Fig. 23 EIS Make Raster Tool

The science planning tool for the Extreme-ultraviolet Imaging Spectrometer (EIS) on the Solar-B mission is written in IDL by members of the science support team. The illustrations here are reasonably mature work-in-progress in early 2006. As the subject matter and the teams involved have much commonality with the earlier CDS instrument illustrated in 2.1, the science tools broadly follow the ideas conceived for the CDS. There are three interlocking tools; one for generating rasters, one for studies and one for plans for observation proposals. Fig. 23 shows a partially completed raster proposal, and Fig. 24 shows the timeline created by the plan tool after the rasters and studies have been defined. A study by definition contains one or more rasters.

The tools prompt the user for choices of spectral lines to be observed, and from the instrument settings supplied estimate the duration of the study and the volume of data required. These details are merged with the other instruments' requirements and the spacecraft requirements to ensure there are no unwanted interactions such as spacecraft pointing disturbances or data recorder overflow.

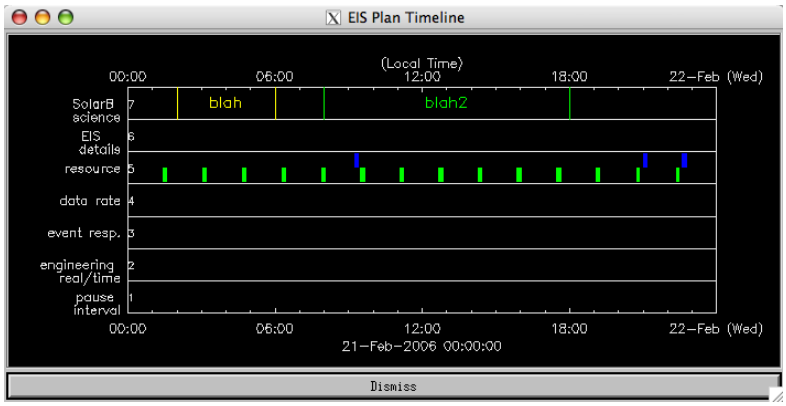


Fig. 24 EIS Timeline Display

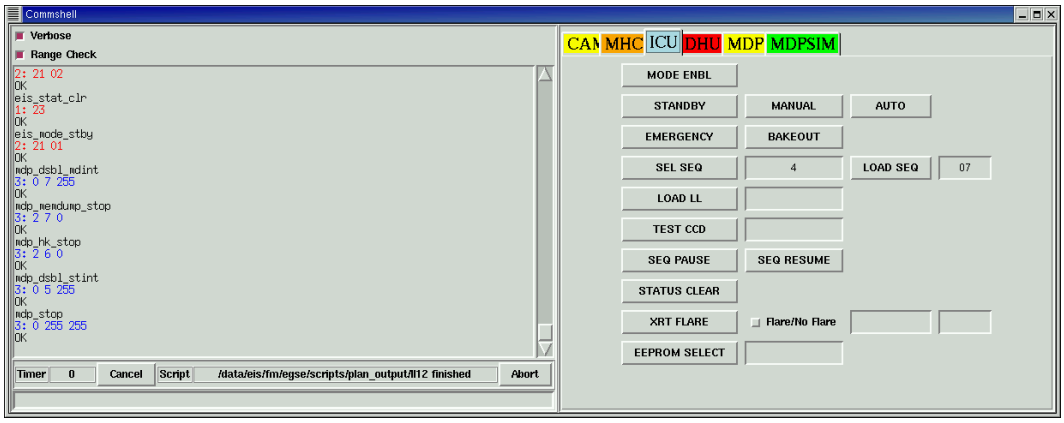


Fig. 25 EIS Command Interface

The command interface (Fig. 25) for EIS shows on the left hand side a scrolling window displaying the command mnemonics and the instrument response as each is transmitted. On the right hand side is a set of tabs for each instrument sub-system and a set of controls and parameters fields for each.

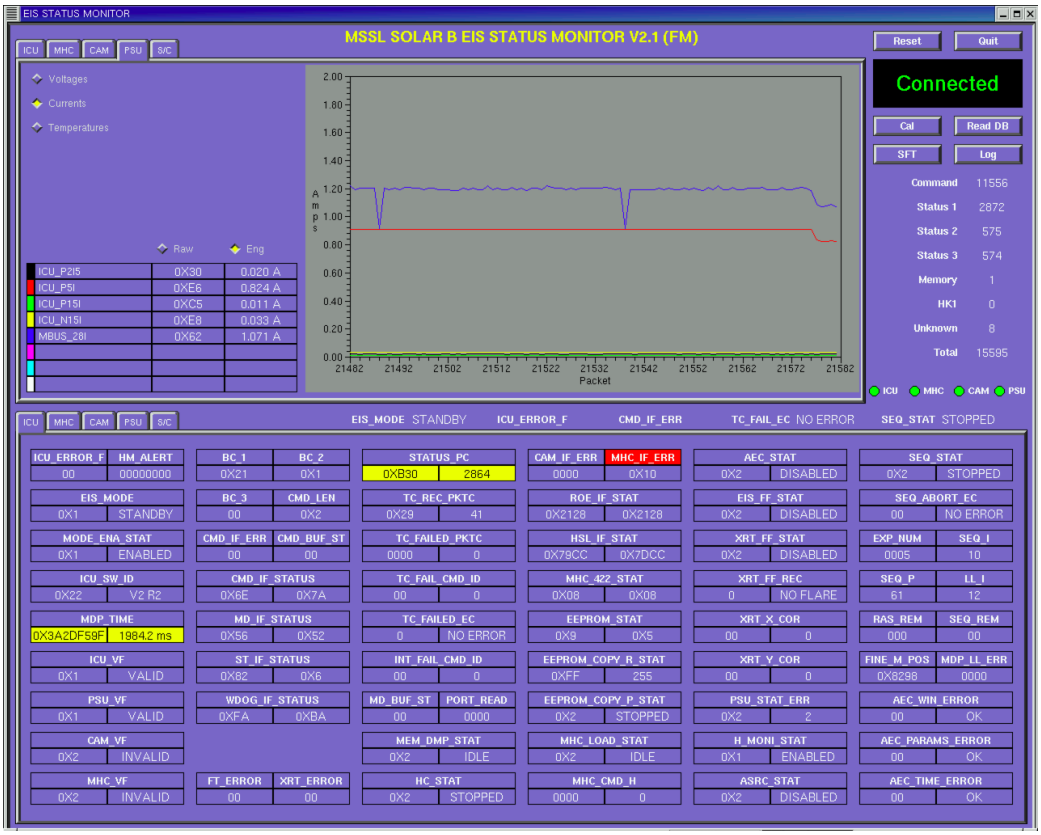


Fig. 26 EIS Status Monitor

Like the command window, the lower half of the EIS telemetry status monitor window (Fig. 26) also has a tab-selection for different sub-systems of the instrument. Each parameter has its mnemonic displayed at a fixed screen position, and below it are shown the received value and a converted value if appropriate. In some places two mnemonics are fitted in the space normally used for just one. The upper half of the window is used to select a number of parameters to be plotted on a graph for diagnostic purposes. The upper right of the display has four indicators to give a summary status of the main sub-systems, plus data packet counters and controls for the display.

2.8 Comment

Clearly there is a wide range of approaches here towards constructing the interface between the user and the instrument. Most of the software packages execute on the machine on which they are viewed, although this is largely unimportant given remote window techniques such as ‘X11’ which allow screen displays to be carried over public network links of indefinite length. One example uses a web browser as its display, freeing the user from dependence upon a particular machine. A mixture of computer languages are used, including C, C++,

IDL, and Java, Both text-based command lines and a graphics based interface are used for sending commands to instruments.

2.9 Usability

Chapter Five in this thesis makes specific assessments on six of the interfaces illustrated here, as a means of validating the design and analysis criteria developed in Chapter Four. Some general comments can be made, though. Of the three control and telemetry interfaces shown, the CDS may appear simplistic but this also reflects the limited software tools available at the time it was written. The Liverpool Telescope ECI makes good use of bold graphics and an uncluttered display, whilst the SOAR fits both a set of the most-needed parameters and a command interface into the same page at the slight sacrifice of a denser layout.

The science planning tools reflect that they have a prime role in assisting in the completion of an (electronic) form for planning an observation, rather than direct instrument control. This shows in the text bias of the presentation, with graphics being used to produce timelines, for example. The Hubble APT (and the Gemini OT with its common inheritance) allows its electronic form to be filled in by reference to international databases, and continuously checks for completed fields. It also has a comprehensive help system. In particular the presentation and operation of the graphics interface of the VTT tool (and the Gemini Position Editor) is very sophisticated, presumably due in at least part to the long gestation period and resources it has had to call on.

The XMM-Newton planning tool was very difficult and frustrating to use for the first few months after launch, as the web page interaction with a remote server proved to be a critical weak point. Response times on submitting data were one minute at best and frequently much longer, and multiple submissions were necessary to complete a proposal. Initially scientists found themselves taking up to a month to successfully submit a complete proposal (personal communication). The tool was neither responsive or forgiving, and in at least some opinions put the mission at risk. The current (early 2006) performance is much improved partially by adoption of a two-phase submission strategy to reduce the server workload, and partially by attention to server performance.The revised strategy filters proposals through a committee first, and only those accepted into the second phase then need to use the planning tool.

3 Exploratory Interface Programming

3.1 Introduction

In order to test some of the concepts of interface construction discussed above, and also in order to gain some insight into the actual processes involved in design and programming, the author set himself the task of fabricating a software simulator for an imaginary instrument. A spectrometer was chosen, as shown in the block diagram in Fig. 27 and described in Fig. 28 and Fig. 29.

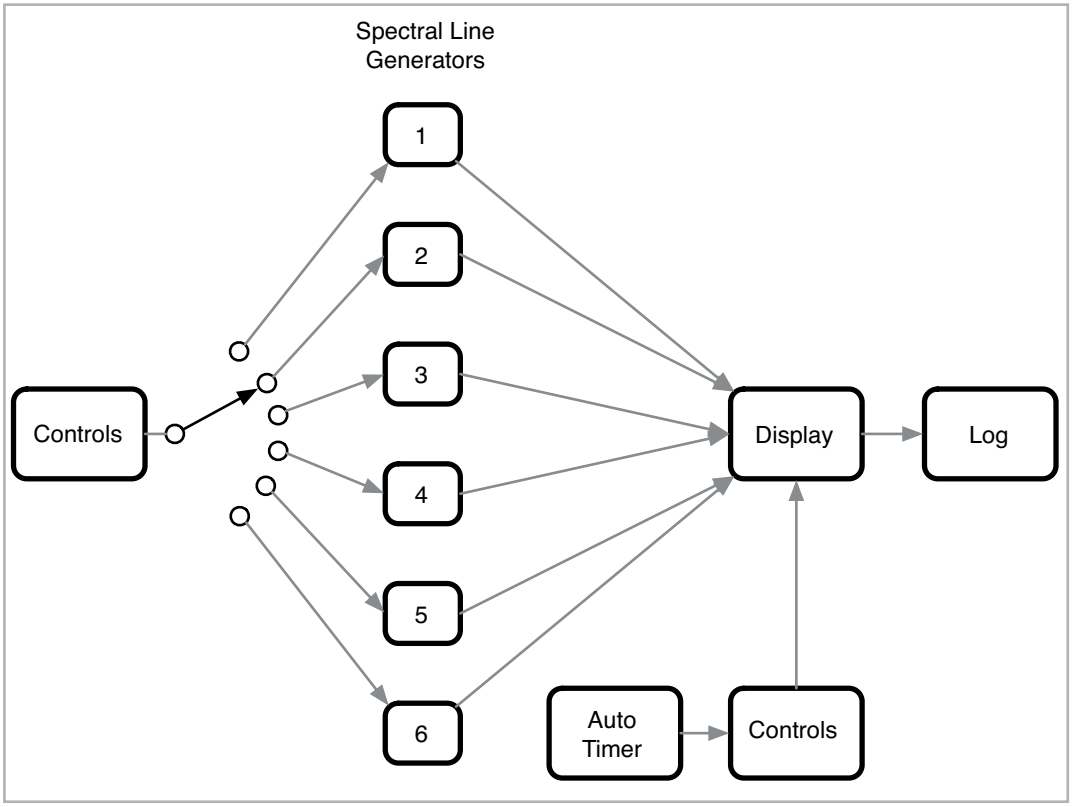


Fig. 27 Spectrometer Block Diagram

The spectrometer was developed using Java 1.4.2 under MacOS 10.3 and, whilst it performs as expected on a Microsoft Windows platform, might benefit from further understanding of how to optimise the visual appearance. Running on a Sun platform has yet to be tried. It was developed to a fixed deadline from a position of virtually no knowledge of Java, and so features were added on a prioritised basis as time and experience allowed. Part of the concept was to build experience of an object-oriented language and its tools, in order to allow further prototyping and demonstrations. It was developed using the Eclipse tool (Eclipse Foundation 2006), an open-source integrated development environment written in Java and initially coordinated by IBM. The iterative, on-the-fly compilation technique this tool

offered did much to improve the learning time for the work, and is an example itself of the iterative incremental technique suggested in this thesis for use in astronomy. The graphical objects are all items of the Swing class, part of the standard Java distribution.

All development files are available on the CD at the back of this thesis, as described in Appendix A section 2.

3.2 Graphical Object Terms

The terms for the graphical objects referred to here are the Java names. See Fig. 5 earlier in this chapter for the corresponding names for similar screen objects from Apple Computer.

Slider	control with an actuator moved along a (normally) straight scale
Button	an object that carries out an action when pressed. May be momentary or latching
Radio button	one of a set of buttons where only one of the set can be selected at a time
Spinner	an up and a down button with an associated text field
Text field	area for text. May be writeable or read only
Label	a non-writeable text area, typically describing another object
Progress bar	an indicator that moves to indicate action, such as elapsed time
Panel	a container for other objects

3.3 Overall Goal

The goal is to create an interface that mimics what a physical spectrometer might be like, with controls, displays and labels. As far as possible, it should be self explanatory in operation and be robust against incorrect user input. A science week event for about one hundred school sixth formers was the target for the first test of the as-yet unborn instrument. The anticipated user was someone used to operating simple instrumentation and looking at results on a graph. The event was four months in the future, giving a very firm delivery deadline if the opportunity were to be taken.

Willing experimenters were handed a sheet describing the purpose of the instrument and loosely describing the tasks they were asked to perform. Fig. 28 and Fig. 29 show the two sides of the sheet.

Spectrometer Simulator

If a thin cloud of gas becomes hot, it will start to emit light. One of the first people to observe this was J.J.Balmer in 1885, who showed that the light from hydrogen gas was actually composed of a series of single colours over the width of the visible spectrum and into the ultra-violet. Other people such as Paschen and Lyman showed similar line spectra in the infra-red and extreme ultra-violet. Each of these spectra are known as an emission spectrum.

If a hot body such as a star shines through a gas cloud the opposite happens - known as an absorption spectrum.

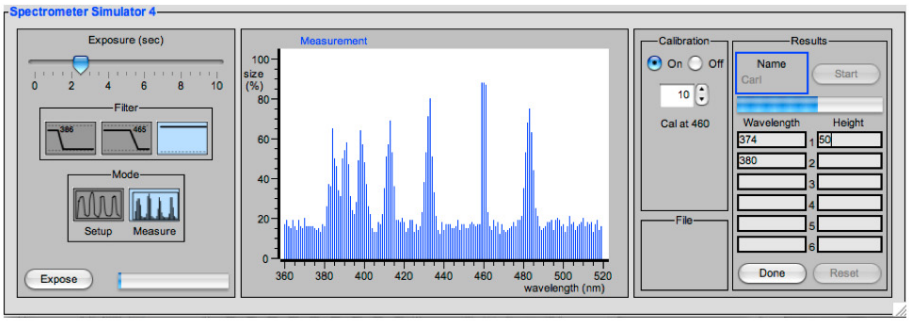
This is the emission spectrum of hydrogen in visible light:

and this is the absorption spectrum.

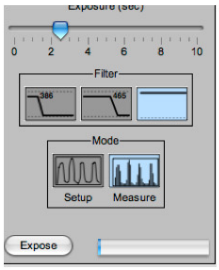
The lines are caused by the decay and excitation of electrons in discrete steps in the atomic shell.

Measuring the wavelength and intensity of these lines has all sorts of uses. Hydrogen is of particular interest to astronomers as it is found in abundance. The wavelengths of the lines are known to good accuracy and therefore form a method of calibrating the universe. If a group of lines are not in their expected position, it is likely the reason is a doppler shift caused by relative motion between the observer and the source. Here is a method then of measuring how fast the universe is expanding (or even contracting).

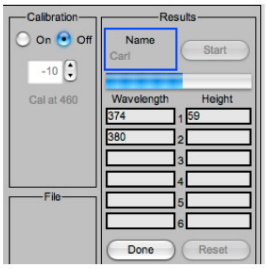
Fig. 28 Experimenter Sheet Side 1



People build all sorts of instruments to measure the parameters of spectral lines, and it's the controls and readouts of instruments that this research is involved with. What you see is the front panel of a spectrometer, which is bolted on the end of a telescope staring at a cloud of hot hydrogen gas between some stars. It's built to show six lines from the hydrogen series. You can vary the exposure to give a measurement neither lost in the noise or overloading the instrument, and you can use a convenient filter to separate the shortest wavelength line from the rest so you can measure it accurately.



The Mark 2 model - shown here - even has a mode to give a fast setup, and when you've set the controls correctly you then go back to the slow measurement mode for the real results.



Like all real technology, it's calibration drifts every time it starts up and so there's a calibration marker which you can turn on and off, and adjust the scale to match it.

Unusually there's even a place to write those results for each of the six lines. Don't forget your name - it doesn't have to be real, but please use something you think's unique.

Real astronomers have to work against the clock - although not usually as bad as the couple of minutes here. The bar on the right allows you to estimate the time left. Longer exposures give greater accuracy - if the instrument does not overload - but give you less time to try another exposure. Your international prestige increases if you finish early, but only if it's accurate. Once you've hit the Done button there's no turning back.

Fig. 29 Experimenter Sheet Side 2

3.4 Graphical Factors

The virtual instrument uses a low visual impact two-tone grey theme to give a level of consistency. Mid luminance colours allow both bright and dark areas to stand out yet give a structure to hold the graphical elements together. Blue is used as a highlight colour primarily as it is the default for the Java graphics used. A green shade is used for the simulation mode on the graph, to hint that it is different from the normal measurement.

In order to make the instrument as immune to false operation as possible, all commanding is achieved by use of graphical items such as sliders for continuously variable values and push buttons for on / off or momentary functions. Text fields are used to capture user written responses, one field per response value. All fields and controls are only allowed to be active when change is appropriate, and the inactive state is made explicit by changing their colour so as to present a lower contrast against the background than when active. This is frequently referred to as ‘greying out’. The highlight colour is used to signal both active and ‘in operation’ information. The inherent limited ranges of the graphical controls are used implicitly to limit the range of values that can be entered. Where a text field is available as part of the control, as for calibration, the control will still only accept the limited range of values that can be obtained by using the buttons. Buttons that change their function when operated, such as a single button toggling between on and off, are not implemented. Instead, a pair of radio buttons are used. This became important in using a computer for development with a trackpad instead of a mouse, as it was very easy to accidentally generate two operations in sequence rather than just one and the control would not perform as intended. A pair of radio buttons overcomes this problem.

The display area is broken up into four panels to separate the major functions of exposure control, display, calibration plus reporting, and presets. The presets panel is not intended to be seen by a normal user whilst the others are.

Functions that are closely related, such as user input fields or selection of filter response, are grouped together with a box drawn around the group. A label is used to describe the box. All of the text is from the same font family and most is the one size and colour. Where different in size or colour it provides a particular highlight or panel label. Limiting the choices of font reduces the mental effort on the user by removing the need to query why there are differences. Within this constraint, the size for all lettering is the minimum required for good legibility, in order to maximise use of the panel area.

There is currently some inconsistency in greying out boxes that are grouping objects. The presets panel has a control (the lines tab) that currently is still highlighted when inactive due to programming problems.

When significant time delays are involved, such as in waiting for the exposure, a progress bar is provided to give feedback to the user on how much time is left. The bars are set so that in their quiescent condition a small portion of the bar is visible so that the function is clearer than that of just a white strip. Insufficient space was left for the reset button to have a progress bar, and so the text on the button is modified instead.

Where appropriate, graphical descriptions are used on the controls, for example on the filter buttons, to present information in a succinct fashion.

At the start of operation or after reset the controls are set to give a usable instrument to the operator. (Presets panel controls are not changed at a reset, but would normally be hidden anyway).

3.5 Software Architecture

The software is written so that it can run either as a Java applet inside a web browser, or as a stand-alone application. Exactly the same file is used for both situations, with the startup method selected to suit the occasion. Startup as an applet always enables the presets panel; startup as an application allows a choice of how the presets panel is configured from a command line parameter. The default is not to show the presets panel at all, as this would be the normal user mode.

There is just one explicit warning, a symbol that appears in the File area if the Java security manager does not allow access to the machine file system. This will happen if the simulator is run as an applet from a web browser, but not if it is run as a stand-alone application. A browser is considered a security liability by the Java virtual machine. The symbol appears as an icon with a line drawn through it, and drawn in red. This is considered a reasonable means of indicating an inability to access the file system.

The file system is used to log the user input. A text file is written, using XML-like tags to facilitate machine reading later, giving the name of the control used and a timestamp. The user entries in the text fields are stored also, along with the actual settings of the controls that users were trying to match (see intended operation, section 3.8, below). Full details are shown in Appendix A, including the file path. This path is not currently user selectable.

Of some relevance scientifically, the data displayed on the graph is all generated from one or more pseudo-random noise generators and then shaped (except for the background noise) with a Gaussian profile. This profile can be changed, either individually for each spectral line or with all the profiles locked to the same value. This latter mode mimics the effect of a point spread function of a real optical system.

3.6 Design

The instrument mimics what a spectrometer constructed to analyse several spectral lines from the Balmer series might do. It generates six Gaussian-shaped lines from random noise, with each line programmable for wavelength and count rate. The wavelengths are preset to the Balmer series, but with a small amount of random error added at every restart. The background noise is programmable with two parameters. The detector and thermal noise contribution is proportional to the square root of the exposure time, whilst the system noise is independent of exposure time. This matches the situation where an increased exposure will usually improve the signal to noise ratio of a detector.

The exposure time is adjustable by the user and a progress bar shows the proportion of elapsed time after pressing the Expose button. Another timer controls how long the user is allowed to try various ideas and settings before the session times out. This mirrors the situation in real life where an astronomer has a finite allocated time on an instrument. This time is pre-settable in the presets panel. After the time-out and after a 10s delay the instrument resets ready for the next user. The reset delay is shown as a countdown in the 'Reset' button. The 'Done' and 'Reset' buttons allow the session timer and reset timer functions to be completed early by the user.

Three filter buttons are provided to select either a flat wavelength response, or a low-pass response with two different turnover points. Their action is to filter the data in a numerical manner, just as a physical filter might act on optical data. The concept is to allow accurate measurement of close spaced spectral lines.

Two modes of using the instrument are implemented, with the idea that the user can first make a series of quick setup actions which use an (implied) stored database of previous measurements and which display a result immediately. Once this is satisfactory, the user can select the measurement mode, which uses an (implied) real detector and which takes the actual indicated exposure time to make the reading. To emphasise the difference between the two modes, 'Setup' draws the result with a green line graph, whilst 'Measure' uses a histogram in the default blue.

The plot is drawn in red to imply detector overload if it would otherwise be drawn off the top of the display.

Measurements can be taken from the graph by visual alignment against the scales, but it is much easier to use the two axis readout that is also provided by the cursor. This is controlled to only appear over the graph panel, and to only show numerical values over the drawn graph.

Finally, as all real instruments drift, at every restart a small offset is added or subtracted from the wavelength scale. A calibration source can be switched on at 460nm and the scale re-adjusted to align with it. If the 'Setup' mode is selected, the calibration can take advantage of the fast response available.

The presets panel "Exposure Delay' buttons allow the operation delay defined by the 'Exposure' timer to be set to a minimum for development and testing.

The "Presets' buttons simply enable or disable the presets panel controls.

3.7 Startup

When started from a command line in a terminal window, a choice of parameters can be passed. The syntax is:

java -jar InstrAp4.jar <parameter>

where the <parameter> term is, without the brackets, one of:

setup_on	- displays the presets panel in an enabled state
setup_off	- displays the presets panel in a disabled state
no_file	- marks the file system as unusable
no_shifts	- removes the random line shifts at restart time
no_mode	- does not show the mode buttons, allowing 'Measure' mode only
(none)	- if no parameter is supplied it defaults to no display of the presets panel

This only needs to be performed at the start of each session and could be automated; the spectrometer resets itself to be ready for the next user.

3.8 Intended Operation

The intention of the design is for user action to happen in the following manner. The goal is to measure the wavelengths and count rate of all the lines in the most accurate way.

The user optionally fills in a name in the 'Name' panel on the right, then presses the 'Start' button. This starts the session timer. First the user calibrates the instrument by turning the calibration signal on with the 'Calibration' 'On' button. Pressing the 'Expose' button gives a display showing the calibration signal. The user then adjusts the 'Cal at 460' control to set the graph scale up or down, and presses the 'Expose' button again. This is repeated until the signal agrees with the scale at 460nm, and the calibration can be turned off.

The user then can set the exposure time and make an exposure. The process is repeated to use as much of the graph height as much as possible to maximise the measurement accuracy, without saturating the 'detector'. The cursor can then be used to read out wavelengths and counts, which can be manually entered in the text boxes on the right hand side. The shortest wavelength line can be measured more accurately by using a filter to prevent it being swamped by an adjacent signal.

With all the results entered, the user can then either press the 'Done' button or wait for the time-out. Their results along with the actual programmed values will be displayed over the graph window briefly, and written to a file if the file system is available.

The whole process can be significantly speeded up by selecting the 'Setup' mode initially, and then selecting 'Measure' for taking the actual readings.

Fig. 30 shows the normal user's view of the instrument, caught in the middle of a set of measurements.

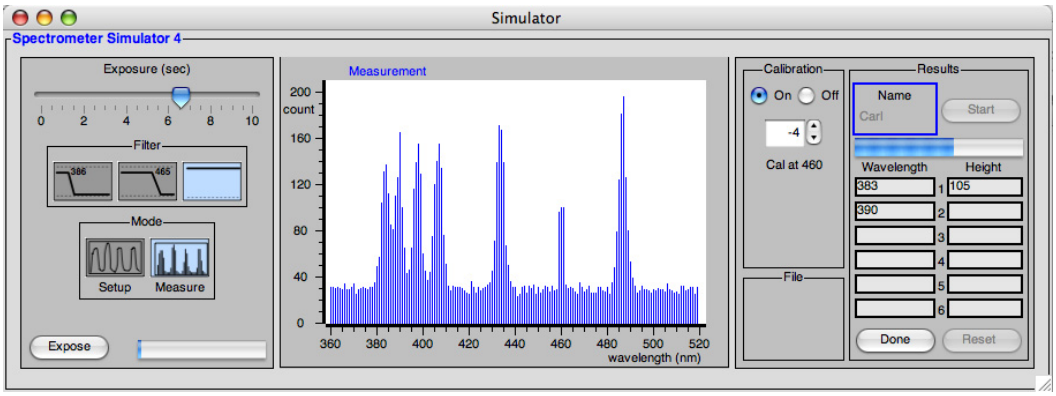


Fig. 30 Spectrometer User Panel

Fig. 31 shows the view of the presets panel as well.

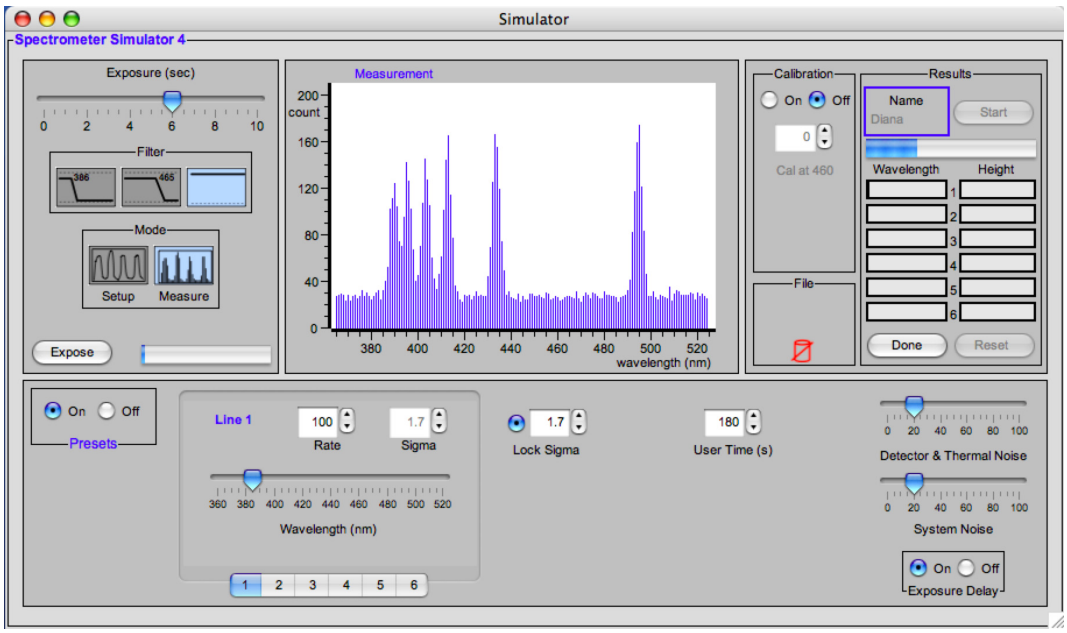


Fig. 31 Spectrometer Including Presets Panel

3.9 Results

A typical set of filed results from one run is shown in Appendix A, taken when re-running the software to obtain the diagram used in Fig. 30. The results from the sixth form science day were inconclusive, partly due to problems with the results logging software on the day and partly due to a smaller than expected sample set due to unforeseen local arrangements.

3.10 User Testing with a Simulator

Creating this simulator gave a good insight into the processes involved in writing software with a significant graphics content for user interface purposes. It (not unexpectedly) becomes very easy to fixate on pixel-perfect graphics at the expense of the rest of the design. The use of the Eclipse development tool was intriguing in the context of this thesis, as its incremental mode of operation essentially gave continuous compilation of the developing software, rather than the 'job submission' mode that is often associated with software development. This gave continuous feedback which meant that mistakes were spotted immediately and corrected on the fly. This is precisely the aspect of operation of an instrument simulator that this thesis argues for in Chapter Seven, for much the same reasons. Considering that this is a tool that appears to be rapidly gaining in popularity, the use of the incremental technique in this software tool gives evidence, albeit circumstantial, that this is a promising route to follow.

Java was a good choice for this exercise, giving predictable, stable results that were cross platform and could be delivered by network download or a local data store. Like many environments that offer a wide range of inbuilt facilities, one of the most difficult parts of the process is to become aware of exactly what is available in all the libraries that are there for the browsing.

The lack of definitive results from the sixth form day was a sharp reminder of how easy is it to spend a great deal of time trying to obtain results from user testing. Users can be inconveniently self-willed, and unless one has plenty of help and organisational support the validity of results that are gathered may be questionable. This is an important contributory factor in the attraction to the author of the persona technique described in Chapter Four, which by its very structure minimises the amount of user interface testing for a given design task. The inconclusive results of the spectrometer were actually a spur to look for an alternative way to tackle the definition of user interfaces.

In reviewing this user test, one other conclusion is that too much was expected of the users on the day in the time allowed to them. Unless an individual had a background familiarity with instrumentation and with the idea of spectra, it might have been perceived that there was too much to learn, leading possibly to operation based on guesswork. The conclusion to be drawn is to test interface ideas with people who have a similar skill set to the intended final user. Since space science skill sets might be considered as rare, the alternative is to explicitly spend time with potential subjects before a test ensuring that the necessary background information is understood.

4 Summary

This chapter has looked at some of the human aspects of general purpose interfaces, such as ease of use and the idea of mental models to consolidate a concept. All manner of pointing and display devices are available to help to achieve this, and each operating system vendor produces a toolbox of moderately standardised tools with which to implement the on-screen display. Use of the expected affordance for a control improves its usability, as does promoting good operator response times by ensuring that the size and placement of those controls takes Fitts' and Hick's laws into account. Use of colour, with appropriate selection of hue, saturation and brightness can assist in making aspects of a display clearer, and aesthetic principles should not be forgotten. Following this a brief survey of static snapshots from the context of space science give a flavour of examples of user interfaces from the last decade or so. Finally a useful exercise in exploring the creation of a working user interface and capturing results from it is described. Whilst the results are inconclusive, it gives a

valuable insight into user interface design, the typical technology behind that design, and aspects of user testing.

The next chapter moves from an introduction on interface detail to focus on several examples from diverse fields where deficiencies in the user interface have led to very serious system failures, including loss of life.

Chapter Three: Human - Computer Interface Case Studies

Human Computer Interface issues are becoming more accepted as important facets of system design. The five case studies in this chapter take a number of well-known instances where inappropriate design has caused unexpected results and led to system failures that were both technical and political. They are intentionally wide-ranging in scope in order to identify a large number of issues. The details are then analysed to extract common HCI factors contributory to the incidents, such as 'insufficient training' and 'presentation of information'. These are used to build a list of questions which can be used to help analyse a system for potential weaknesses. The studies are:-

1. Three Mile Island nuclear power station accident, March 1979
2. Air accident at Kegworth near East Midlands airport, January 1989
3. Air accident near Strasbourg airport, January 1992
4. SOHO spacecraft loss of attitude control, June 1998
5. US presidential election Palm Beach ballot, November 2000

1 Case Study - Three Mile Island

The Three Mile Island (TMI) accident in 1979 is often quoted in case studies of accidents but this does not change the fact that many of the concepts discussed then are just as relevant today. The plant site is on a sandbar in the Susquehanna River in Pennsylvania about ten miles from the state capital Harrisburg. The immediate area is farm land with several towns within a radius of a few miles. The population of Harrisburgh is about 50,000 whilst several million people live within an area that might be rendered uninhabitable if a serious failure of containment of the radioactivity at the plant occurred.

Two reactors were built on the site in the late 1970s. TMI-1 is still operating (2006) and has a good safety record, and TMI-2 had been operating for a little under a year when the accident occurred. The plants were built to a Babcock & Wilcox (B&W) design as a pressurised water reactor and owned by Metropolitan Edison, currently (2006) known as First Energy. The Nuclear Regulatory Commission (NRC) was responsible for licensing the operation and ensuring operating standards were maintained. TMI-1 was sold in 1999 to Amergen, a company 50% owned by British Energy, for $100m.

The accident was a turning point for the public perception of nuclear safety in the USA and beyond. Prior to the event, perception ranged from one of nervousness based upon an unquantified fear to one of happy acceptance - "Electricity too cheap to meter". In the time since, despite a major revision of safeguards and training by the NRC, no new nuclear power

stations have been built in the USA. The expectation is that TMI-2 will never be re-opened, and that the site will be cleaned up when TMI-1 is decommissioned. This is currently expected not before 2014. The plant was operational for less than 4% of the expected 25 year life.

The combined capacity of the two rectors was 1,700 MW, suitable for supplying about 330,000 homes. Each reactor generates heat by the use of a moderated chain reaction. The TMI-2 design used 36,816 fuel rods, formed as vertically-mounted assemblies including control rods for moderating the neutron flux and tubes with instrumentation. The control rods are raised up to accelerate the reaction and vice versa. In an emergency (a 'SCRAM') the electromagnet holding them up is disabled and they are allowed to fall under gravity fully into the reactor, inhibiting the chain reaction. The heat generated in normal operation is removed by a pressurised water primary cooling circuit at 2,155 psi and used to heat a secondary cooling circuit via two steam generators. The circuit drives a turbine which powers an electrical generator, and the steam is condensed and recycled. The condenser uses a tertiary system of cool river water to remove heat from the steam, and this hot river water is pumped to the top of the cooling towers and allowed to cool before being returned to the river.

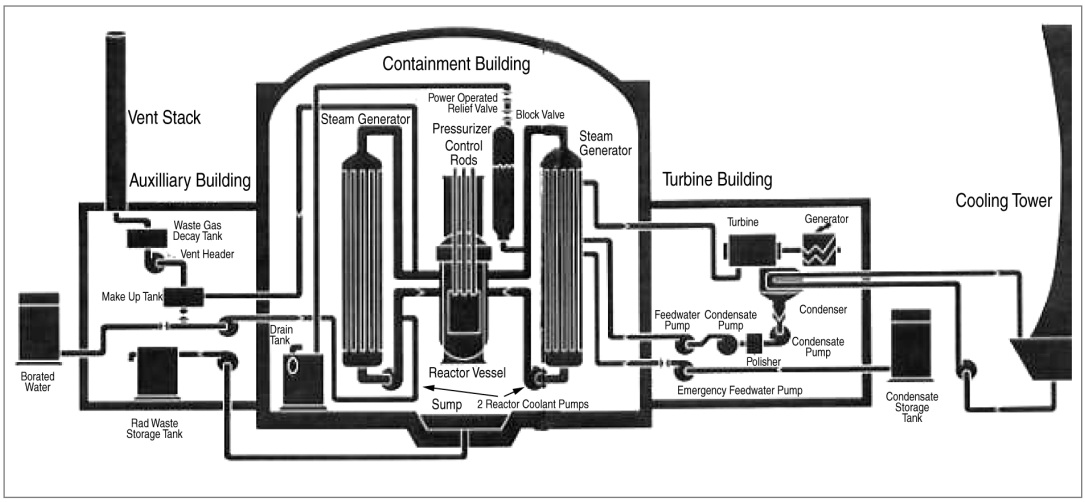


Fig. 1 TMI-2 Functional Diagram (Kemeny 1979)

TMI-2 (Fig. 1) suffered a partial melt-down in March 1979 when essential maintenance caused a secondary cooling system failure. The pumps supplying feedwater to the steam generators stopped and the resulting pressure increase in the primary coolant circuit caused the power operated relief valve (PORV) to open as intended. The reactor control rods were dropped automatically to stop the chain reaction, but this still required the dissipation of a large amount of heat in the core. Emergency feedwater pumps were started but were ineffective as two feed valves were incorrectly set closed.

The PORV meanwhile failed to close properly after the over pressure had been relieved. The resultant drop in pressure in the primary coolant triggered the high pressure injector (HPI) pumps to flood the reactor core with water to prevent it becoming uncovered. Without cooling the reactor core could melt, fall to the floor, fracture the building base and cause a very large escape of radioactive material. The operators however mis-read the situation and manually drastically reduced the HPI rate after two minutes, fearing over-pressure in the primary cooling circuit. The two valves preventing the emergency feedwater were then noticed and opened. After a while the vibration caused by the steam and water mix required that the pumps were turned off again and the reactor core continued to over-heat.

The stuck PORV was noticed by a new operator two hours and twenty minutes into the emergency and a valve to block the flow was closed. Some cooling of the core became possible, but was limited by pump malfunction due to the steam and water mix in the pipes. The core was eventually left to cool by itself. An explosion of hydrogen gas liberated by a reaction between the water and the fuel rod cladding happened about nine hours after the initial events, but fortunately did not breach the containment building structure. Subsequently a large bubble of hydrogen was generated but did not ignite. During the series of events there was some release of radioactivity into the environment locally, but opinions differ widely on how much and the consequent effects. The reactor was allowed to cool and will eventually be decommissioned. There are no plans to attempt a repair.

The reports mention only a few facts appropriate to the study of HCI issues but they are relevant. The era from which the design came meant that computer driven displays were basic by current standards. The control panel was a large room, 12.8m x 12.2m, and used mechanical switches and individual indicators. The status of the two unintentionally closed valves was displayed, but one of the indicators was hidden by a repair tag on the switch above it. The relief valve position was difficult to monitor and so the display on the board only indicated the commanded status, not the actual status. The computer monitor was slow enough that the printout it generated was at times a long way behind the actual events. One report quotes about one hundred alarms ringing at the same time which must have lead to severe confusion.

1.1 Inquiry Extracts

One report on the inquiry from Bignell and Fortune (Bignell 1984) had the following comments on the control room ergonomics (see also (Perrow 1985) and Fig. 2):

- For one piece of equipment, the pressure was displayed on panel 10, the temperature on panel 8, whilst the control that could be used to halt undue changes was on panel 4. Indication of the flow of make-up water was on panel 8 but the control for it on panel 3.

- Indicator lights associated with particular controls were placed variously above, below or to one side of them.
- On some controls a clockwise turn switched operation from manual to automatic; on others the reverse was the case.
- At the emergency feedwater control station, the locations of the control did not mimic the actual valve and pump positions in the plant and the relative layout on the panel was inconsistent.
- Nearly seventy items in frequent use were out of reach so that in leaning over for them the operator could inadvertently knock a switch, or would not be able to see on a distant gauge the effect of his control actions.
- There was poor utilisation of the available space. Some controls and displays were unnecessarily large whilst others were too small in relation to their importance. The pressuriser water level was one particular example.
- Some important indicators were absent, such as the emergency feedwater flow and the flow in the discharge line from the PORV.
- Other displays were out of sight of the operator or had no visual correlation to their associated controls. For example, the vibration indicators for the coolant pumps were on the rear of a control panel.
- Only the top half of sixteen rows of indicator lights could be seen from the normal operating position.
- Coolant tank instrumentation was outside the main operating area. Greater prominence could have lead to earlier correct diagnosis of the leaking PORV.
- Critical alarms were not colour coded or graded in priority. Legends were excessively wordy or used inconsistent abbreviations.
- The colour of a light did not carry a specific meaning. For example, a red light could mean one of fourteen different states.
- Some lights were white against a white background or had poor contrast against adjacent lights.
- A multiplicity of flashing lights obscured the overall picture.
- Well over one hundred gauges suffered from potential parallax reading errors.

Fig. 2 TMI-2 Control Room (Rogovin 1980)

The following quote from the Kemeny report (Kemeny 1979) highlighted a deficiency inside the regulatory commission itself:

> Section G, Para.8.e:
> *There is no office within NRC that specifically examines the interface between machines and human beings. There seems to be a persistent assumption that plant safety is assured by engineered equipment, and a concomitant neglect of the human beings who could defeat it if they do not have adequate training, operating procedures, information about plant conditions, and manageable monitors and controls. For example, despite recognition within NRC and various industrial groups that outdated technology in the control room could seriously handicap operators during an accident, NRC continues to license new plants with similarly deficient control rooms. As noted before, problems with the control room contributed to the confusion during the TMI accident.*

The report found that the situation had actually happened before:

Section A: Para 7a:

In September 1977, an incident occurred at the Davis-Besse plant, also equipped with a B&W reactor. During that incident, a PORV stuck open and pressuriser level increased, while pressure fell. Although there were no serious consequences of that incident, operators had improperly interfered with the HPI, apparently relying on rising pressurizer level. The Davis-Besse plant had been operating at only 9 percent power and the PORV block valve was closed approximately 20 minutes after the PORV stuck open. That incident was investigated by both B&W and the NRC, but no information calling attention to the correct operator actions was provided to utilities prior to the TMI accident. A B&W engineer had stated in an internal B&W memorandum written more than a year before the TMI accident that if the Davis-Besse event had occurred in a reactor operating at full power, "it is quite possible, perhaps probable, that core uncover and possible fuel damage would have occurred."

1.2 Presentation of Information

The TMI control room (Fig. 2) seems to have been poorly thought out, with indicators often separated from the controls they related to. There was little consistency of direction of operation of controls or colour of indicators, and wasted space leading to larger control panels than necessary. The photos show alarming visual clutter leading potentially to operator confusion, and the multiple audible alarms would have added to the confusion. Incorrect association of (for example) a control and indicator due to parallax errors caused by viewing a panel from the side appear to be a real risk. Poor layout leads to increased operator stress, whilst crisis management requires calm clear thinking. One must assume that the control room was designed without allowing for the management of a crisis. It seems that at the time (at least) the regulatory framework of the NRC had little concept of human factors in engineering.

There is little evidence of any use of interlocks in the control room design. Just as a car with an automatic gearbox cannot be started with the drive engaged, one would have thought that TMI could not have run without the emergency feedwater enabled.

1.3 Timeline and Fault Tree

The following timeline (Table 1) is derived for this study from the original Kemeny report (Kemeny 1979) in order to give a set of facts from which to derive a fault tree (Fig. 3). Both are written with the aim of bringing out the HCI issues involved in the TMI accident. It is notable that the initial events are very close together in time.

Date & Time	Event	HCI Comments
Wed 28 March		
	Polisher operation allows water into air controlled valve system	No understanding of risk by operator
04:00:36	First pump trip. No feedwater to steam generators	Initiating event
04:00:38	Turbine and generator shut down	Automatic
	Power operated relief valve (PORV) opened at 2,255 psi	Automatic
04:00:44	Reactor scram	Automatic
	3 emergency feedwater pumps started	2 closed valves not noticed. Service tag covered one indicator
04:00:49	PORV should have closed at 2,205 psi. Valve stuck open.	Indicator showed command status, not valve position
04:00:50	Emergency pump operation noted in log	Note
04:01:24	Pressurizer level started rising again	
04:02:21	Steam generators boil dry	
04:02:36	Pressure drop in prime coolant system starts high pressure injector (HPI) pumps	
04:04:36	Operator drastically reduced HPI injection rate	Loss of coolant through PORV not understood
	Operators start draining through let-down system due to high pressurizer level	High pressurizer equated to high pressure
	Operator opened emergency feedwater valves ('twelve valves”)	Valve status not clear
04:11:00	High water alarm in containment building sump	
04:15:00	Rupture disc on drain tank burst sending more water into sump	
04:20:00	Neutron count started increasing, implying reactor core being uncovered	Not understood by operator?
04:39:00	Sump pumps stopped by operator after about 8,000 gallons on water pumped to aux building	
05:00:00	Severe vibration in 4 reactor coolant pumps due to steam and water mix.	Steam + water mix not understood
05:14	Two coolant pumps shutdown by operator	
05:41	Two remaining coolant pumps shutdown by operator. No reactor cooling	
06:00	Radiation alarms inside containment building indicate some fuel rod cladding rupture. Hydrogen generated.	
06:22	Block valve for PORV shut by operator, stopping loss of coolant	Insufficient operator training
06.30	Rapidly rising radiation levels found in auxiliary building	
06.48	Subsequent evidence shows maybe 8 ft of 12 ft core uncovered at this time	No means of monitoring this directly
06.54	One reactor coolant pump turned on.	
07:00	Site emergency declared. Station manager arrives and takes command.	
07:13	Coolant pump turned off because of high vibration.	
07:15	Auxiliary building evacuated.	
07:20	Radiation dome monitor alarm at 800rem per hr.	
07:20	HPI pumps turned on again.	
07:24	General emergency declared.	
07:38	HPI pumps turned off.	
08:00	Containment building automatically isolated at about 4 psi overpressure in containment building.	

Table 1 TMI Accident Timeline

Date & Time	Event	HCI Comments
08:26	HPI pumps turned on again.	
10:30	Core fully covered with water. TMI control room declared hazardous.	
11:38	Operators open block valve and reduce HPI pump rate to reduce pressure.	
13:50	Hydrogen explosion inside containment building, but not understood as such.	
15:08	Attempt to reduce pressure abandoned. Core partially uncovered.	
Thur 29 March		
~14:00	Dumping started of slightly radioactive waste water into river	
18:00	Dumping stopped.	
20:00	Realisation of severe core damage.	
Fri 30 March		
00:00	Dumping of waste water restarted, to prevent overflow.	
07:10	Radioactive gas transfer to waste gas decay tank	
08:01	1,200mrem per hour measured 130 ft above vent stack.	
12:30	Public advisory issued for pregnant women and children inside 5 mile radius.	
All day	Growing public concern and confusion; evacuation plans prepared.	
Afternoon	Realisation of existence of hydrogen bubble of about 1,000 cu ft above reactor core.	No direct monitoring of an unexpected situation
Sat 31 March		
03:00	Agreement reached with medical company to manufacture potassium iodide for radioactive iodine protection.	
All day	Estimates made to understand explosion risk of hydrogen with oxygen liberated by radiolysis	
Sun 1 April		
01:30	First iodine arrived of total 237,013 bottles.	
13:00	US President Carter toured plant. (Carter trained as nuclear engineer)	
Late afternoon	Signs that bubble was diminishing in volume.	
Mon 2 April		
	Debate over distribution of potassium iodide. Not distributed for fear of inducing panic.	
Tues 3 April		
	Continuing public concern.	
Wed 4 April		
	Schools within 5 mile radius remain closed, all other restrictions lifted.	
Sat 7 April		
	All restrictions removed on public movement.	
Future (2006 +)	Release of radioactive gas in containment building, decontamination, decommissioning.	

Table 1 TMI Accident Timeline

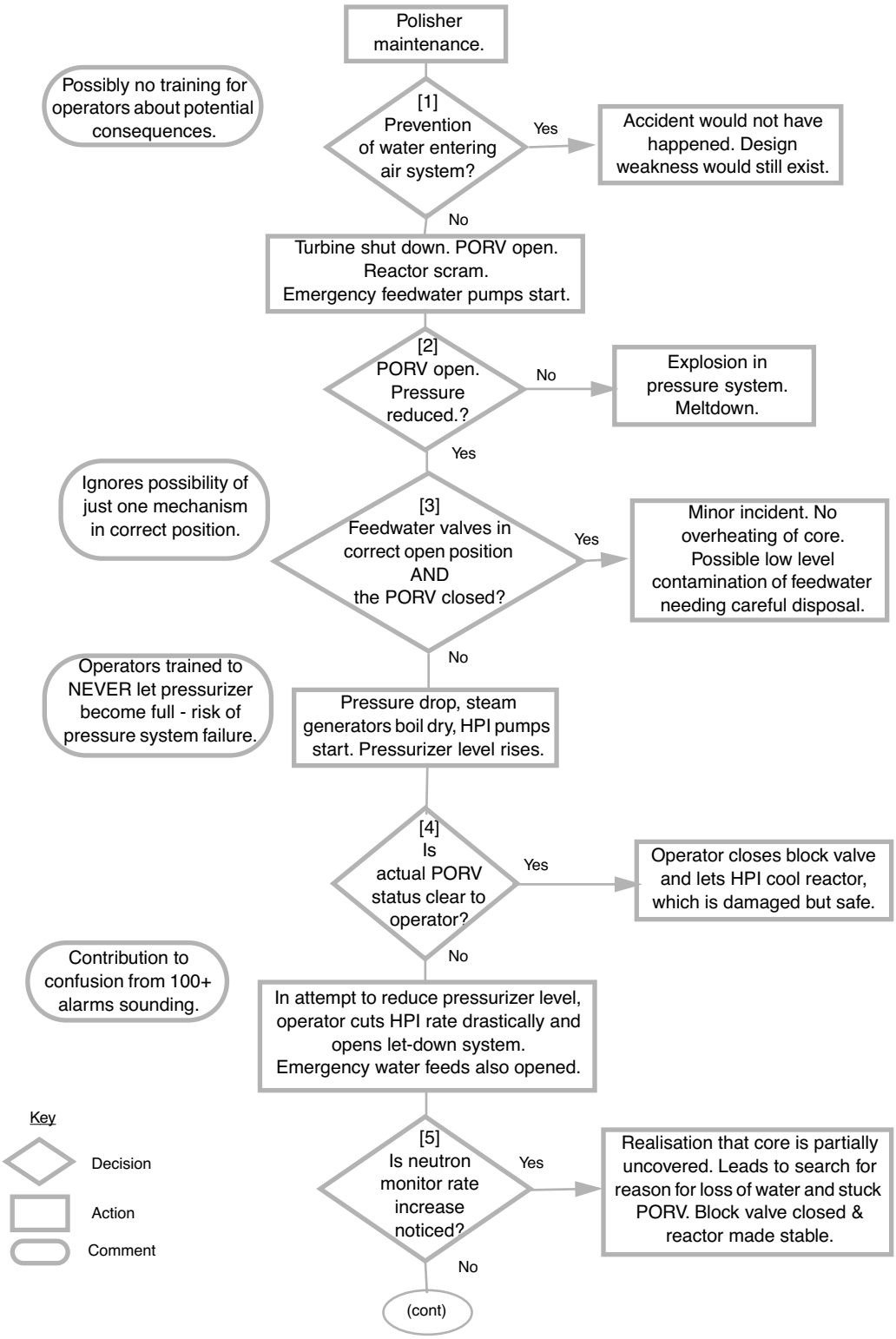


Fig. 3 TMI Fault Tree

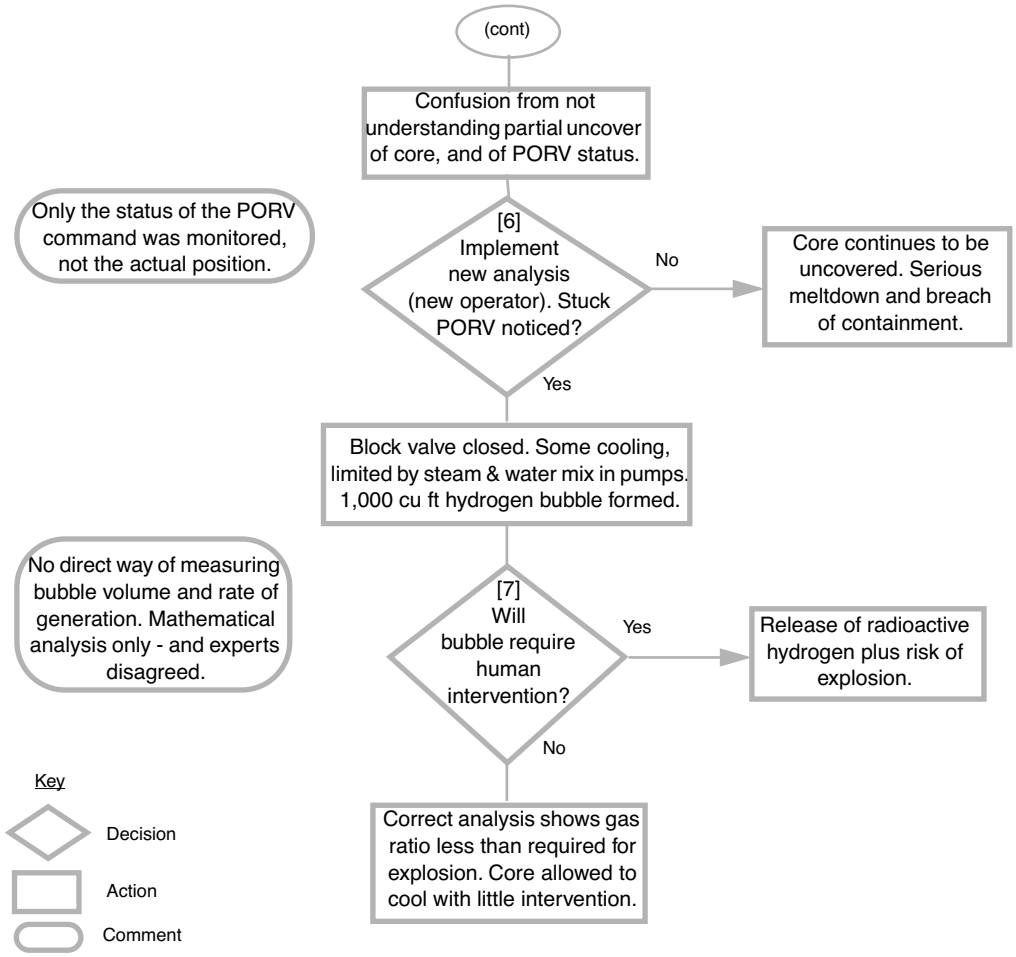


Fig. 3 TMI Fault Tree

1.4 HCI Issues Derived from Timeline and Fault Tree

The causes related to human machine interfaces for the TMI accident seem to split between inadequate training and poor control room design. A simple fault tree (Fig. 3) derived from the accident timeline (Table 1) shows seven critical decision or branching points.

At [1], the polisher used to keep the secondary cooling pump from becoming fouled with resin had failed and in attempting to clean it the operator allowed water to enter the pneumatic valve controlling the feedwater pump. This immediately shut the pump down removing the reactor cooling and starting the cycle of events. Whilst other mechanisms should have stopped the problem becoming an emergency, it seems reasonable to suggest that the operators had insufficient training in assessing the risk of what they were undertaking. At the very least, a full check of the controls beforehand should have happened which would have revealed the closed emergency feedwater valves. The plant could have

been run at reduced power to give a greater safety margin than the 94% at which it was functioning. The design weakness would still have existed, but the risk would have been better managed.

If the PORV had failed to open at [2], the over-pressure would eventually have caused a failure of the primary coolant circuit. The role of the PORV was well understood here and the design had only a very small chance of failing to open. Rupture of the primary cooling circuit could only have been managed by the high pressure injection (HPI) of cooling water. There are no particular HCI issues here.

At [3] and [4], if both the PORV had closed and the feedwater valves had been open, the incident would have been a minor one. The emergency feedwater might have needed disposal as it is unclear how this might have been recycled. Lack of a direct monitor of the PORV status was a direct contribution to the accident, although training in the use of the PORV block valve for diagnosing a problem would have minimised the risk. It is clear from the inquiry report that the operators did not realise the valves were closed until four minutes had elapsed and the control room layout must be the prime cause for this.

The design of a control room with the potential for “over one hundred” alarms to sound at once seems incompetent. The aural confusion induced must have been contributory to the accident.

At [5], the neutron monitor is used as one indicator of the health of the reactor. A reactor that has a falling water level will have a rising neutron count (other things being equal). The monitor seems to have been missed. The information from it might have caused an operator to look further and discover the PORV stuck open. This may have been both training and an HCI issue; the relevance of the information may not have been made clear, and it may have been difficult to see.

Decision [6] is a training issue. If you’re in a hole, stop digging and ask someone else to give you a hand. It took 2 hours 22 minutes for a new operator to arrive and he spotted the PORV problem quickly. In the meantime a great deal of damage had happened to the reactor core. Without the PORV closed a breach of containment was likely with huge radiation releases.

At [7], about three days after the start of the incident, several NRC scientists spent a lot of time calculating whether the oxygen released by radiolysis of the water would lead to an explosive hydrogen and oxygen mix in the containment chamber. They failed to agree and fortunately the majority view of insufficient oxygen was accepted. The alternative was the risky process of bleeding off radioactive gas. It sounds like this scenario was never

anticipated, as otherwise a tube through which a gas sample could be drawn could have been provided.

Background information on TMI also came from (Moss 1981), (Rogovin 1980) and (Myrddin Davies 1979).

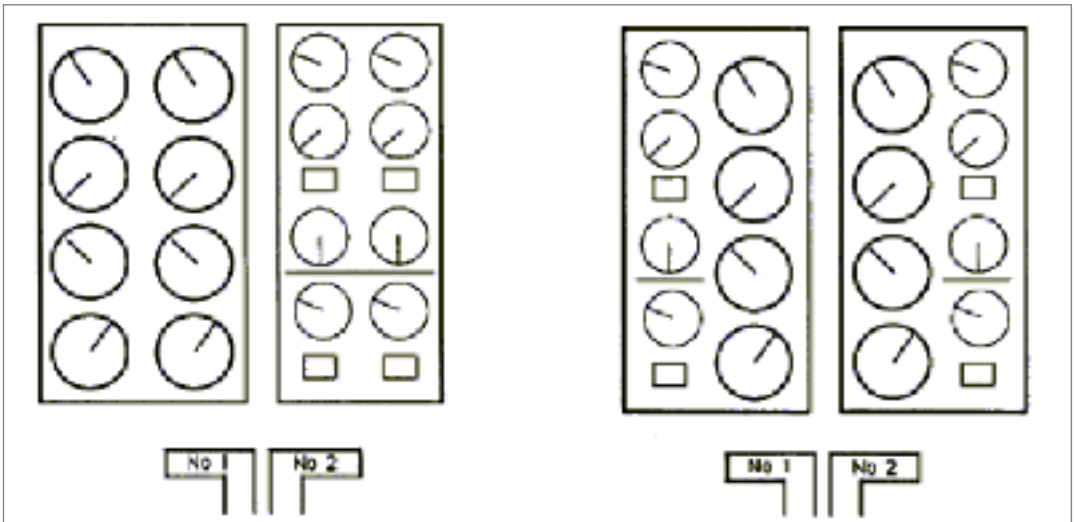
2 Case Study - East Midlands Air Crash

In January 1989 a Boeing 737 with a declared engine failure just failed to make an emergency landing at the UK East Midlands airport near Kegworth and crashed into a motorway embankment. The pilots had been aware of a problem soon after takeoff with severe vibration and smoke in the cabin. In diagnosing the problem, the pilots shut down the starboard (No.2) engine when it was the port (No.1) engine that was failing. There were several contributory factors to this decision, but this discussion concentrates on the HCI issues.

The analysis of the engine at fault came primarily from the instrumentation on the flight deck backed up by an expectation of how the cabin smoke was being routed. As the pilots shut down the starboard engine the vibration ceased, giving a most unlucky false positive. The other engine, the faulty one, stopped vibrating at that time due to the disconnection of the autothrottle, which resulted in a small reduction in power demand and stabilisation of the engine. This fact was not appreciated by the flight crew. It was clear to several people and cabin staff in the passenger cabin that the pilot announced the wrong engine shutdown as they had witnessed flames from the other one, but no one felt it appropriate to challenge the authority of the flight deck. About 50s before impact the No.1 engine failed completely and the first officer attempted to restart the No.2 engine. By this time there was insufficient power available from the No.1 engine to enable a restart, and the airspeed was too low for the engine to be started by itself. The aircraft crashed as the airspeed was too low to sustain flight.

The instrumentation layout (Fig. 4) was criticised during the inquiry. The diagram shows the primary engine monitors as the larger dials, with the secondary monitors as the smaller dials. The left and right throttle positions are shown below them. The 737-400 concerned had been equipped with electronic engine management indicators which visually mimicked the old mechanical indicators that they replaced. The secondary engine information set for both engines was grouped together and placed to the right of the primary set, rather than being grouped symmetrically around the primary set. The inquiry report recognises the deliberate trade-offs in clarity here, but concludes that to break the mental left-right

association with the engine position was probably not the optimum solution (Air Accidents Investigation Branch 1989).



Layout for aircraft involved in crash Layout revision suggested by inquiry

Fig. 4 Boeing 737 - 400 Sketch of Engine Information Sets (AAIB 1989)

2.1 Presentation of Information

The 737 involved in the East Midlands crash had flight deck engine information that lead to confusion under mental pressure. Placing the secondary information sets for both engines to the right of the primary set broke the implied rule set by all the other engine information, that the left engine had left hand controls and indicators (and vice versa). If one assumes that the optimum positioning of indicators is the one that requires the least mental processing then a simple symmetry about the aircraft centre line seems appropriate. The actual positions required a mental spatial transposition of one set of dials to the other side. The unfortunate false positive of mis-assigned reduced vibration is difficult to deal with, and is probably an issue of training aircrews to continuously check parameters. The readability of the indicators had been reduced by the substitution of electro-mechanical readouts with electronic readouts, but which simulated the old design. Possibly the redesign to electronic readouts should have taken the opportunity to use a rather different layout, possibly with linear indicators rather than rotary ones.

3 Case Study - Strasbourg Air Crash

On 20 Jan 92 an A320 Airbus of Air Inter crashed into a hillside shortly before it was due to land at Strasbourg. The approach was on instruments due to the cloudy conditions, and there was no distress call. Afterwards it was realised that the aircraft had been descending

far too steeply. This event fits well into the pattern of controlled flight into terrain (cfit) accidents where the aircraft struck the ground without anyone being aware of a problem.

The inquiry report (Aviation Safety Network 1992), (Bureau d'Enquêtes et d'Analyses 1992) concluded the fault lay with the pilots who had mistaken the heading and vertical speed mode with the track and flight path angle mode of setting the auto pilot.

The A320 was introduced in 1987 with a flight deck equipped with electronic displays, where one display panel could be switched between multiple functions. This major change from the established practice of panels with dedicated displays is put forward as the reason for several aircraft incidents. In the Strasbourg accident, a descent into Strasbourg was initiated at 3,300 feet per minute (fpm) when it would have been more usual to have set a glide slope of 3.3 degrees. The aircraft hit a mountain. One or other parameter was selected by means of a toggle switch and the adjacent display read the figure.

As the display was calibrated either as kilo fpm or degrees, the figures shown would have been virtually identical in both cases (‘3.3’) and the only differential would have been the position of the switch. In hindsight, the confusion seems understandable.

Air Inter had decided not to fit ground proximity warning radar to this aircraft, depriving it of a backup system that might have given a warning in sufficient time to change course.

As a result of this accident Airbus made some design improvements to the displays giving the digital vertical speed mode read-out 4 digits and the flight path angle read-out just 2 digits.

3.1 Presentation of Information

The Strasbourg accident appears to be a lack of risk analysis in design of the auto pilot controls. The input required was either the vertical speed or the flight path angle; it would have been invalid to enter both and therefore one indicator might have seemed appropriate. In this case the units of the display became just as important as the actual numerical value and needed at least equal visual prominence. The fact that the display was misinterpreted implies this had not happened. The improvements made by Airbus of differentiating the number of digits displayed seems not really to solve the problem - prominent display of the units of measurement might appear to be a better solution.

4 Case Study - SOHO Spacecraft

The SOlar Heliospheric Observatory (SOHO) was launched in December 1995 as a joint ESA / NASA mission to explore the relationship between the Sun and Earth. It carries twelve

instruments to measure such parameters as the composition of the solar wind, the temperature and structures of the solar corona, and the magnitude and frequency of solar oscillations. It was - and still is in 2006 - controlled from the NASA Goddard Space Flight Center. Experimenters work from the same facility.

SOHO was very nearly lost in June 1998 when a combination of events centred on the Flight Operations Team (FOT) caused them to send a series of commands which caused the spacecraft to lose sun alignment and communication with the ground. The problem was not triggered by any failure of the spacecraft systems. It took about three months to recover the craft and further time to return the damaged but serviceable mission to operation.

The causes included poor tracking of modifications to command scripts, an over-ambitious workload, a new operations procedure and a bypass of standard review procedures for critical activities.

The operations procedures such as momentum management, gyro calibration and science instrument calibration had previously been carried out as separate operations each during one twelve hour period. The changes shortly before the spacecraft accident compressed the procedures into one continuous sequence with a new script and no time left for contingencies. The occasion of the accident was the first time this script had been used.

Interestingly, an early deficiency report in 1994 highlighting the inability of the control centre to present critical data in an easily understandable manner had never been resolved (Section D.12 of the report (NASA/ESA 1998)).

The sequence of events started eighteen months prior to the accident with a decision to minimise the time that the three gyros were powered on, due to adverse reports about limited lifetime. Gyro A performed roll rate sensing during Emergency Sun Reacquisition (ESR), Gyro B sensed excessive roll rate, and Gyro C sensed roll attitude during thruster control modes. ESR was a spacecraft autonomous mode designed to ensure sun pointing was maintained. Three months later in March 1997 modified procedures were introduced, but no-one outside the FOT was made aware of them. The new procedures were used for regular momentum management in April and September 1997 with no problems.

The spacecraft incorporated a 48 hour minimum safe mode design to allow autonomous operation in the event of an ESR. There was an ESR in March 1998, three months before the accident, where the FOT intervened immediately rather than waiting. A shortcut in the recovery sequence promptly lead to another ESR from which recovery was made without long term impact. It lead to a recommendation for a comprehensive review of the software which did not happen. By this time significant experience had been gained controlling SOHO

and the team may have developed a false sense of confidence, as described in Section B.4 of the report (NASA/ESA 1998).

A modified timeline using the new compressed procedure was planned for June. NASA could not independently confirm the accuracy of the procedure on the simulator before the timeline was started.

After a routine gyro calibration, gyro A was powered off to conserve life, but the Central On-Board Software (COBS) function which activated it in case of an ESR was unintentionally left disabled.

A planned momentum management action took place, but afterwards gyro B was left in a mode with 20 times higher gain than required. This soon triggered an ESR because the roll rate appeared much greater than was actually true. Gyro A, at the time unpowered with the auto-power action disabled, was switched by the spacecraft into roll rate sensing as programmed by the ESR routine on the spacecraft. The incorrect setting of gyro B was noticed and corrected, but the disabled and de-spun status of gyro A was missed.

The spacecraft flight control system, ignoring the de-spun status of gyro A, integrated the gyro drift rate bias and computed a non-existent roll attitude error after fifteen minutes. Roll thrusters were fired to correct this non-existent error.

After another minute gyro B detected the anomalous roll rate and triggered another ESR. The FOT compared gyros A and B and decided that gyro B had to be faulty and deactivated it. This removed the fault detection capability from the spacecraft. The spacecraft was still relying on gyro A (deactivated) for inertial reference and soon fired the thrusters again, increasing the spin rate more. This was not sensed as gyro B was de-spun.

Eventually the pitch and yaw errors tripped another sensor set at five degrees sun-pointing error, triggering another ESR. With little control left, the spacecraft lost sun position and the telemetry was lost.

After the loss, another anomaly was discovered; three of the four bus discharge regulators had been disconnected from the bus several months before and no one had recognised the change in configuration. It would have limited battery discharge current when the spacecraft needed it for control. There is no evidence that this contributed to the loss of attitude, but may have had some influence on the speed with which telemetry was lost.

The inquiry board recommended a number of operational changes involving the teams and procedures, which can be read in the full report (NASA/ESA 1998). No details were given of the presentation of information to the FOT. The only comment was that a new Integrated Mission Operations Center (IMOC) with better visibility of telemetry frames was being used

up to the point where an ESR was generated due to the incorrect gain of gyro B. The team then reverted back to the old Transportable Payload Operations Control Center (TPOCC) due to a software error in the new system.

The review board was unequivocal: "...At any time over the five hour emergency situation, the verification of the spinning status of Gyro A would have precluded the mishap".

4.1 Presentation of Information

The detailed event timeline and failure event tree are available in the NASA/ESA Final Report, but a summary (Fig. 5) from the event tree is used here to highlight the HCI issues.

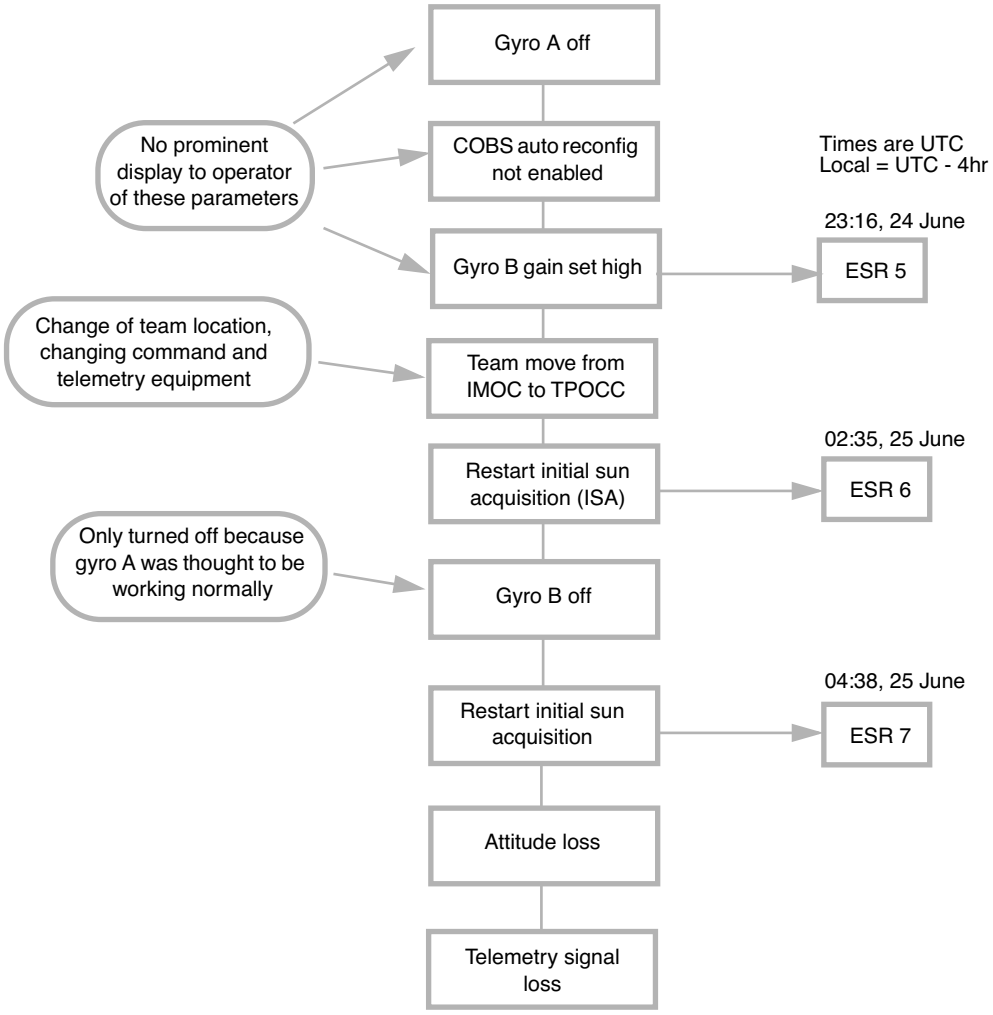


Fig. 5 SOHO Fault Sequence

If the facts that gyro A was off or the COBS was disabled had been visible to the operators it is unlikely the chain of events would have started. Leaving gyro B gain set high certainly caused an increased workload in trying to fix the developing SOHO crisis. At this point a software error with the IMOC caused the team to move back to the older TPOCC. This is not

brought out in the original report as significant, but it must be described as bad timing at least to happen in the middle of a set of abnormal events. It seems possible that the IMOC had not been thoroughly tested. Soon after this, gyro B was turned off but it is unlikely this would have happened if the status of gyro A had readily been apparent. Turning it off left the spacecraft with no inertial reference, sealing its fate.

4.2 Other factors

The failure to clear the 1994 deficiency report on the display of critical data was a related management function that could have prevented the accident.

5 Case Study - Presidential Election 2000

The United States presidential election in November 2000 was notable for a virtual dead heat between Al Gore and George Bush. The Electoral College result eventually rested on the vote in Florida where there were several close county ballots. The vote in Palm Beach county went to several re-counts, amongst complaints of a misleading election form, and the final count gave Bush gave the Florida votes needed for the presidency. A different result might have had very different consequences for American government policy.

Different counties were free to choose their voting machinery and Palm Beach used a voter operated machine to punch a hole in a form to align with the chosen candidate's name. The punched forms were then read by machine. The possible hole positions were pre-defined in one column. The county had too many candidates to fit in one column without either using a small font size, splitting the text up at the punched hole position or using a second sheet. The electoral officer chose to use two columns with the text alternately offset left and right about the centre column of holes (Fig. 6).

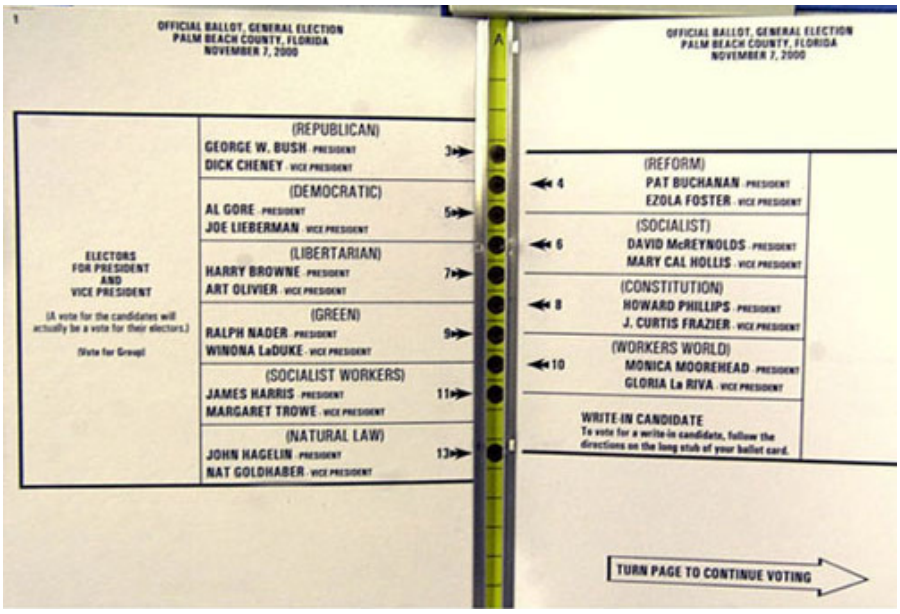


Fig. 6 Palm Beach Ballot Paper (Scott Fisher, Florida Sun Sentinel)

On the day of the election there were many stories of people leaving the voting booth not sure that they had voted as they intended, and some calls for the election to be re-run. The results showed a skew from the expected situation and an unusually large number of papers were declared invalid with two or more punched holes (known as over votes).

Following is a brief summary of the results. Only the first three candidates on the ballot paper are included as relevant to this discussion, although there were ten candidates altogether (Orszag 2000).

Gore (Democratic)	268,945
Bush (Republican)	152,846
Buchanan (Reform)	3,407 (0.74% of total Palm Beach vote)
Reform Party members in state	337 (0.006% of total state vote)
Overvotes (more than one hole)	19,120
Total votes	451,406
Registered voters	656,694

For the Florida vote overall (Federal Election Commission 2006):

Bush	2,912,790
Gore	2,912,253
Buchanan	17,484 (0.29% of total Florida vote)
Total	5,963,110
Bush majority:	537 (0.009% of Florida total vote)

The state of Florida is normally heavily Democratic; this result tipped it into being Republican.

One reason for choosing this example is that the number of sample results available - seen here as the number of votes cast - is high enough to generate reasonable statistics.

The electoral officer was quoted as choosing the two column layout to allow use of a large font for the benefit of the relatively high proportion of Palm County's population that was elderly with weak eyesight. Another county chose to go to two pages and found an even higher incidence of over voting than Palm County.

5.1 Presentation of Information

The situation of the 2000 presidential election seems to be one of unexpected changes from accepted practice. In this case a significant number of voters were expecting the typical single column layout that they had become used to. Although the form takes only a little study to see what was intended, the pressure of unfamiliar surroundings in the voting booth coupled to a form that looked similar to what was expected seems to have lead to many more votes for Buchanan than expected. Even Buchanan doubted their validity. Many voters appeared to look down the left hand column, see that Gore was second in the list and punch the second button down. By doing this they voted for Buchanan. The two halves of the ballot paper either side of the holes to be punched at first glance represent two pages of a book. A reader (assuming a western alphabet) would start at the top of the left hand page, read to the bottom and then start the right hand page. A desire to vote for Gore would be noted as second in the list, but would require punching the third hole. The much higher than typical proportion of spoiled votes that were overvotes seems to imply that a number of voters realised their error and pushed another button. To be sure of this it is necessary to check that this other button was the third button down; this detail is not available in the public literature.

If the argument for the Buchanan and his Reform Party vote has validity, then one would expect to see the statistics for Palm County with the double column form to be skewed when compared to the other counties with single column forms. Wand (Wand 2001) and others

showed (Fig. 7) this was so by plotting the proportion of votes for Buchanan against the total number of presidential votes cast for each county. Absentee votes were cast on a different layout of ballot paper, allowing a test to be made between this layout and the election day layout of Fig. 7 Palm Beach has the third highest number of votes cast allowing reasonable control of statistical errors and the graph shows the proportion for Buchanan is significantly higher than the trend.

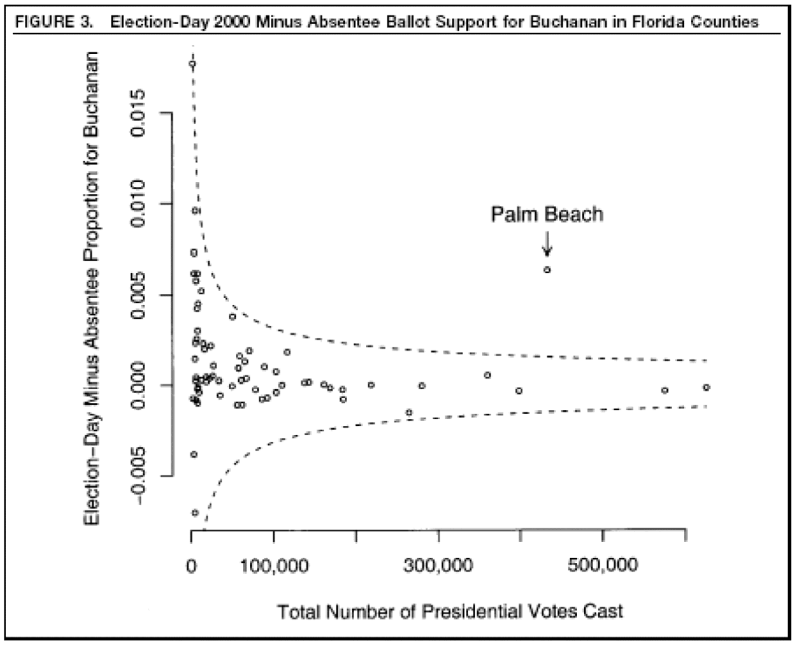


Fig. 7 Florida Voting Distribution Showing Palm Beach Anomaly (Wand 2001)

Wand calculates that if election day voters had cast votes in the same proportion that absentee voters did Buchanan would have received 854 election day votes. He actually received 3310 election day votes, suggesting that the design of the ballot paper led to 2,456 accidental votes for Buchanan. If the race between Gore and Bush had not been so close this result would have been unimportant on a federal scale. In November 2000 the whole system was so finely balanced that the result changed the federal outcome.

5.2 Alternative Ballot Paper

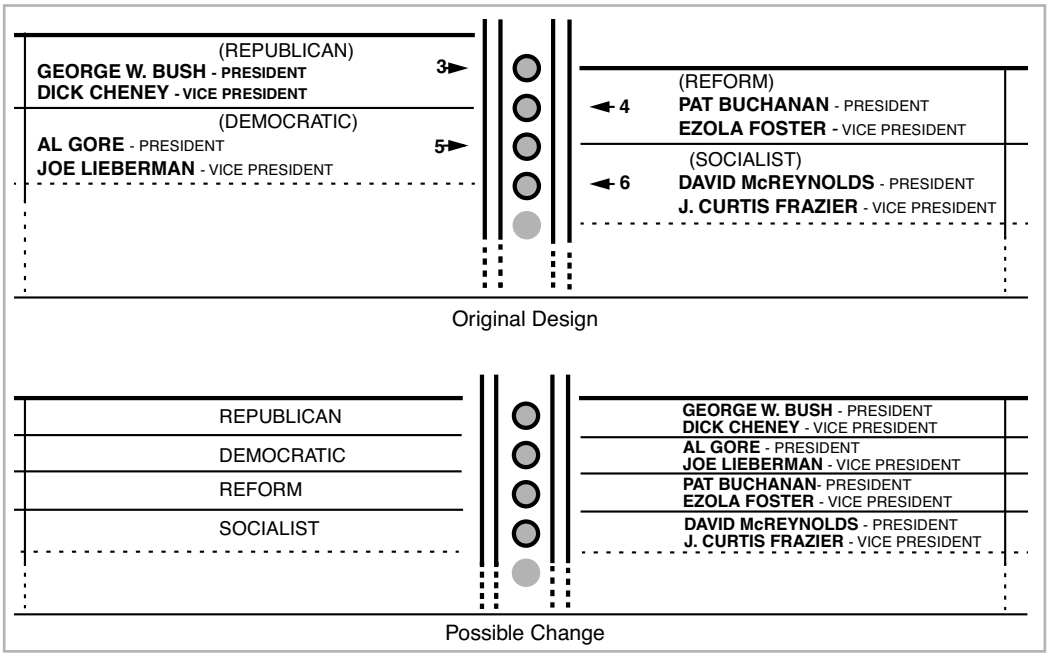


Fig. 8 Possible Option for Palm Beach Ballot

The ballot paper extracts in Fig. 8 were drawn as an example of what could have been done. The top illustration is drawn to look like the original paper, with similar font sizes and styling, but just showing the first four candidates for the sake of brevity. The lower takes the same hole punch array and restructures the information around it. The lower version degrades the readability a little by needing to use smaller font sizes and reduced line spacing to make the characters fit, and it is debatable on which side the candidates' names should be. Even so, it is reasonable to suggest that it is less likely to lead to voter confusion than the original. The partitioning between names and between parties leads the reader to one unambiguous hole choice, rather than the uncertainty of the original. The arrowheads and their numbers are removed also as being unnecessary information for the voter - they cause visual clutter.

The real problem here is the design of the voting machine. The hole punch positions are too close together to allow an adequately spaced listing of two candidates per hole position.

6 Contributory Factors Extracted from Case Studies

The case studies discussed here were analysed for any specific factors that could be extracted and used as a basis for avoiding pitfalls in new designs.

6.1 False familiarity with a system

When operating a complex system one builds a mental picture of how that system functions and how it is interconnected. That picture may be completely abstract or a model based on detailed knowledge of particular sub-systems. It is probably rare for operators to be familiar with more than a limited subset of the whole system, depending on the level of complexity. Reliance is put upon manuals and good feedback from the control panel. There is a hazard here that it is easy for familiarity with day to day operation to become an assumption that one knows most of what is necessary to drive that system fully - as expressed in the proverb “a little knowledge is a dangerous thing”.

This hazard probably applies to the SOHO accident. The inquiry report pointed out that the FOT had once previously recovered the spacecraft from an ESR without relying on the onboard autonomous systems, giving them a confidence in handling problems without waiting for the spacecraft to stabilise itself. When that autonomous recovery mechanism was compromised by disabling gyro A the system was then very vulnerable to further problems - in this case turning off gyro B as well.

6.2 Insufficient training

TMI would possibly have cooled without external intervention if the operators had not reduced drastically the HPI pump rate. The operators appeared not to realise the importance of the PORV and failed to check that it had closed, which lead to confusion about the system pressure and caused inappropriate action to be taken.

The Strasbourg crash may be analysed in terms of a secondary issue of insufficient training. The control that switched the autopilot descent rate from degrees to feet per minute should have been recognised as a weak design point by Airbus. It should either have been changed or at least the ambiguity pointed out repeatedly to pilots undergoing A320 training.

6.3 Unexpected changes from accepted practice

The Palm County ballot showed the effect of unexpected changes. Enough people were confused, and the electoral system was so finely balanced, that the federal outcome was probably changed. After a few moments study now the ballot paper may appear clear, but on

the actual day the potential confusion was changed into a real small shift of the vote of about 0.3% away from Gore.

6.4 Distractions

A significant amount of distraction must have been caused to the TMI operators by the large number of audible alarms sounding at once (“over one hundred”).

The SOHO team felt it necessary to move from one set of control equipment (IMOC) to the older set (TPOCC) part way through the ESR recovery. This could be classed as a distraction, caused by a software problem with the IMOC.

The East Midland inquiry recognised that the initial error in shutting down the wrong engine was compounded by the amount of work necessary on the flight deck to prepare for an emergency landing. If there had been more time, or the aircraft had carried a flight engineer, it is possible the error would have been noticed earlier with sufficient time for correction.

6.5 Wrong Assumptions

The 737 involved in the East Midlands accident suffered a very unfortunate false positive diagnosis about the developing problem, where shutting the wrong engine down coincided with the vibration from the faulty engine reducing.

A similar event occurred during the early stages of the TMI accident involving the pressuriser. This sealed vessel contained a volume of air designed to allow small variations in the volume of the primary coolant. The operators were trained never to let the pressure in the pressuriser rise to a hazardous level - “going solid” in industry terms. It appears that during the TMI crisis, the *level* in the pressuriser was thought to be a measure of the *pressure*. In retrospect, it seems curious why there was no prominent pressure gauge. Therefore the water flow was incorrectly cut back to try to reduce the pressure, allowing the reactor core to overheat.

6.6 Authority Within a Team

There has been significant evidence in air accidents particularly with crews brought up in Japan and Asia that the co-pilot can be reluctant to challenge the authority of the captain. Mistakes and sudden illness can be initially ignored under such a deferential relationship until the event is too serious to correct.

There is no evidence in any of the case studies here of this effect amongst the team directly in control, but there is also no evidence that any inquiry looked for it. There is evidence however in the case of the East Midlands accident that had any one of several passengers

or cabin crew queried the captain's comment over the intercom that he had shut down the starboard engine the situation might have been different. The port engine was visibly on fire for the cabin occupants, but this was never communicated to the flight deck presumably either due to a deference to authority or a wish not to interrupt in an emergency.

6.7 Time Pressure on Operator

Under pressure to complete tasks in a short period, the error rate for most people will increase. At the start of the TMI accident, ten major events occurred in the first four minutes, at which point the operators made the first manual and incorrect intervention to reduce the water injection rate.

The SOHO operations team were using a new method of working which involved a considerably compressed time scale.

The East Midlands accident involved a flight deck which was under pressure to reschedule the flight and land with a faulty aircraft.

The circumstances of the Palm Beach ballot may have exerted a certain amount of pressure on the voters not to wait too long at the voting machine for those situations where a queue had built up.

6.8 Backup Systems

Many systems are designed with backups so that in the event of a failure enough is left functional for basic control. The design of the user interface should take this into account and alert the operator appropriately. TMI had multiple systems - two emergency water feeds, a blast-proof containment building, a reactor scram mechanism and fortunately a block valve on the PORV - and all were required.

SOHO was safe as designed for forty eight hours autonomous control but the operator intervention undermined this.

The East Midland accident involved a plane capable of flying on one of its two engines, except that the backup had been powered down and there was insufficient power left to restart it.

The Strasbourg crash involved a plane that had for reasons of economy had ground proximity warning radar left unfitted, and so had no backup system to warn of an incorrect autopilot setting.

6.9 Presentation of Information

Implicit in the analysis here is an assumption that system information should be presented clearly. TMI certainly had a problem with the layout of controls and indicators, and included poor colour choices, displays out of sight of the operator, and inconsistent control operation.

The SOHO mission was missing the explicit display of the state of gyro A.

The East Midlands aircrash involved an aircraft with a potential confusion in engine management information which became real under pressure.

The aircraft in the Strasbourg air crash lacked the prominent display of the units to which the autopilot was set.

The Palm Beach ballot had a layout that confused a critically significant proportion of voters.

6.10 Specific System Design Issues

The analysis of the TMI control room shows that the correct use of system interlocks would probably have prevented the emergency. For example, the plant was run with the emergency feedwater supply disabled, which was a situation that was most unlikely ever to be required for correct operation. The plant should have been designed to shut down with the feedwater disabled.

7 Questions to Ask of Machine Controls and Displays

The following is a list of points drawn from these case studies that could be used to assess the HCI factors of a machine. The values in (brackets) refer to section 6 on “Contributory Factors Extracted from Case Studies”. These points are carried forward to Chapter Four to form part of a method for analysing system interfaces.

False familiarity with a system (6.1)

1. Does the design of the HCI attempt to establish a clear mental model of a system to the user?
2. Are abnormal situations catered for, and does the HCI adapt to these conditions?
3. Are precautions taken against operators becoming over confident in their knowledge of the system?

Insufficient training (6.2)

4. Is the level of training appropriate to the responsibility carried by the operator concerned?
5. Is the competence of the operator regularly assessed?

6. If a training simulator is used, is it adequately representative of the real system?
7. Are the risks of inadvertent operation of controls minimised?

Unexpected changes from accepted practice (6.3)

8. Is the design of the HCI checked against what would be established practice in the minds of the users?

Distractions (6.4)

9. Does the system minimise the risk of causing aural confusion (the inability to associate a meaning to a particular sound)?
10. Have precautions been taken to avoid aural or visual sensory overload?
11. Does the design of the system ensure that in a crisis an operator will not be distracted by external events yet will still able to communicate when necessary?
12. If audible effects are used, have precautions been taken to inhibit multiple alarms sounding at once?

Wrong assumptions (6.5)

13. Has the risk of wrong assumptions during operation been minimised by ensuring the state of unmonitored system nodes can be deduced accurately, possibly by using an algorithm to combine several parameters into one indicator?
14. Has the risk on system operation been considered for those parameters that are not available for display or control?

Authority within a team (6.6)

15. If the system needs more than one operator, is all the information available to all operators allowing decisions to be easily confirmed?
16. Are commands visible at all operator positions even though only one may have command authority?
17. Is there a good team culture?
18. Is use made of security levels and passwords so that a junior operator might only have access to common controls?

Time pressure on an operator (6.7)

19. Is sufficient time allowed for tasks to be done to the required standard?

Backup systems (6.8)

20. Has the question of redundancy been addressed in the system design and, if so, is enabling a backup mechanism a straightforward manual or automatic operation?

Presentation of information (6.9)

21. Are the most commonly used controls and displays placed in the most readily accessible position?
22. Are labels applied in a legible and consistent manner?
23. Are icons used appropriately and is their meaning clear?
24. If a display is used for more than one function, is the use to which it is currently put unambiguous?
25. Does the physical distribution of controls and displays follow to the best degree possible the physical layout of the system it refers to?
26. Have steps been taken to make the layout easy to understand and to reduce visual clutter?
27. Are audible effects used as hints for correct operation?
28. Is intelligent use made of colour as a secondary aid?
29. Are appropriate functions grouped together?
30. Is the direction of operation of controls consistent?
31. Are displays grouped with the control to which they refer?
32. Do displays and controls exhibit a good contrast against their background?
33. Have allowances been made for colour impaired operators?
34. Do any displays suffer from parallax viewing problems?
35. Does the layout of controls and displays aid a natural way of scanning the information (typically left to right, and top to bottom)?
36. Does the system clearly define a safe operating area inside which no mechanisms are over-stressed?

Specific system design issues (6.10)

37. Does it allow safe operating area limits to be deliberately exceeded in an attempt to bring a crisis under control?
38. Is appropriate use made of interlocks between controls to prevent inadvertent operation?

8 Conclusions

This chapter has shown some examples of situations where a lack of attention to detail in how humans interact with technology may have very unexpected consequences. There seems to be a common inability of the *designers* of the systems involved to be able to view

that system from the viewpoint of the *user*. Some examples are clearly worse than others; TMI clearly had many problems, whilst one small detail about the presentation of information cost the lives of all the passengers and crew of the Strasbourg-bound A320. In other respects the design of the flight deck of this particular aircraft is excellent. The SOHO example shows how a presumably competent flight operations team still managed to lose a spacecraft they knew how to drive - perhaps *because* they knew too well how to drive it and bypassed the autonomous safety systems. Clear presentation of the gyro A status is acknowledged as one change that would have prevented the accident. In the case of the East Midlands accident, after the initial - and incorrect - diagnosis of starboard engine failure the crew were too busy rescheduling their flight to look at engine displays again. What was required were displays that were unambiguous even at a glance rather than those that needed even slight consideration. The Palm Beach ballot is fascinating as it shows the effect of even marginal changes in understanding when the system itself is in a critical, finely balanced state. Normally few would have remarked upon the ballot paper, despite its limitations; in this particular case the outcome probably caused major changes to US government policy.

These examples from a few different fields show the importance of properly understanding HCI issues. They imply that it is necessary to design these concepts in at an early point in any development rather than as an afterthought. A set of deductions is shown which provides a framework around which to hang those design decisions, whilst a set of questions provide a mechanism with which to test the results. It seems reasonable to suggest that similar arguments shown as relevant to the systems here apply in the case of maximising the data return from science instruments. The same considerations of good communication between human and machine apply, and although poor design is unlikely to directly affect human life the data set returned may be barely usable - and that fact may not be discovered until after many weeks of fruitless analysis. There may be real instrument hazards too, such as accidentally pointing a fragile photon counter directly at the Sun, the chance of which in a well designed system should be virtually zero.

Chapter Four: Interface, Goals and User Analysis

So far we have looked at the technology available to implement a user interface, some of the standard methods of organising that technology into a functional whole, and we've extracted key interface issues from a set of case studies where unexpected things happened. This chapter takes the next step of exploring how to define the user interface. Sections 1 to 5 primarily consolidate the work so far and perform an analysis of the problem in the space science context. Section 6 describes the process referred to as persona analysis. Section 7 takes the process described in section 6 and applies it to a generic space science instrument. The result is a fully worked account of how to identify the users for whom an interface should be implemented, along with a description of the process to synthesise a full user interface.

1 Introduction

In this chapter we first attempt to define the guidelines for a user interface with a particular emphasis on space science, although it is likely to be valid for many other contexts as well. The approach comes partially from the case studies in Chapter Three, partially from a consideration of the specific space instrument context, and partially from a study of ergonomic criteria by Bastien and Scapia (Bastien 1993). These criteria were seen as a way of defining dimensions of usability in HCI by their authors, who tested them experimentally. The combined result is a set of 62 guidelines for use in analysing and synthesising user interfaces.

The interaction of a science user with an instrument is then looked at in terms of the science group type, the mode of operation, and the operating timescales. Following this, an initial analysis is made of the range of users and stakeholders over the lifetime of an instrument that are likely to have an active interest in it, and the longer-term goals that might be expected from such a set of people.

Then the concept of goal analysis using personas is introduced, based on work by Cooper and which forms the basis of a commercial consulting business in the USA (Cooper 2003). A persona can be regarded as an idealised user. A synopsis of his approach is included, and then the process is worked through in detail to analyse the users associated with a space science instrument. Cooper's method is useful as it describes a technique based on an end to end analysis of the problem, with extra information brought in by interviews with typical users of an instrument. Unlike many other methods of creating user interfaces, it does not rely upon user testing, which can be very time consuming, until the analytical work has been

completed. The results show that two personas, and therefore two interface designs, are needed to cover the range of users for a typical space instrument.

2 Categorisation

The case studies are useful for formulating a set of questions one should be asking about the usability and safe operation of a complex system, and are based upon a deliberately disparate set of examples to ensure a broad range of concepts are put forward. However, they need some structure and categorisation to make it easier to use and to allow focus in different areas. From the fact also that each of the case study events was a report of failure in one sense or another, it is possible that concepts that just worked normally were thinly reported. It is a sad fact of life that people rarely write detailed reports about success stories. The Bastien and Scapia study is very useful here, as it is aimed at categorising the requirements of software packages for everyday use, rather than safety critical systems, and some additional concepts have been taken from it. The five categories of Interaction, Representation, Consistency, Error Management, and Operator Aspects are extracted by inspection of the two sets of studies. Finally a few points are included to recognise specific needs of space science instruments, recent HCI ideas, and people management. An important distinction here is that the subject matter is the control of a sophisticated, vulnerable machine rather than the operation of a word processor. Failure in comprehension of the former is likely to have more serious consequences than failure in comprehension of the latter. For each category a description and a set of key words is given to show the basis for listing them together.

The derivation of each group of guidelines is given in brackets. ‘CS’ represents the ‘Case Studies’ from Chapter Three, ‘B&S’ represents ‘Bastien and Scapia’ (Bastien 1993), ‘Ch2’ represents Chapter Two of this thesis, and the paragraph number follows. One guideline comes from Raskin (Raskin 2000). The terms “space science” and “management” are used to indicate respectively a term specific to space science, and a term controlled by project management. This latter term is outside the scope of the discussion here.

2.1 Interaction

Interaction is the term used here to describe the two-way flow of information between a machine and a user or users. Devices to achieve this may be the familiar screen, mouse and keyboard, but might also include others that use sound or tactile feedback.

Key words: accessibility, adaptability, layout, expandability

2.1.1 Accessibility

This term is used to indicate items that are easy to reach or understand.

- Place most used controls in the most accessible positions. (CS 7.21)
- Group controls and associated displays together. (CS 7.31)
- Group items by function or sub-system. (CS 7.29, B&S 1.2.1)
- Allow command authority for one operator whilst allowing confirmation by all. (CS 7.15)
- Make it a trivial operation to discover operation and limits of controls. (CS 7.22)
- Suggest help for data entry based on immediate context. (B&S 1.1)
- Interface devices should be appropriate to the tasks. (Ch2: 1.3)

2.1.2 Adaptability

Adaptability is defined as the ability to change or be changed to fit new circumstances.

- Ensure the needs of all users are catered for. (B&S 4.2)
- Allow a user to customise their view of the system, whilst minimising the risk of this introducing operational hazards. (B&S 4.1)
- Allow for different levels of experience from different users. Allow the user to evolve their use of the interface as they become more familiar with it, including back tracking in the case where that familiarity has been lost through a period of disuse. (B&S 4.2)
- Allow parameters to be combined algorithmically. (CS 7.13)
- Allow resources for extra operators. (CS 7.15)

2.1.3 Control

The system should always feel under the control of the user, rather than the other way round.

- Ensure the user is always in control of the system. This requirement should include initiating, cancelling, undoing, pausing and continuing actions. (B&S 3.2)
- Anticipate all possible user actions and deal with them in a controlled fashion. (B&S 3.2)
- Keep system logs and make the information in them readily accessible. (space science)

2.2 Representation

Representation is the concept of something standing in for a tangible object, and is applied here to the indicators and controls used to represent the instrument. A control might be a joystick on a panel or an image of a slider on a screen. A display could be a mechanical meter with a real moving pointer or an electronic moving image of the same thing. Good

choices for component graphics are necessary, with use of contrast, colour and size to maximise legibility. Correctly chosen parameters should be able to ensure that for example even operator colour blindness is not a serious issue.

Key words: compatibility, legibility, conciseness, immediate feedback, mental model

- Take cultural conventions into account when displaying information, such as date formats and left to right and top to bottom reading direction. (CS 7.35, B&S 8)
- Controls and indicators should be self explanatory. Use legible and consistent labels or icons as appropriate. Choose fonts and use upper or lower case carefully. Choose abbreviations to convey the clearest meaning. (CS 7.22, B&S 8)
- Minimise visual clutter. (CS 7.26)
- Ensure an adequate information density per page to avoid the user needing to access an unnecessary number of display pages.(CS 7.26, B&S 2.1.2)
- Ensure each key entry provides immediate feedback. (B&S 1.3)
- Allow the effects of new control settings to be seen immediately. (B&S 1.3)
- In cases of long transmission times, give both immediate feedback of command sent and then eventual feedback of the instrument response. Indicate the system is waiting for a response. (B&S 1.3)
- Avoid visual and aural overload. (CS 7.10)
- Use the design of the HCI to create a clear mental model for the user. (CS 7.1)
- Allow parameters to be presented graphically. (Ch2: 1.6)
- Ensure there are no problems with parallax in viewing the displays or controls. (CS 7.34)
- Make audible effects available as hints for correct and incorrect system operation. (CS 7.27)
- Ensure multiple audible alarms if used are individually distinguishable. (CS 7.9, 7.12)
- Allow the physical layout of controls and displays to follow the physical layout of the system if possible. (CS 7.25)
- Ensure all parameters necessary for correct system operation are available for display or control as appropriate. (CS 7.14)
- Use colour appropriately (Ch2: 1.6.4)
- Use adequate size controls to make targeting with a pointer easy. (Ch2: 1.6.1)

2.3 Consistency

Consistency is used here to mean adherence to a set of ideas.

Key words: clarity, format, standards, accepted practice

- Avoid ambiguity in controls and displays, such as if used for more than one function. (CS 7.24, B&S 3)
- Use a consistent style to group items, and ensure the layout of controls and displays is consistent. (B&S 6)
- Ensure the operation of controls is consistent, such as in the direction of movement. (CS 7.30)
- Use accepted practice if appropriate in the design of the HCI. (CS 7.8, B&S 8)
- Use published standards for colour and contrast choice. (CS 7.28, 7.32, 7.33)

2.4 Error Management

Error management refers to the handling of operator and system errors, ensuring appropriate reporting and recovery actions. The scope of the term includes instrument safety.

Key words: error handling, redundancy, failure, hazards, interlocks

- Minimise the risk of inadvertent control operation. (CS 7.7)
- Protect the system against damage from a non-privileged user. (CS 7.18)
- Allow system operation in potentially hazardous modes only to fully qualified users, and ensure indication of this mode is unambiguous. Protect against the system being left unattended in a hazardous mode. (space science)
- Ensure the system safe operating area is defined clearly and abnormal operation is signalled clearly. (CS 7.2, 7.36)
- Allow safe operating area limits deliberately to be exceeded in an attempt to bring a hazardous situation under control. (CS 7.37)
- Use interlocks between controls to minimise risk. (CS 7.38)
- Check operational parameters against an internal list as they are entered. (space science)
- The interface should detect data entry errors and, where possible, suggest the correct entry. (B&S 2.1.2, 5.1)
- Avoid the use of error dialogues requiring operator acknowledgement except in instrument safety situations.(Raskin 2000) p84
- Minimise the actions required to correct an error. (B&S 5.3)
- Implement system redundancy if possible. (CS 7.20)

- Take precautions to prevent the operator becoming over confident in their mental model of the system. (CS 7.3)

2.5 Operator Aspects

The specific example of control of high value machines, as considered here, brings in extra considerations of machine safety and operator training. For remote network full access applications (sometimes referred to as eScience), autonomous instrument protection will be the norm and detailed operator training may only be appropriate for development and maintenance functions. For other applications more general training at various standards may be needed.

Only the second two entries under section 2.5.2 are directly associated with HCI design. The remainder are very important in the management and operation of complete systems.

Key words: education, workload, environment, responsibility, team work

2.5.1 Education & Training

- Ensure the operator has an appropriate level of background education. (CS 7.4)
- Check the operator has sufficient opportunity to become acquainted with the system and that the operator has sufficient time to build their own usable mental model of what they are trying to achieve. (management, CS 7.1)
- Make sure the level of operator training is appropriate to the responsibility carried and that any simulator is adequately representative of the real system. (CS 7.4, 7.6)
- Regularly assess the competence of the operator if work involving instrument safety is concerned. (CS 7.5)

2.5.2 Workload

- Ensure operator workload is reasonable for their ability, so as to neither overwork or allow boredom. Prevent operator workload increasing over time to the point where it becomes excessive and ensure work breaks are a natural part of the day. (CS 7.19)
- Create plans to avoid operator distraction in a crisis yet still allow communication. (CS 7.11)
- Ensure the computer provides filled in data entry fields where the value is already known or can be computed, to reduce the workload on the user and the risk of error. (B&S 2.1.2)
- Adopt access paths to page displays that are broad and shallow rather than narrow and deep, in order to reduce the workload on the user's memory. (B&S 2.1.2)

2.5.3 Environment

- Ensure physical environmental parameters in the room such as temperature, humidity, draughts, noise and frequent air changes are all appropriate. Check that immediate work parameters such as worktop height, suitable seating, lighting, display, keyboard and graphical device are all appropriate. (management)
- Allow the operator to have control over their environment. (management)

2.5.4 Teamwork

- Create a good team culture. (management)
- Allow operators to have a means of solving disputes and disagreements. (management)
- Encourage operational transparency by allowing all operators to see full details of items such as commands even if only one has command authority. (CS 7.16)

These ideas are summarised in Fig. 1.

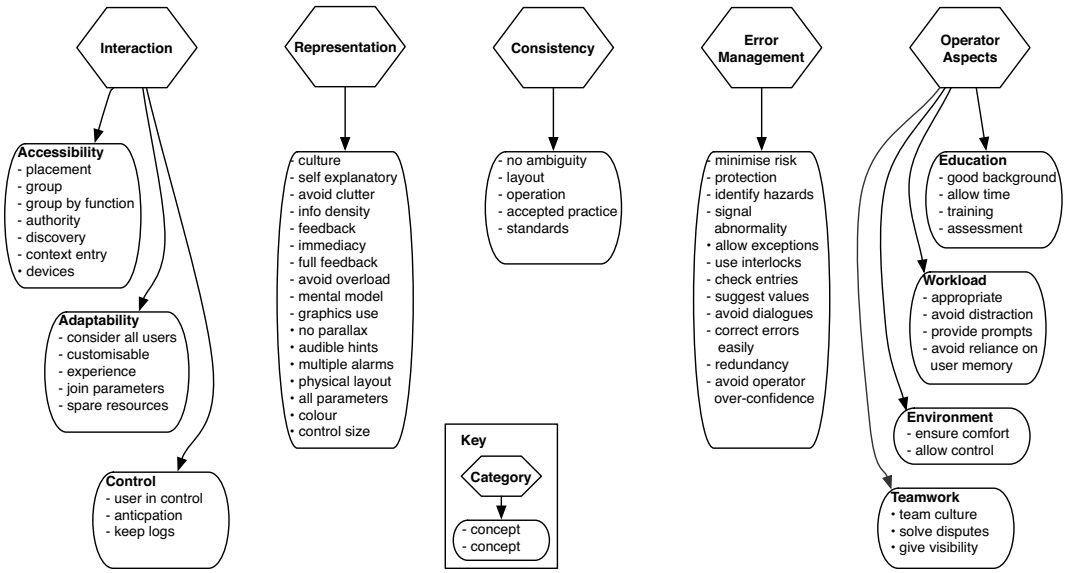


Fig. 1 Categories and Guidelines

3 Science Access Type

The previous section discussed general interface categories, with a bias towards space science. This next section looks at the factors that are specific to the space science context.

Interaction of a science user with a generic space science instrument can be categorised by the type of user group that they implicitly belong to, by the directness of operation of the instrument, and by the timescales involved. Those terms are described here.

3.1 Science Group Type

The space science field tends to divide into instruments planned and used by a small principle investigator (PI) group, instruments available to a wider community, and looking into the future instruments that have a very wide access facilitated by fast and prolific networks. These classifications are explored below. The dominant variable distinguishing them is the size of the user group, showing that the group types are only markers on a continuum.

3.1.1 Limited PI-type Access

A PI (Principal Investigator) type instrument is built by a small group of people (maybe no more than ten) who intend to use the instrument themselves. They are most likely to have been closely involved in the design, build and test of the instrument and know implicitly how to obtain the best results. Training is only an issue for new team members and is probably carried out within the team. Interface limitations will be tolerated as most of the team will know how to work around them. This type of organisation is currently used less and less.

3.1.2 Common User Access

Many instruments invite bids from the general science community for observing time. Potential users may have had experience with the general category of instrument before, or may be complete novices; it is unlikely they will have had any development involvement or any opportunity to meet any of the design team. The need for a good interface is high but probably not appreciated as expert advice may be easily available. More subtle instrument modes may be either not recognised as available, or little used as they may be perceived as too difficult. With limited time per user on a given instrument there are real pressures to take results and write the paper that is already late for publication.

3.1.3 Networked Science Access

Networked science (sometimes called e-Science) access implies users on fast networks with goals that are dynamic and comparatively short term. Ease of access will tend to demand ease of understanding within a short time frame, and imply an interface that is quick

to learn, and quick to make use of the full capabilities. Consideration of ways of working using this approach in particular seems to form a rationale for some standardisation between different instruments. This way of working is implemented for a few ground-based instruments at the time of writing, but needs consideration to form a balanced approach for the future. It may be very applicable for data extraction from archives and the idea of 'virtual observatories' (Koratkar 2001), (Euro-VO 2006).

3.2 Operation

3.2.1 Direct Operation

Direct operation is the mode of working where the user directly controls the instrument, either in real time or by means of a stored program executed later without further checking. It is becoming less common as space missions become more complicated, and is becoming less common also for ground-based observatories.

3.2.2 Operator Control

Operator control is the more commonly found manner of working where only a very small number of technical experts are allowed direct access to the instrument, and all other accesses including observations are made by submitting a plan which must be verified first. This approach clearly gives more protection to the instrument against accidental damage but restricts immediacy of access. It gives the important advantage of ability to schedule observation requests to give the best possible use of the overall time available for observing, and can make a large difference to the utility of an instrument. The disadvantages show in the teaching of astronomy, where there is real concern that it is difficult for a student to develop a realistic mental model of an observatory if one has never been visited. Setting parameters on a display screen in relative comfort does not appear to develop the imagination that personally handling a real telescope on a cold mountainside does (Page 2005).

3.3 Timescales

3.3.1 Real Time

Real time contacts between instrument and user are defined here as those where the instrument reacts immediately to a command, and where the physical distance is small enough to cause a negligible transmission delay there and back. Delays of less than 15s might fit into this category.

3.3.2 Near Real Time

Near real time contacts takes the idea of real time contacts, and add the need to make special consideration for longer loop times to allow for radio propagation delays.

3.3.3 Non Real Time

Non-real time contacts are those where a series of commands are stored on board the instrument or platform and executed at a defined later time. The way of working is also referred to as ‘deferred commanding’, or ‘stored programme commanding’.

3.4 Access analysis

3.4.1 Science Group

From the discussion above, the PI user may benefit from a rigorously developed system interface, the common user will certainly benefit, whilst the networked-science user may find it nearly essential. Therefore it is reasonable to describe the first two users as subsets of the networked-science user; anything good enough for network-science will certainly be adequate for the other classes and the need to analyse separately does not exist. There is the likelihood of doing more work than the absolute minimum required, but the potential usability gains are likely to benefit all users.

Using set notation:

$$\text{PI user} \subset \text{common user}$$

and

$$\text{common user} \subset \text{network user}$$

giving

$$\text{PI user} \cup \text{common user} \cup \text{network user} = \text{network user}$$

3.4.2 Operation

The case where an operator is required to enter the instrument commands is less demanding on the user than for the situation where the user is controlling the instrument directly. Therefore from the user viewpoint, operator control is a sub-set of direct operation, and there is no need to consider the situations separately. The design can simply assume direct operation, giving the expression:

$$\text{direct operation} \cup \text{operator control} = \text{operator control}$$

3.4.3 Timescales

For non real time communication, where the instrument works from a stored sequence, it is generally necessary to check out the intended sequence on a simulator beforehand. This may be an engineering hardware duplicate of the instrument, or a virtual instrument in software. For a near real time mode, it is generally desirable to be able to check out the integrity of any commands on a simulator before they are sent, if only to avoid wasted time when the instrument finally responds with an unwanted interpretation. Real time commands can be regarded as a sub-set of the near real time case, assuming the commands are checked on a simulator first. Therefore it seems reasonable to say that design should take place assuming non real time communication, that use of a simulator is essential, and thus there is no need to consider the timescales separately as far as user interfaces are concerned. This can be stated as:

real time **U** near real time **U** non real time = non real time

3.5 Conclusion

The terms of 'networked science access' for the science group, 'operator control' for operation, and 'non real time' for the timescales parameter fully enclose all the other terms. Therefore the parameters of science group, operation, and timescales do not result in the need for any specific provision for multiple instances of them in the interface to the user. Each parameter is required, but the analysis and implementation can be simplified as a single term in each case will cover all eventualities.

4 Users and Stakeholders

A range of users have interests in operating parts or all of a space instrument. A great deal of work takes place during initial conception, design, development and ground testing as well as the intended uses for astronomy. Each of these phases places specific requirements and differing priorities on the instrument user interface.

Those various categories of user are described here, roughly in chronological order of need of use from instrument inception to end of life. Inevitably the descriptions are somewhat simplistic, as there will be significant overlap between the nominal categories and one individual may well support several roles.

4.1 Instrument Scientists

The original idea for a particular instrument usually comes from a scientist researching in the associated field of interest. He would typically perform initial feasibility studies, carry out data

simulation, find collaboration partners, secure the funding, and be the anchor for the whole project.

4.2 Designers & Developers

Prior to building hardware, designers and developers may use software simulation techniques to do initial system design and verification of the concept. Parts of sub-systems will be prototyped, the systems built, tested at their individual unit level, integrated into one system and tested again. Depending on the programme, various system prototypes and engineering models may be built along with the flight model.

4.3 System Testers

Responsible for ensuring full end to end operation and calibration of the instrument. They may frequently be the same team that designed the instrument.

4.4 Trainers

Responsible for the training of astronomers and operators to use the instrument.

4.5 Observing Scientists

For observing, access for scientists is normally limited to non-hazardous operations only.

4.6 Operators

Most instruments have a small team of operators to ensure the interpretation of scientists' work with a planning tool into safe use of the instrument. They are typically also responsible for the implementation of regular maintenance tasks.

4.7 Developers (post commissioning)

Frequently problems need to be rectified and instrument capabilities extended after launch. This implies highly skilled people, but for certain disciplines only. They may not, for example, be particularly aware of safety issues and may require other support as a result.

4.8 General Science Community

The science community are people with an interest in the detailed results of the instrument, even though they may not have used the actual instrument itself. The science results are likely to be distributed far from the original people involved in the instrument design.

4.9 Teachers and General Public

The results of science missions are increasingly likely to be used in schools and for further education. The planning of observations for ground based telescopes by schools is already happening (in early 2006) and it would be reasonable to expect it for orbital instruments in the future.

4.10 Funding Bodies

Many people at various tiers of national organisations may be involved in committing significant funds for a new instrument. Whilst they may not be involved directly with using the instrument, their continuing support over the mission lifetime is critical.

Fig. 2 summarises this section on users and stakeholders:

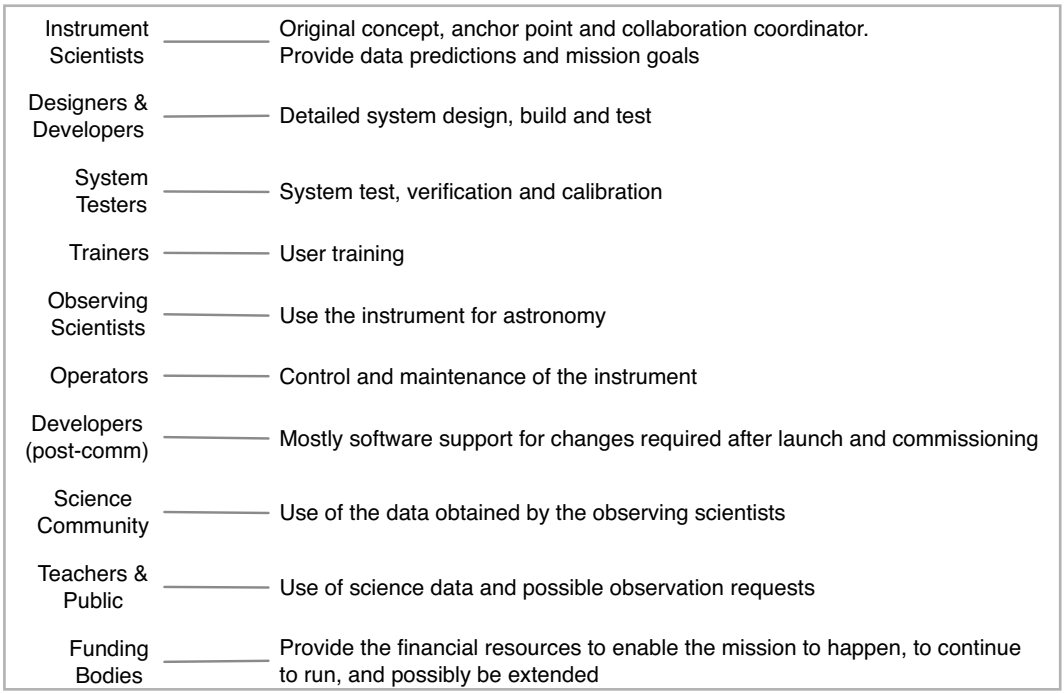


Fig. 2 Users and Stakeholders

5 Goal Analysis

As used in this context, a goal is an aim or purpose to be achieved, whilst a task is a job to be done. For example, a goal might be to image the corona of the sun, whilst one of the constituent tasks to achieve that goal might be to scan a mechanism from side to side. A goal is effectively a collection of related tasks. From this usage a goal is a longer term objective, more abstract than the task which is a shorter term well-defined piece of work. For another example, a goal might be to travel to work, whilst the first task might be to find the train ticket. An activity might be a goal or a task depending on the viewpoint of the observer; to travel to work could be viewed as one small task of the goal of making a best-in-class telescope.

An initial set of goals was derived by inspection of the roles carried out by the list of users defined above. These were then broken down further with the aim of identifying areas of work that would require some interface with the instrument concerned. The concept of goal analysis is taken further with the introduction of personas in section 6, where the concept of personal goals is also introduced.

5.1 Design and development of the system

- System modelling and breakdown into manageable sub-systems.
- Construction of fundamental budget plans, such as mass, power, thermal, radiation and data flow.
- Detailed sub-system development and initial testing.

5.2 Verification and calibration of the system

- Ensure correct operation and accurate calibration of the science data sub-systems such as the optical elements, detector and pointing mechanism.
- Ensure correct operation and calibration of all the support sub-systems such as power, thermal control, and telemetry.
- Characterise the instrument to understand the size of the system operational margins outside the normal points of operation.

5.3 Training of end users

- Ensure that scientists and operators planning to use the instrument know how to get the best out of it and know how to avoid situations that are hazardous.

5.4 Perform observations

- Observe various astronomical objects and make detailed measurements of various parameters over a period of time.
- Allow for wide accessibility under e-Science schemes.

5.5 Carry out maintenance

- Carry out regular, pre-emptive maintenance to enable continued operation.
- Ensure re-calibration of the instrument at suitable intervals to prevent loss of accuracy.
- Fix problems as they occur.
- Extend the capabilities of the instrument with, for example, improved on-board data processing algorithms.

5.6 Provide publications and publicity

- Provide data handling, archiving and retrieval facilities to enable the efficient writing of scientific reports.
- Provide means of generating content for web pages for general public viewing.

5.7 Provide damage protection

- Provide protection at all stages of the instrument life against accidental damage.

The goal of ‘damage protection’ may require separate consideration, as it applies to all phases of the instrument life.

6 Goals Analysis Using Personas

6.1 Introduction

So far this work has looked at analysing the goals of a potential space instrument by simple inspection of the roles *perceived* to be taken by the groups of users that have been identified. There is no particular reason to believe that this analysis is inaccurate; however, it is a set of views taken by an external observer albeit with experience of working in the field, and taken without asking directly those user groups involved. What is needed is a way of gathering information from user groups in an unbiased yet structured manner, and this section details an approach described by Cooper based on the concept of user definition using hypothetical idealised users known as personas (Cooper 2003). In terms of research method (see Chapter One) it uses an ethnographic approach (the researcher is immersed in the environment), followed by a well defined procedure for dealing with the information obtained. This persona-based method is described here with comments from personal experience in applying it to the space science context, and includes a description of the process of taking it to user interface definition. The analysis structure is due to Cooper, with space science context additions due to the author.

The appeal of this approach is that it offers a well-defined methodology which does not rely upon user tests and arbitrary user exercises. Testing users with structured exercises is useful the first or even second time it is carried out, but it carries the risk that the subjects start to anticipate the motivation for the exercise and modify their behaviour accordingly, even if they do not realise it. New subjects are then necessary, which may be satisfactory for some fields of work, but the population of the space science world is not that large! It is also geographically scattered, implying some difficulty in assembling a suitable number of subjects in one place. The alternative of self administered tests over a network could work, but implies some loss of control of the test conditions and therefore reduced integrity of the results. Persuading geographically remote space scientists already up against several deadlines that they should do some user testing as well may have a variable success rate.

6.2 First Stage

The developer starts by making himself familiar with the particular field of work under discussion. A number of personas of different user types are then hypothesized as a first attempt, ready to be refined and changed as necessary. A persona is defined as a description of a hypothetical user in terms of likes, dislikes, typical activities, approach to work, and similar goals. In other words, it is a behavioural description of a user in terms of their life, experience, work and end goals. Most importantly, it is not work goals alone.

Different types of user are likely to have different ways of tackling tasks; for example, a software developer might not have much knowledge of pulsars whilst a scientist might not care much about language structures. However, both users are likely to take care about the fine detail of their work. This gives an example of three constraints, the behavioural one of which is common to both parties. The researcher should then make a first attempt to describe a number of relevant persona hypotheses to create the opportunity to bring out initial thoughts - and prejudices - for consideration. Cooper suggests that 100 to 150 words each is adequate.

6.3 Interview Subjects

The next step is to interview typical subjects in the user groups concerned. One person may actually fill more than one role. Ideally the interview should take place where the work is carried out, although this is not always practical. The concept is of a minimally invasive technique, where the interviewer says very little and the subject describes what their goals and tasks are. If possible, the subject should describe their work by referring at the same time to the computer or other machine they use for it. Two people are better at recording the information, with one asking the questions whilst the other writes. The author feels that the subject may find it easier and give fuller answers talking to just one person though, as some answers may reveal criticisms of the organisation that is their employer, and it is easier to build a sense of trust with just one interviewer. Asking to sound record the interview to make note taking easier met with such unease that all the interviews described here were done by the author alone and recorded with pen and paper. It is important to stress before starting that the information given would be treated in a discreet manner and made anonymous before publication. A single institute (MSSL) was used as the source of interviewees; use of more than one institute might reasonably be expected to emphasise different parameters.

Cooper suggests avoiding a fixed set of questions and to consider the most common activities, favourite and least liked aspects, work arounds, short cuts and subject motivation. Focus on the goals of the subject first and understand why they are performing a particular action. Avoid solutions or technology and concentrate on the problems. Narrative is good at bringing out normal and unusual use of the system. It is important to avoid leading questions which inadvertently suggest the answer the interviewer is expecting. It is also important to write the rough notes up more fully within a few hours, filling in gaps from memory, and particularly so in the case of a sole interviewer. If possible those notes should then be checked by the interviewee.

6.4 Analysis

Having gathered some interview results, the analysis can start. Fig. 3 summarises the method. Initially the original persona hypotheses are reviewed and a list made of the behavioural variables observed. A behavioural variable is a variable across one type of behaviour, such as attention to detail, or the need to have well defined work hours. Then repeat this exercise with the notes from the real subjects. Show the variables diagrammatically, and map the subjects to the diagrams ordered by rating. Identify significant behaviour patterns, looking for clusters across several variables. The variables need to be related for a cluster to be counted as valid.

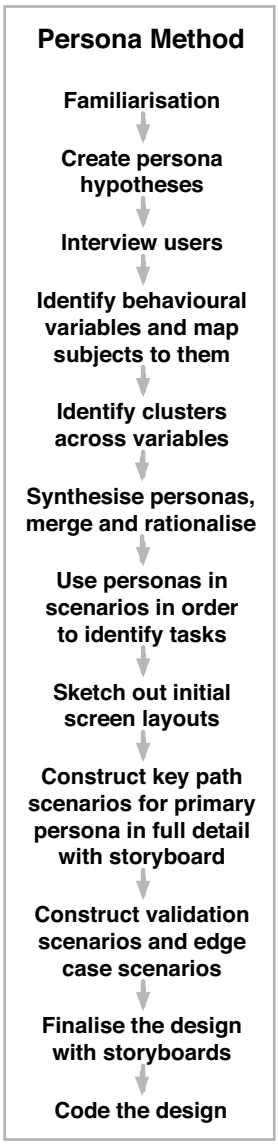


Fig. 3
Persona Method

The next stage is to synthesize characters and relevant goals, creating a persona for each significant behaviour pattern. The emphasis on this process is to keep it simple. Give each persona an appropriate and evocative name, as we are trying to build the idea of personas that are conceivably realistic. Identify their goals, which are most likely to be end goals, but also include life goals. End goals are defined as those of the project under discussion, and life goals are the long term personal goals of the people involved. At this point check for completeness and distinctiveness, and check through the subject notes again. Take the opportunity to remove or merge redundant personas if one is close to another in terms of behaviour. Then fully develop the persona with a narrative of up to a few hundred words, and possibly include a photo or sketch that captures demographics, environment and their general attitude. Cooper is particularly keen, based on experience, on the use of an image to consolidate the persona (Cooper 2003). This seems appropriate, as what is happening here is that the opportunity is being created to form a series of mental models (see Chapter Two) of the system users in the mind of the system designers.

6.5 Persona Types

Designation of the different persona types into well defined categories is the next task. Cooper suggests six types:

6.5.1 Primary

The primary persona has needs and goals not satisfied by any other persona in the analysis. This persona also minimally satisfies all the others. It is possible that in any given context there is more than one primary persona, shown by clear but divergent user needs, requiring more than one design of interface. Space science seems a possible candidate for this approach as superficially the needs of developers and the needs of science end-users seem rather different. We must not follow this route without any robust supporting evidence, as if one design of interface is viable it has all the benefits of simplicity, minimal maintenance, and so on.

6.5.2 Secondary

A secondary persona has the goals of the primary but requires one or two more for completeness. These extra goals must not interfere with the primary.

6.5.3 Supplemental

A supplemental persona has differences from the primary, but is satisfied by the primary interface.

6.5.4 Customer

A customer persona has extra interests such as providing the funds for the work. They are unlikely ever to operate the instrument, and therefore any usability comments should only be considered in the viewpoint of a main user. They are generally grouped with the secondary personas.

6.5.5 Served

A served persona does not use the instrument but is affected in one way or another by it.

6.5.6 Negative

A negative persona is the type of user who should explicitly *not* be catered for. At the risk of leading to some difficult decisions, this group might typically include the programmers actually building the interface. Their needs, during the time that they are performing the actual programming work, are by definition *not* a major end use of the interface. This grouping highlights the strength of the persona method; the needs and requirements of the actual users can be analysed and isolated from the features requested by other people who come into contact with the evolving system. It ensures that the analysis of user needs and the design of the interface is performed separately from and prior to the actual implementation. A programmer may well have extremely valuable information to contribute

to the interface design; what is important is that the information is included at the *design* stage in conjunction with other design information and prioritised correctly. The *implementation* stage should take this information and use it to construct the interface but not change it without re-invoking the perspective of the design decisions. There is an issue here of ensuring and enforcing this way of working which may involve using extra resources.

6.6 Non-elastic users

It is vital that personas are treated as well defined and, once defined, treated as inflexible unless the design process is carried out again. The target should be to design for a specific persona and not to stretch their parameters so that an unsatisfactory interface design appears to fit. If more than one primary persona is needed to reconcile the situation, that is what has to happen.

6.7 Build a Solution

At this point the personas are complete and can be used to build a working interface. There is no definitive route for this and the following suggestion is again due to Cooper. This is carried out for each primary persona, which is normally a very small number and in the space science context is likely to be one or two. It may be appropriate to consider the secondary persona also.

Cooper's recommendation is to approach this by putting the persona in likely scenarios to identify the actual tasks necessary to reach the goals described earlier. Use scenarios with a narrowing focus of first a broad context, then key paths and then validation. A narrative technique is effective in exposing the necessary actions that form the scenario. Here is this method described in more detail:

6.7.1 Context Scenarios

Create problem and vision statements to identify what the goals are and how they might be tackled, maybe using a brainstorming method to maximise their visibility. Then identify the expectations of the personas, possibly using the concept of mental models. Next construct context scenarios of a typical series of interactions as they should ideally happen using the goals defined earlier, and then use these to identify the actual tasks - the objects, actions and contexts - that need to be carried out. The sequence of identification of expectations, construction of scenarios and identifying tasks should be iterated until it is stable.

6.7.2 Key Path Scenarios

The next requirement is to start the design of the actual user interface using the requirements of the primary persona. Decide the physical structure of the hardware (the form factor), the display device and the input tool. Conceivably there could be more than one of each of these. Space science has typically made use of conventional displays, keyboards and mice, but this is the point to decide whether personal digital assistants (pdas), touch screens or another technology is appropriate. Define various views, including the initial introductory one in particular. Then consider the functional display elements from the system tool box (Chapter Two Fig. 5), and decide whether to create any bespoke elements for the particular context. The standard system toolbox is likely to cover most scenarios, but one possible extra in the space science context is a graphing tool. Consider also any common interface items that may be available from other instruments, as discussed in Chapter Six.

The designer should then take this set of parts and decide how to use the available display space. Functional groupings and their hierarchy need defining, along with the use of containers to delineate those functions. The layout should probably follow the flow of the task to be performed, and should reinforce any mental models that have been identified. At this point the layout can be sketched, either on paper or electronically, but in a simple manner and with limited detail.

Having done this, the tasks that will be performed frequently, the key path scenarios, can be constructed with a story-board based walk-through in full detail.

6.7.3 Validation Scenarios

With the major framework of the key paths defined, we are now in a position of constructing the validation scenarios for the minor, less used, paths for the primary persona. There will be key path variants where infrequently the task flow splits from a main path, and invokes rarely used but *necessary* actions which cannot be omitted. Then there will be edge case scenarios, defined as those which are both *optional* and infrequent, and a decision to include them may depend on whether their presence might obscure the main objective. An example of the former might be the need to protect against specific hardware failure, where the correct action must be taken on the rare occasion a failure happens. An example of the latter might be the question of whether to provide a quick route to achieve a very infrequent activity, when a more time consuming route already exists. To provide the quick route might risk a more cluttered screen and hence risk operator confusion, and so the decision may be taken not to implement it.

6.8 Finalise the Design

We now have a detailed framework as a set of narratives for a design of the interface for the primary persona, ready for the concept to be developed fully. This can be done with a full set of story boards able to demonstrate the solution, either on paper or using software capable of fast prototyping. The choice may depend on the skill set of the people available for the work.

6.9 Skill Level

Particularly relevant to the space science context seems to be the user skill level to assume. Take the example of an observing scientist. They are likely to be highly competent but busy people who use a given instrument on one occasion, and then may not use it again for months or longer. They need reasonable control over the nuances of a particular instrument, but the memory of any specific detailed knowledge needed for a given action is likely to have faded somewhat when the next opportunity to use it comes along. One solution to this might be to provide simulators that allow potential users to practice their skills to minimise memory loss, but there would be little incentive in the real world to make time for this. Cooper has a very appropriate strategy for this of assuming three skill levels (Cooper 2003). First is the 'beginner' whose experience of a given interface is zero or effectively so. Then there is the majority of users grouped under the elegant term of 'perpetual intermediates'. These users have enough experience to know how to start, but like our hypothetical science user do not use it enough to remember all the fine detail. The third group are the 'experts', a typical member of which might be a systems programmer who is using the system all day for days on end, and probably designed most of it. Their need will be a means of communication with the instrument that minimises any overhead and time delay. Important to note here also is that it is likely there will be system programmers (repeating the particular user example) who will *not* be experts in all the areas covered and will need the occasional prompt.

Skill level could therefore be seen as another parameter attached to each class of user described earlier. This potentially complicates the issue by multiplying the number of users classes by three. In practice, as any item of technology is used, people move rapidly from the beginners category to one where they have some confidence, the perpetual intermediates. Most stay there, and only a few have the close and prolonged involvement where they might find an interface correctly designed for a range of intermediate skills is actually slowing them down. Therefore designing for 'perpetual intermediates' will cover the great majority of users, particularly in space science, and the edge case requirements of expert users will be covered in the 'validation scenario' process above. The edge case

requirement of beginners should aim to generate confidence in the interface fast enough that their demands are covered by the intermediates category.

6.10 Creating the Design

The design is now ready for implementing with an appropriate software tool. At this point the programmer must resist any temptation to re-design whilst coding, except by going back over the design cycle and applying the methodology to the proposed change. Following this route will ensure the design model is kept current. It might be realistic in a real design project to allow at least one cycle of design change and re-coding before enforcing configuration control on the results.

7 Detailed User Analysis

Having described an outline of the overall persona process in the previous section, this section tackles the detailed analysis of users for a space science instrument. The groups of users and stakeholders described in section 4 are taken as the persona hypotheses referred to in section 6.2. Behavioural variables and their measurement limits are proposed for each user group and then the descriptions sorted by behavioural variable. Then a number of people active in space science are interviewed and behavioural variables extracted from those results. The two lists are merged to eliminate duplicates and the data plotted along scales of behavioural variables. Results and deductions are extracted from those graphs and finally a set of personas are constructed. This gives the first and major stage in creating a full user interface optimised for a space science instrument. It does not attempt the next stage of that actual interface design as the work would be purely speculative for a hypothetical instrument and would not seem to contribute much to this thesis. The full procedure is described in outline in section 6 earlier in this chapter. Fig. 4 illustrates this process.

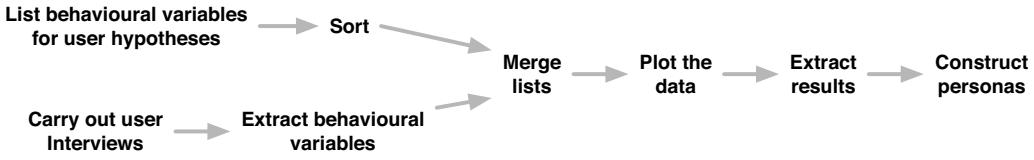


Fig. 4 Behavioural Analysis

In the past, it has been typical for a space science instrument to have been implemented with four groups of tools and their interfaces:

Science planning tool - This is used for operational mission planning. Typically it is specified late in the development programme and is also often constructed by the science analysis team.

Engineering control system - This is used for development, testing, commissioning and operational work. Often it is incrementally built as the programme progresses. It is the responsibility of the software engineering team.

Science quick look tool - This is used for near real time inspection of incoming science data to give a degree of confidence in the data quality. Frequently it is a software tool written by the science team.

Miscellaneous interface tools for development work - These are used for development and testing of instrument sub-systems during development. They are necessary because of the lack of maturity of other parts of the system. They are frequently implemented by just one person with minimal formal documentation, and a minimal command line interface.

This should be compared with the results at the end of this chapter, and also with the approach advocated in Chapter Eight.

7.1 Behavioural Variables for Persona Hypotheses

By simple inspection, we propose that likely behavioural variables for these persona hypotheses from section 4 are:

[The syntax used here is: *behavioural variable (one behaviour limit - other behaviour limit)*]

Instrument Scientists

finished instrument (best in class - minimum to meet specification)

be a team leader (inclusive - insular)

want accurate science data (fastidious - carefree)

require financial accuracy (precise - minimum necessary)

care about schedule progress (cautious - lax)

carry out public outreach (important - don't care)

provide collaboration (cooperative - recluse)

Designers & Developers

follow technical budget limits (careful - loose)

provide attention to detail (good - bad)

provide collaboration (cooperative - recluse)

be a team leader (insular - inclusive)

System Testers

take care of instrument (exceptional - adequate)

provide collaboration (cooperative - recluse)

be a team leader (inclusive - insular)

provide reporting & communication (thorough - minimal)

testing (basic test all features - thorough test selected features)

Trainers

offer use of teaching skills (good - bad)

provide attention to detail (good - bad)

Observing Scientists

want accurate science data (fastidious - carefree)

carry out paper writing (stimulating - uninteresting)

provide collaboration (cooperative - recluse)

carry out public outreach (important - don’t care)

Operators

take care of instrument (exceptional - adequate)

provide collaboration (cooperative - recluse)

be a team leader (inclusive - insular)

provide reporting & communication (thorough - minimal)

Developers (post commission)

take care of instrument (exceptional - adequate)

provide collaboration (cooperative - recluse)

be a team leader (inclusive - insular)

provide reporting & communication (thorough - minimal)

provide attention to detail (good - bad)

Science Community

carry out paper writing (stimulating - uninteresting)

provide collaboration (cooperative - recluse)

carry out public outreach (important - don’t care)

Teachers & Public

support astronomy funding (for - against)

Funding Bodies

finished instrument (best in class - minimum to meet specification)

require financial accuracy (precise - minimum necessary)

care about schedule progress (cautious - lax)

provide reporting & communication (thorough - minimal)

carry out public outreach (important - don’t care)

Then sort these persona hypotheses by behavioural variable to identify common areas, as in Table 1:

(The behavioural variable names below are shortened slightly from the previous section)

Behavioural Variable	Persona Hypotheses
accurate science data (fastidious - carefree)	instrument scientists, observing scientists,
attention to detail (good - bad)	trainers, designers & developers, developers (post commission)
care of instrument (exceptional - adequate)	system testers, operators, developers (post commission)
collaboration (cooperative - recluse)	observing scientists, science community, instrument scientists, designers & developers, system testers, operators, developers (post commission)
financial accuracy (precise - minimum necessary)	instrument scientists, funding bodies
finished instrument (best in class - minimum to meet specification)	instrument scientists, funding bodies
paper writing (stimulating - uninteresting)	observing scientists, science community
public outreach (important - don't care)	instrument scientists, observing scientists, science community, funding bodies
reporting & communication (thorough - minimal)	system testers, operators, developers (post commission), funding bodies
schedule progress (cautious - lax)	instrument scientists, funding bodies
support astronomy funding (for - against)	teachers & public
team leader (inclusive - insular)	instrument scientists, operators, system testers, developers (post commission), designers & developers
technical budget limits (careful - loose)	design & developers
testing (basic test all functions - thorough test selected features)	system testers
use of teaching skills (good - bad)	trainers

Table 1 Hypotheses sorted by Behavioural Variable

This work with initial personas is used to bring out the researcher's ideas and prejudices. It is then repeated with actual interviewees.

7.2 Interviews

Five interviews were carried out with people working directly with instrumentation in space science. Excerpts of the interviews are in Appendix B and only derived information is presented in this chapter. A model interview is included in Appendix B. The interviews were conducted under an agreement of anonymity and therefore no personal identities are divulged. Randomly assigned names of Jane, Nick, Kevin, Helen, and Bill have been used for easy reference. The subjects were asked to contribute for reasons based on their personal experience as part of the user groups above and therefore it is thought reasonable not to require a statistically balanced selection of people.

These groupings show how much the somewhat arbitrary user groups derived earlier tend to be performed by the same people in actual practice. This is a natural consequence of the small teams typically involved with a given instrument. No explicit candidate could be found to interview for the Trainer and Operator groups. Their activity is considered to be covered to a reasonable degree by the available groups. The Teaching & Public group is typically not a user of an instrument interface with current instruments and is therefore reasonable to omit for the present. An instrument specifically targeted at schools access, for example, would necessitate reworking of the analysis to include this group. The Funding Body person (by definition) will never be a user of the interface and is therefore safe to omit.

The groups and interviewee natural assignments are:-

Group	**Name**
Instrument Scientist:	Jane, Nick, Kevin
Designer & Developer:	Helen, Bill
System Tester:	Helen, Bill
Trainer:	n/a
Observing Scientist:	Jane, Nick, Kevin
Operator:	n/a
Developer (post comm):	Helen, Bill
Science Community:	Jane, Nick, Kevin
Teaching & Public	n/a
Funding Body:	n/a

This list essentially shows just two groups; there are those who deal with the science (Jane, Nick, Kevin), and those who look after the technology (Helen, Bill). This seems to be a common dichotomy; it is unusual for one person to be involved in both areas and it is a situation that will be referred to again in Chapter Seven on the possible use of simulators in instrumentation.

The interviews were carried out as described in section 6.3 and then analysed to look for trade-offs each person had to make whilst working, or goals that they aimed for. These parameters were then used as the behavioural variables shown in Table 2 below. To give traceability, the source is given as well. Care has been taken here to be as objective as possible and to avoid including the interviewer's viewpoints. This clearly would be easier with a second interviewer with whom to corroborate, although there might be a trade-off against openness as noted in section 6.3.

Ref	Behavioural Variable	Source
1	astronomy work (paper publishing - observing & research)	Jane, Kevin, Nick
2	attitude to management (management as authority - management as facility)	Helen
3	calibration (rely on others - do it personally)	Kevin, Nick
4	consider rewrite of legacy software (support - resist)	Bill
5	consideration of others (try to achieve consensus - take own view only)	Helen
6	consortium feedback to developer (vital - unconcerned)	Bill, Helen, Kevin
7	development programme (get job finished - add more features)	Bill, Kevin
8	display style (cluttered if necessary - clear layout with fewer items)	Bill
9	independent way of working (important - unconcerned)	Jane, Nick
10	interest in instrument design (involved - leave to others)	Jane, Nick
11	interested in work (do extra if needed - do minimum)	Bill, Helen, Jane, Kevin, Nick
12	isolate technical decisions from political influence (important - unconcerned)	Bill, Helen

Table 2 Behavioural Variables from Interviews

Ref	Behavioural Variable	Source
13	keep self-motivated (understand big picture - concentrate on fine detail)	Helen
14	modify specification (never change once started - accept late changes)	Bill, Kevin
15	obtain results (simple plan to guarantee some results - complex plan with risk of no results)	Kevin, Jane
16	operation of new instrument (take the time to learn - use existing methods)	Jane
17	organisational control to stop late changes (need it - unconcerned)	Bill, Kevin
18	personal involvement at instrument inception (strong wish - not concerned)	Bill, Helen, Kevin
19	resolution of data (better spatial, poorer temporal - better temporal, poorer spatial)	Jane
20	resources for training developers in new methods (vital - unconcerned)	Bill
21	teaching (vocation - distraction)	Jane
22	team working (collaboration - individualism)	Bill, Helen, Jane, Kevin
23	time to produce development tools (vital - unconcerned)	Bill, Helen
24	training (action - reading manual)	Helen
25	trust in others (full - none)	Helen
26	usability development (spend time on interface design - rely on printed manual)	Jane, Nick
27	use of detector (best performance - longest life)	Nick
28	why do it (be first to discover something new - just a job)	Jane, Nick

Table 2 Behavioural Variables from Interviews

7.3 Merge Results

Next inspect the two lists of variables, from the persona hypotheses and from the interviews, with the intention of forming a combined list. This method is at variance with Cooper's approach, who uses the hypotheses as a template for deciding if the interview results cover enough areas of behaviour. Merging the lists is used here as a means of including specific

information from the hypotheses, which would otherwise be lost. With a larger number of interviewees Cooper's approach would be the appropriate choice. In Table 3, the hypothesis list is written down again and compared against the interview list for close matches. The interview wording is taken each time there is a match. Unshaded boxes and bullet marks show the selected final wording, and the table includes the reference numbers from Table 4.

Final Ref	Hypothesis Behavioural Variable	Interview Behavioural Variable
3	accurate science data (fastidious - carefree)	• calibration (rely on others - do it personally)
33	• attention to detail (good - bad)	(none)
29	• care of instrument (exceptional - adequate)	(none)
22	collaboration (cooperative - recluse)	• team working (collaboration - individualism)
30	• financial accuracy (precise - minimum necessary)	(none)
31	• finished instrument (best in class - minimum to meet specification)	(none)
1	paper writing (stimulating - uninteresting)	• astronomy work (paper publishing - observing & research)
34	• public outreach (important - don't care)	(none)
35	• reporting & communication (thorough - minimal)	(none)
36	• schedule progress (cautious - lax)	(none)
37	• support astronomy funding (for - against)	(none)
22	team leader (inclusive - insular)	• team working (collaboration - individualism)
14	technical budget limits (careful - loose)	• modify specification (never change once started - accept late changes)
32	• testing (basic test all functions - thorough test selected features)	(none)
21	use of teaching skills (good - bad)	• teaching (vocation - distraction)

Table 3 Comparison of Hypotheses and Interview Variables

Item ref. 22 appears twice as two variables are merged to one.

7.3.1 Merged List

Then merge the lists to give Table 4, showing the sources and omitting variables that are close matches to existing entries. The 'review' term includes the hypothesis items from Table 3 that have no 'interview' equivalent:

Ref	Behavioural Variable	Source
1	astronomy work (paper publishing - observing & research)	Jane, Kevin, Nick
2	attitude to management (management as authority - management as facility)	Helen
3	calibration (rely on others - do it personally)	Kevin, Nick
4	consider rewrite of legacy software (support - resist)	Bill
5	consideration of others (try to achieve consensus - take own view only)	Helen
6	consortium feedback to developer (vital - unconcerned)	Bill, Helen, Kevin
7	development programme (get job finished - add more features)	Bill, Kevin
8	display style (cluttered if necessary - clear layout with fewer items)	Bill
9	independent way of working (important - unconcerned)	Jane, Nick
10	interest in instrument design (involved - leave to others)	Jane, Nick
11	interested in work (do extra if needed - do minimum)	Bill, Helen, Jane, Kevin, Nick
12	isolate technical decisions from political influence (important - unconcerned)	Bill, Helen
13	keep self-motivated (understand big picture - concentrate on fine detail)	Helen
14	modify specification (never change once started - accept late changes)	Bill, Kevin
15	obtain results (simple plan to guarantee some results - complex plan with risk of no results)	Kevin
16	operation of new instrument (take the time to learn - use existing methods)	Jane
17	organisational control to stop late changes (need it - unconcerned)	Bill, Kevin

Table 4 Merged Behavioural Variables

Ref	Behavioural Variable	Source
18	personal involvement at instrument inception (strong wish - not concerned)	Bill, Helen, Kevin
19	resolution of data (better spatial, poorer temporal - better temporal, poorer spatial)	Jane
20	resources for training developers in new methods (vital - unconcerned)	Bill
21	teaching (vocation - distraction)	Jane
22	team working (collaboration - individualism)	Bill, Helen, Jane, Kevin
23	time to produce development tools (vital - unconcerned)	Bill, Helen
24	training (action - reading manual)	Helen
25	trust in others (full - none)	Helen
26	usability development (spend time on interface design - rely on printed manual)	Jane, Nick
27	use of detector (best performance - longest life)	Nick
28	why do it (be first to discover something new - just a job)	Jane, Nick
29	care of instrument (exceptional - adequate)	review
30	financial accuracy (precise - minimum necessary)	review
31	finished instrument (best in class - minimum to meet specification)	review
32	testing (basic test all features - thorough test selected features)	review
33	attention to detail (good - bad)	review
34	public outreach (important - don’t care)	review
35	reporting & communication (thorough - minimal)	review
36	schedule progress (cautious - lax)	review
37	support astronomy funding (for - against)	review

Table 4 Merged Behavioural Variables

7.4 Mapping and Presenting the Data

The situation is complicated by the interviewees each typically having several roles within a space science context. This is normal for the topic, but poses some problems in analysing the responses. For example, a person may be a manager in one role, and an observer in another role on the receiving end of management decisions. Therefore on the behavioural variable scales, a two part nomenclature is used to represent both interviewee (upper part) and the user group (lower part):

K - the interviewee

a - the user group

Key

Interviewee:

K	Kevin	H	Helen
J	Jane	B	Bill
N	Nick	r	review

User group:

a	Instrument scientist	f	Operator
b	Designer & developer	g	Developer (post commission)
c	System tester	j	Science community
d	Trainer	k	Teaching & public
e	Observing scientist	m	Funding body

We then construct a simple graphical representation of each variable, with the limits at each end of a line and the variable name in the centre. The position of the interviewee and user group data pair then represents the relative balance between the two end points. Fig. 5 shows an annotated example from the data:

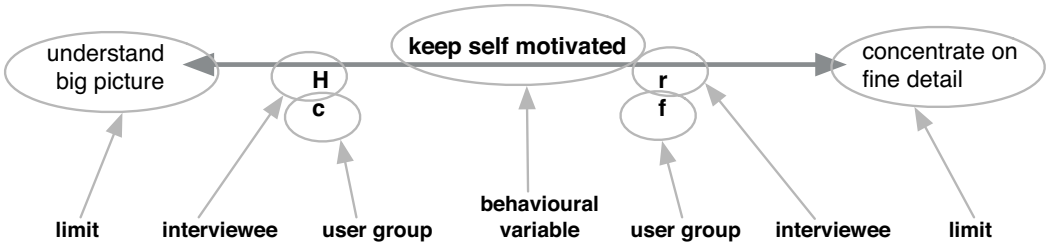


Fig. 5 Mapping Example

The position of the data pair is clearly a highly subjective decision, depending on personal judgement - that of the author, in this case. It would probably be better made by consensus between a small group of people, as would the choice of variables to be deduced from the interviews. What is more important though is that the decisions are consistent, thereby *normalising* the placements across the set of variables, and allowing the result to be reasonably balanced. The relative positions, and the order with respect to others, are the important parameters. In this respect, one person's judgement should be acceptably accurate, particularly if the assessments are all made in one session within a short time of each other.

This approach is one solution to the problem described earlier in Chapter One section 7 of how to extract numerical data, which in this case is ordering or the ranking of information, from non-numerical source material. The attractiveness of this particular approach is that there is no need to actually assign arbitrary numbers to the values of the data pairs. The numbers would only be present to allow a relative comparison to take place anyway. The graphical representation used, which is refined slightly from Cooper's description (Cooper 2003), takes minimal page space, allows easy comparison from one variable to another, and is drawn consistently so as not to introduce other implied but unexplained parameters. The original is drawn against a background grid to aid consistency, but this is not reproduced here as it appears to hinder comprehension by presenting visual clutter to the reader.

Appendix B Table 6 has details of the reasoning used to produce the 'review' (r) plots in Fig. 6. These represent points not brought out clearly in the interview data.

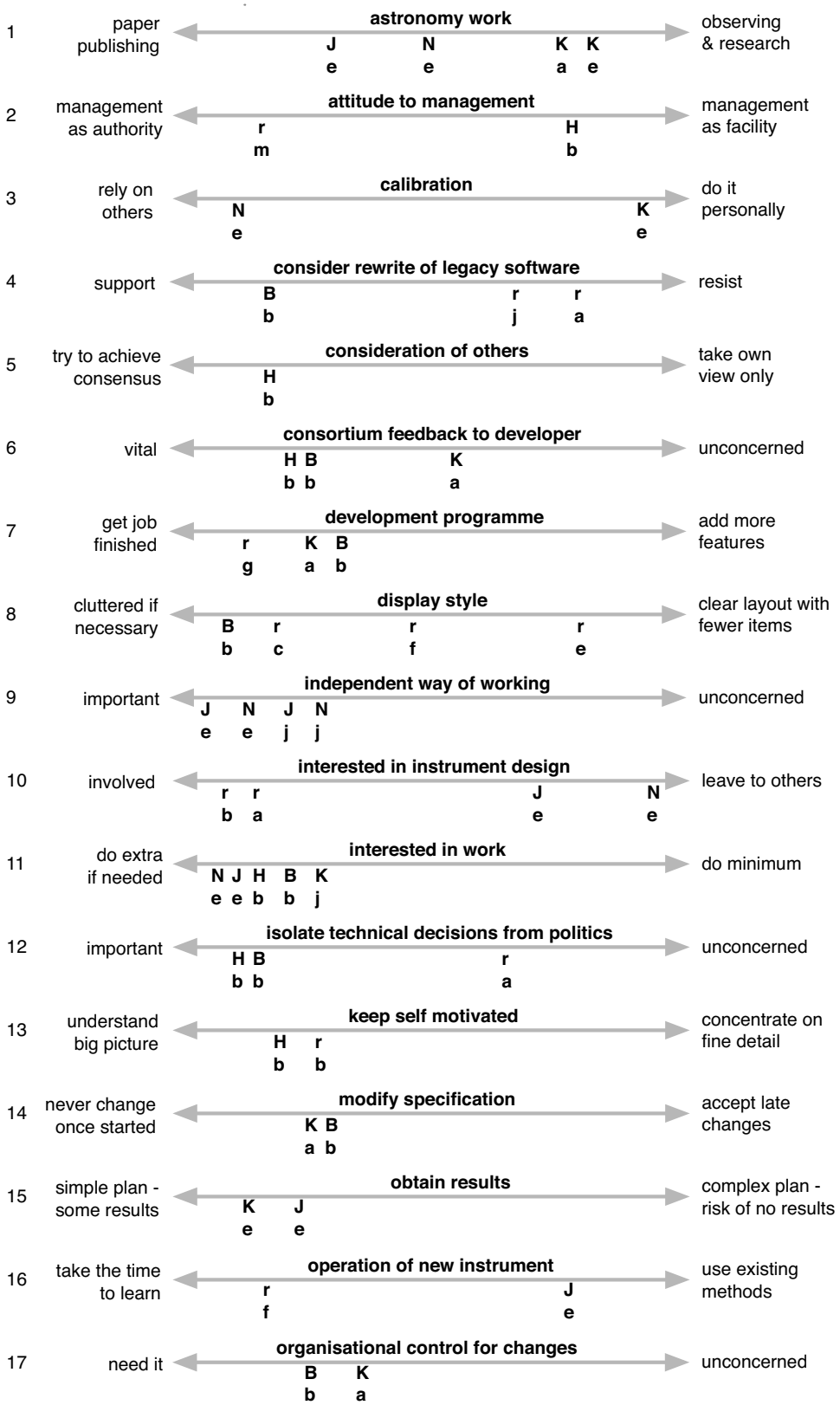


Fig. 6 Variable Data

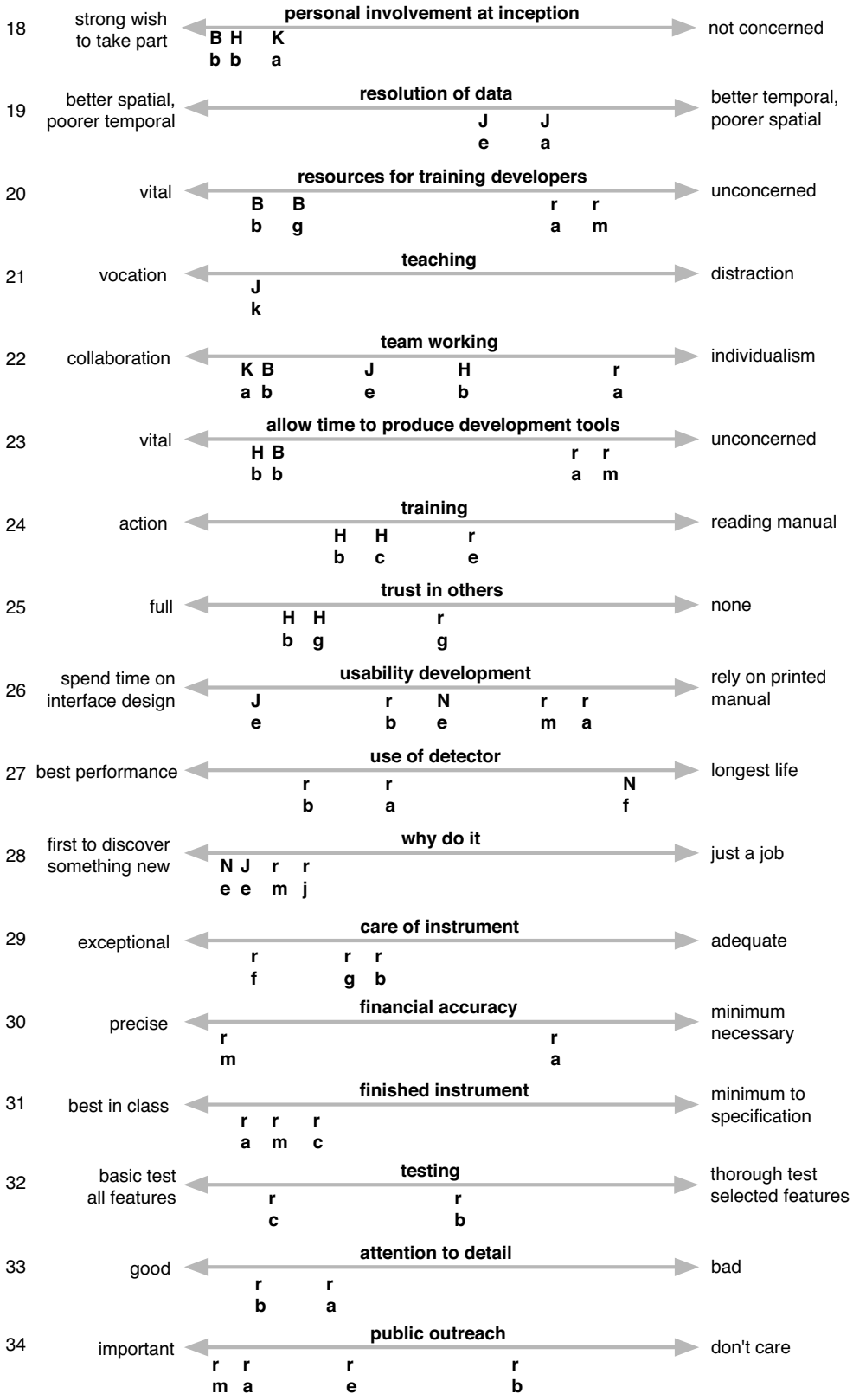


Fig. 6 Variable Data (cont)

The original intention was to use every variable, including those derived by looking at the initial hypothetical users and those from the reviews of the interviews. It proved very difficult though to put meaningful assignments against some of the hypothetical users, a fact which probably reflects inexperienced choices of variables, and so the following behavioural variables are omitted from the analysis:

35. reporting & communication (thorough - minimal)
36. schedule progress (cautious - lax)
37. support astronomy funding (for - against)

7.5 Extracting Results

The expectation is that plotting users against behavioural variables should show clusters where the goals of users overlap. The method is to look for clusters of group (a) with group (b), then (a) with (c), ... , (b) with (c), (b) with (d), and so on. The behavioural variables linked by each cluster form one aspect of a synthesised user, or *persona*.

7.5.1 Clusters

So what do the plots in the graphs above show? There are some clusters of users, such as for “independent way of working” (9) and “interested in work” (11), where all users indicate at one extreme. The same is true of “personal involvement ...” (18), “why do it” (28) and “finished instrument” (31); clearly from these opinions space science is more than just a job and personal motivation is important. In terms of one particular group of user showing a few clusters there is nothing very obvious, except for maybe both developer interviewees (B and H) showing the same opinions *and* grouped on their own for the variables of “consortium feedback …” (6), “isolate technical decisions …” (12), “resources for training …” (20), and “…development tools” (23). To re-state the process, these interviews were carried out with absolutely no collusion that the author is aware of between the various parties. Some variables lead to no conclusion; the ranking of astronomy work (1) between “paper publishing” and “observing and research” is spread out widely, and the spread is probably due to the demographics of the interviewees.

One conclusion must be that in terms of looking for clusters, the data are too thin. There probably needs to be at least twice the number of interviewees, and a little more care taken in the interview to bring out opinions where each person is involved in a specific role. Each person may well have multiple roles in the field of space science, but they need to be discussed separately. Roughly equal numbers of each user group would help to give a more balanced result. Some useful extra information could be gathered by going back to the original interviewees and using more targeted questions based on the behavioural variables

extracted after the first interviews, although there is a risk of falling into the trap of using leading questions and receiving biased answers. Interviewing is not a precise technique, and it would be reasonable to presume that practice might bring out more points, allowing extraction of additional behavioural variables.

One slight pattern that does emerge for some variables is the grouping (as in 20, 23, 25) between developers (groups 'b' and 'g'), and the separation from others (such as in 2, 4, 6, 12, 20, 23, 25, 27). In retrospect maybe this is not surprising, as an instrument developer's role is one of dealing with extreme detail whilst keeping a larger view. There will be typically several specialisms within the development team as well. Very few people understand or have experience of the level of detail required for development work, and of how catastrophic simple errors can be, that it may be the path of least resistance for those not involved not to try to understand too much.

One factor not brought out in the data links the developer groups (b & g), trainer (d) and operator (f); it is that they all need to operate the instrument in real time. Other user groups, as discussed in section 3.4.2, only interact with the instrument indirectly through an operator.

7.5.2 Developer Group

One consideration during this analysis was that of excluding the developer groups from the behavioural variable ranking, as they are responsible for generating the user interface to which this analysis is leading and therefore would not be users in the normal sense of the word. This was recognised as mistaken however, as typically only part of the development work is the user interface and there is a huge amount of other work in all the engineering disciplines. One option would have been to split the 'developer' category into several smaller parts, but it was felt that this would have increased the analysis complexity without useful gain. Another important point is that for much of the time a developer is an actual user of the interface under construction, and the earlier in the programme that this is defined the more opportunity for refinement it will have. It is looking as though one persona may well be the "developer".

To return to the graphical plots, can we use these to extract or confirm what might be another persona? No pattern seems to emerge, although this is possibly due to a thin data set. But here's an interesting speculation: could all the interface needs for a space instrument be met by catering for the developer group alone? This speculation excludes the science analysis work external to the instrument; it is looking simply at the instrument functions and including on-board data handling. If we define the developer role broadly as 'make everything work to specification', then what is true is that at some time in an

instrument programme members of the developer group will have to exercise *all* the functions of the instrument. No-one else will need to do this. This seems a very powerful argument, but we need to be careful that it is not overlooking any facts. For example, we could look again at the concept of different areas of responsibility within the overall 'developer' category and whether it would be appropriate to plan the development of different user interfaces for these different areas. There seems to be little advantage in following this route and plenty of negative aspects, not least that the teams for this work tend to be small and individuals need to overlap their work with the other members of the team. There seems to be every incentive for all developers to be aware as possible of how to operate the growing instrument and a single user interface is a good way to maximise this awareness. A precautionary point in this train of thought is to be aware of the hazards identified by the negative persona described in section 6.5.6.

7.5.3 Software Tools

One can take this single interface idea a couple of stages further. The first point is that bespoke software and hardware debugging tools are useful during the development programme and will be created for a particular task and then discarded later. Instead of seeing them as temporary items, these tools should be regarded as valuable parts of the programme and integrated into the instrument and support systems. This may mean more thoroughness in design, but that should be seen as beneficial. It reflects a common industrial approach of "design-for-test", where the test phase is regarded as the most demanding part of the life of a piece of equipment and the design is adapted accordingly. The only exceptions to this approach may be where system resources are challenged, such as mass, electrical power or data storage space.

The second point is to regard the scientists' software tools for planning observations using the instrument as part of the development programme. They have typically been written by members of the science team, usually with a completely different software architecture to the main development tools. By integrating these science tools into the instrument main development and using them on a routine basis, it is likely to produce a better quality result for possibly less overall effort. Some cultural barriers may have to be dismantled and nominal science researchers may find themselves as explicit members of a development group, but overall that should only promote good team working and wider understanding of the issues. This thinking actually brings all the instrument development together, rather than have a blurred distinction between instrument development and the analysis of the instrument science data. It then becomes clear that the "quick look" science analysis software, written to give early indications of science data quality with reasonable accuracy

and fast response, also properly belongs in the development domain. It can be used at various stages of instrument test, as well as during the orbital mission.

7.6 Building the Personas

7.6.1 Simplifying original hypotheses

From the interview data and discussion in sections 7.5.1 and 7.5.2 it appears that the designer and developer (group b) has a large amount in common with the post commission developer (group g), the system tester (c), and the operator (f). In organisation terms, group (b) appears to be a superset of g, c and f. This set is divided by one major event - the launch. This is where the special requirements of space science are felt, as a problem before launch may be recoverable, whereas a hardware-related problem after launch is likely to lead to a loss of functionality. To include this detail the groups here are divided into (b) and (c) for pre-launch, and (g) and (f) post launch. Also the trainer (d) in this context is a somewhat generic name for a person teaching others how to use the instrument at a technical level. It is likely to be a very occasional role taken on by a member of the “skilled users” set and would typically be an existing operator. On this occasion then we regard it as a superfluous distinction and treat it as exactly equivalent to an operator. As the word ‘developer’ has existing contexts, we need to employ a word without much previous usage and will use the terms “pre-launch and post-launch technologist” from here forwards to describe this role. This allows the simplification of five of the original ‘user hypotheses’ to just two new personas. Fig. 7 illustrates this.

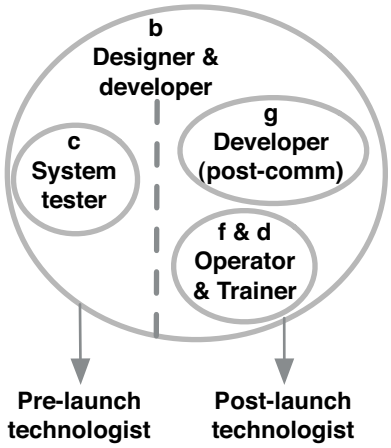


Fig. 7 Technologist Groups

As there are no other obvious groupings from the interview data, we go back to the user hypotheses. This leaves the instrument scientist (a), the observing scientist (e), the science community member (j), the public network user (k) and the funding body member (m) as probable personas.

We now start putting the concept of the persona approach together in some detail. The major purpose of constructing realistic fictional characters is to build mental models of potential users in the minds of the development team. We do this by first listing a set of ‘characteristics’ of each group based on a judgement of the interview data and adding a minimum number of imagined personal details to put some life into the character. Based on

this we then create sets of end, life and experience goals to round off each group's model. We do this for each group and then finish off the process by constructing a narrative based on these lists fully to round out the persona. In this chapter just the group of characteristics is shown as this is a traceable deduction from the interview data and adequate to show the flow of thinking; the fuller synthesis of an example persona which involves invention of details is given in Appendix C.

From the interview data, the following points are relevant for the revised user groups and are listed in no particular order. The source from the merged behavioural variable list in section 7.3.1 is given in brackets.

7.6.2 Characteristics of 'Pre-launch Technologist'

- generally regard the role of management is to facilitate work (2)
- like an independent way of working (9)
- be open to specification changes but only after expert discussion (14)
- interested in space science work (10, 11)
- enjoy working with fine detail but need an overall view as well (13)
- would strongly like to be involved right from the start of a programme (18)
- there must be funds and time made available for technical training in new methods (20)
- team working methods must show good collaboration and trust (22, 25)
- developmental tools must be seen as part of the programme (23)
- usability and interface design is a necessary part of the work (26)
- very motivated by the idea of being first to discover something new (28)
- will tend to take reasonable - but maybe not perfect - care of an instrument (29)
- financial control is one of those uninteresting things that needs to be done (30)
- the instrument will be best in class if possible (31)

7.6.3 Characteristics of 'Post-launch Technologist'

Exactly the same characteristics will tend to apply as for the 'pre-launch technologist' above, except in place of "will tend to take…" substitute the following:

- will tend to take the best possible care of an instrument, as there is a risk of losing it completely (29)

7.6.4 Characteristics of Instrument Scientist (a)

- appreciates there's a balance between actual astronomy research and the need to write papers (1)

- do not really understand all that is involved in designing and fabricating an instrument (10)
- wants to be involved in the planning of new instruments (18)
- some software planning tools are difficult to use (26)
- much more interested in science than the financial work (30)

7.6.5 Characteristics of Observing Scientist (e)

These are very similar to the Instrument Scientist without the responsibilities of the planning and financial control of a new instrument (18, 30). In addition the interview data includes:-

- likes an independent way of working (9)
- tends to operate new instruments just like the old ones (16)
- very motivated by the idea of being first to discover something new (28)

7.6.6 Characteristics of a Science Community Member (j)

These are just the same as the observing scientist, except just as a user of the science data without the responsibility of planning an observation (1)

7.6.7 Characteristics of a Public Network User (k)

A public network user is used here to illustrate potential general public access to the existing observational database of an instrument for simple interest or for school teaching. It could extend to putting forward suggestions for observations.

- The characteristics are the same as the observing scientist, without the responsibility of writing formal science papers (1)

7.6.8 Characteristics of a Funding Body Member (m)

A member of a funding body is defined as unlikely ever actually to use a given instrument, but has a critical role in providing the means by which the programme happens. (Specific people may be both funding body members and scientists, for example, but this situation is covered by idealised persona who are unique to each role).

- Would like to think that their contribution helped discover something new (28)
- Is convinced that they can spot all the mechanisms for reporting overspends in a favourable light (30)
- Demands good accuracy in financial reports (30)
- Within the spending limits will do everything to ensure a first class instrument (31)

7.7 Full Persona Descriptions

If we were to proceed to a full design of a user interface, it would be appropriate here to include a narrative style description of each persona based on the bullet points above. The rationale of this is the power of narrative to serve as a tool for generating and validating design ideas as introduced in Chapter One and discussed by Cooper and Rheinfrank (Rheinfrank 1996) (Cooper 2003). One description is included in Appendix C just as an illustration; the details are not used elsewhere in this discussion. The details would need to be seen to be accurate and balanced against the other personas for a full interface design. The narrative would then form part of the scenario that would be constructed in order to design the interface.

7.8 Definition of Persona Types

We are now in the position of taking the descriptions of our idealised users and prioritising them in terms of design targets. The concept of personas relies upon designing each interface just for one single primary persona (Cooper 2003) and we use his definitions here, as described earlier in section 6.5. To restate those idealised personas:

Primary

Secondary

Supplemental

Customer

Served

Negative

To recap, in the space science context we have these personas:

Pre-launch technologist

Post launch technologist

Instrument scientist

Observing scientist

Science community member

Public network user

Funding body member

The requirement here is to match up the first list with the second.

7.8.1 Information Flows and Communication Model

In considering the persona definitions, the concept of information flow may be useful. In most instances of remote control, including fields outside space science, the information flow to or from the remote instrument may be described by three explicit streams. The technology by which these streams are carried is unimportant - we are simply referring to information flow. The streams are:-

- Direct commanding to the instrument
- Engineering telemetry from the instrument, carrying parameters describing the condition of the instrument
- Science telemetry from the instrument, carrying parameters about the subject the instrument is observing

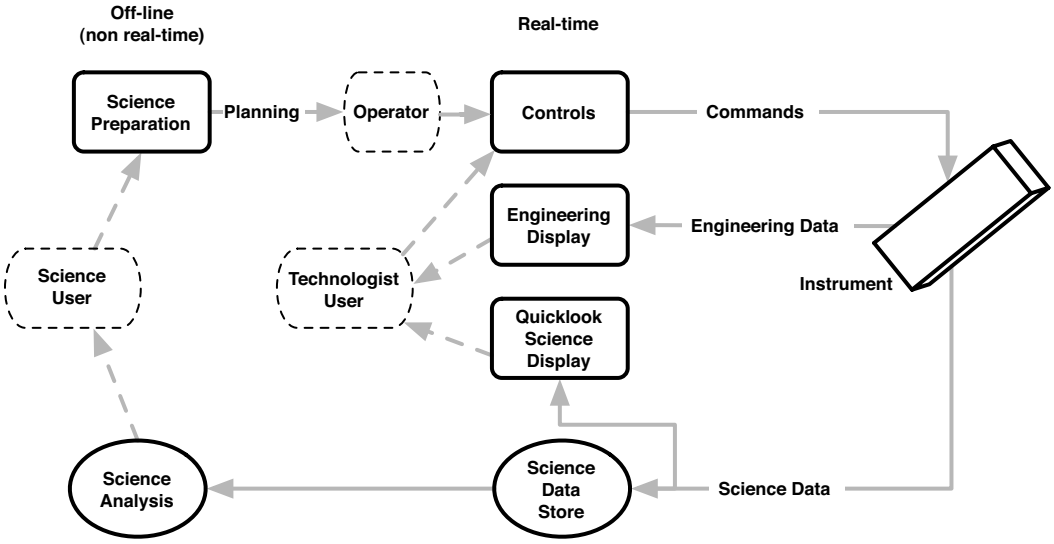


Fig. 8 Communication Model

The information flow to or from the users is through these information channels. If we assume an instrument configuration that includes an operator and off-line science preparation, as discussed in section 3.2.2, then the three streams above have the operator controls, the engineering display and the science data store with quicklook display at the other ends, as in Fig. 8. The quicklook science display is used to give a real time display of the data being stored.

If we consider the instrument user interfaces, full science data analysis is typically performed at some later date with software not under the control of the instrument teams and therefore is of no concern here. The science data store can be regarded as an automatic repository

with an interface provided at computer operating system level and is also not considered here. The operator controls, engineering display, quicklook display and the science preparation all need an interface with a user. The diagram shows that a fourth information stream (planning) can be said to exist from the science preparation to the operator controls, where some form of re-interpretation takes place before it is passed to the instrument.

There is actually an issue with the science analysis interface, as it could be argued that a consistent presentation and common tools might open up analysis of science data to more people and allow it to be done more thoroughly. This is difficult to refute, but the consequences of such an approach are large in terms of international collaboration. It is therefore considered out of scope of this discussion here. Some of the issues are covered in Chapter Six.

We can test the need for each interface against each space science persona, looking for the case of the *prime* persona and remembering that a persona is an idealised person. A real person may take on the roles of more than one persona, particularly in small teams. This shows in Fig. 8 in the use of the quick-look interface. With real instruments this is frequently used to perform a quick check by those whose main role is science analysis. In performing that function they are regarded here as taking on the role of the technologist.

Space science persona	Operator interface	Engineering interface	Quicklook interface	Science preparation interface
pre-launch technologist	X	X	X	-
post-launch technologist	X	X	X	-
Instrument scientist	-	-	-	X
Observing scientist	-	-	-	X
Science community member	-	-	-	-
Public network user	-	-	-	X
Funding body member	-	-	-	-

Table 5 Interface Selection

Table 5 clearly shows two groups which do not overlap, suggesting that two separate interfaces would be the best solution for operating a space science instrument. From the system diagram above, the operator, engineering and quicklook interfaces can be grouped together as the *real time interface*. The other can use the common space science term of *science planning tools*.

7.8.2 Real Time Interface

We consider persona assignment for the real time interface first. If we take the needs of either technologist, a design targeted at any of the science users or at the funding body user would not be sufficient. This suggests the technologist as a prime persona. The technologist is a major user of the real time interface during design, development, test, commissioning and operation, but does not use the science planning tools. The volume of work pre-launch is very much greater than post launch, and so if we refer to the pre-launch technologist as the prime persona and the post-launch technologist as the secondary persona, the definitions are satisfied. The secondary is the same as the primary, with the addition of a requirement to have an additional protection mechanism for the instrument to prevent irrecoverable mistakes in operation.

The instrument scientist and the observing scientist may need to use the planning tools and quick look parts of the real time interface and can be classified as supplemental personas. Their needs are completely met by the interface for the primary persona.

The science community member and public network user do not interact with the real time interface but are certainly directly affected by other peoples' use of it. They can therefore be classified as served personas.

The funding body member, with a high priority on financial accuracy, is a natural candidate as a customer persona and needs no interface to use.

The negative persona is not assigned here, although developers might seem a natural choice. The small teams in the typical space science organisation imply that the developers are very likely to be actual users of the real time interface just as much as they are originators of it. The caveats about negative persona in section 6.5.6 about allowing ad-hoc development to take place outside agreed design parameters still apply, but unless we break up the developer classification into several smaller groups it will be difficult to assign a negative persona. That break up does not seem on inspection to repay the effort.

7.8.3 Science Planning Tools

When we look at the science planning tools, a design targeted at the non-science users would clearly not be sufficient and another is required. With respect to the users, from sections 7.6.4 - 7.6.7 the observing scientist (e) appears to be a simple superset of the instrument scientist (a), the science community (j) and the public network user (k), illustrated in Fig. 9. We can then define the observing scientist as the primary persona and this will meet the need of the rest of the set. The instrument scientist, science community member and public network user are then all supplemental personas, as their needs are completely met by the primary.

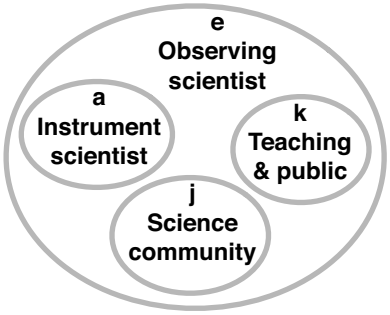


Fig. 9 Science Users

Members of the technologist group are not users of the science interface at all but are directly affected by its use. They are therefore classified as served personas.

As before, the funding body member is described by the customer persona and needs no interface.

The negative persona appears to require no assignment.

7.8.4 Definition of persona types

To summarise the results of this discussion:

Persona type	Real Time Interface	Science Planning Tools
Primary	Pre-launch technologist	Observing scientist
Secondary	Post-launch technologist	-
Supplemental	Instrument scientist	Instrument scientist
“	Observing scientist	Science community
“	-	Public network user
Customer	Funding body member	Funding body member
Served	Science community	Pre-launch technologist
“	Public network user	Post-launch technologist
Negative	-	-

Table 6 Persona Definitions

8 Summary

At this point we have explored most of what can be done towards a user interface in the space science context without a specific instrument for which to develop it. There is a clear result that two completely different interfaces are necessary to satisfy the requirements of the user groups. Also, the function of the quick look software is clearly shown to belong to the development and operational roles rather than the science planning side. Ideally the roles of the miscellaneous development tools referred to at the start of section 7 are covered by the 'real time interface' deduced here, and Chapter Eight introduces the programme flow ideas that are necessary for this to happen.

Here we have a methodology that can be used to analyse the needs and desires of a group of potential users of an area of technology and define a strategy for designing interfaces to an instrument operating in that area. It has a reasonable traceability, in that real data are gathered early on as interviews and can be re-examined if necessary. Whilst the data captured for the work in this chapter are a little thin, the method appears sound. Conceivably a slight adaption of this technique could, by providing a means of capturing potential users' wishes, frustrations and concepts, be used to take a forward look in a particular domain and harness ideas for future routes to follow. A robust technique for turning nebulous thoughts into detailed analysis may have many uses.

Section 6 describes the procedure that Cooper recommends (Cooper 2003) as the next stages in a real world development programme, and it is not apparent that much could be gained here by following that route as it would only be for an hypothetical instrument. Progressively more data would simply be speculative and not serve the purpose of this work.

The results of this chapter can be regarded as a suite of concepts that can be used to specify a user interface. There is a set of attributes describing general terms of interface design and interaction. There is an analysis of the spectrum of users and how they might work directly or indirectly with an instrument. The issues of access type, operation, timescales and propagation delay are discussed, and conclusions drawn. A first estimate of user and stakeholder groups is made and subsequently used as the hypothesis to seed the persona analysis. A first estimate of user goals is made. The persona method is explained and then worked through in detail with results from interviewees active in the context of the discussion. The outcome is a comprehensive statement of interface users and their goals, both technical and personal, which can be used as a basis for future designs.

Chapter Five: Usability Criteria Verification

The previous chapters examine in the context of space science the requirements for generic instruments and the range of their users. The purpose of this chapter is to establish some confidence in the validity of the usability criteria in Chapter Four by testing them against existing examples of interfaces for space instrumentation.

1 Usability Criteria Assessment

Six examples of working interfaces were selected which gave a distribution of different histories together with development over time. The EIS science planning tool is still under final development in early 2006 as this is written, but has intentional similarities with the CDS science planning tool produced in 1995 and subsequently modified. The specification for the EIS tool shows no explicit requirement to consider user interface details, and the author believes the earlier CDS tool evolved in a much less structured manner. The Hubble VTT tool however is a product of the NASA SEA and ESO JSky programmes as described in Chapter Six, and included a significant element of user analysis. The Gemini Position Editor had a similar early development, which then split from the path of the VTT tool. The EIS and CDS engineering interfaces have similar gestation dates to their respective science tools, although developed by different people.

Based on these backgrounds, and on the surmise that techniques often improve with time, one could reasonably expect the EIS tools and interfaces to show a better assessment against the usability criteria than the CDS items. Similarly, the integration of user analysis into the VTT development would lead one to expect that the VTT tool might have better usability than the Gemini tool. One might also expect both the VTT and Gemini tools to score significantly higher than the EIS or CDS tools, as the latter had no explicit analysis of user interaction.

Most of the usability criteria under the term ‘Operator Aspects’ in Chapter Four section 2.5 do not apply to the assessment of interfaces in isolation. They refer to the environment and team management, and therefore these eleven criteria are omitted from this discussion. One criterion is only relevant to the use of mechanical controls, which are not used in these examples, leaving fifty criteria for the engineering interface assessments. Fourteen of these are not appropriate for the non-real time science tool interfaces, leaving thirty six criteria for the science tools assessments. Appendix D has full details of the criteria which are not used, as well as unused assessment sheets. Each of the criteria is given a score from 5 to 0, representing full to no compliance.

Five of the six interface examples are software packages that could be executed by the author on a local machine and could therefore be explored in a dynamic manner as well as the simple static presentation. The dynamics of the remaining one, the CDS engineering interface, are well known to the author due to personal involvement in the CDS programme.

Table 1 lists the interfaces examined.

Programme & Instrument	Tool	Assessment	Type
Solar-B EIS	Science Planning	Table 2	Space flight
SOHO CDS	Science Planning	Table 3	Space flight
Solar-B EIS	Engineering	Table 4	Space flight
SOHO CDS	Engineering	Table 5	Space flight
Hubble	Visual Target Tuner	Table 6	Space flight
Gemini	Position Editor	Table 7	Ground based

Table 1 Interfaces Examined

Note: The references in the sub-headings in the following tables refer to the criteria developed in Chapter Four section 2, where a longer form of wording for the criteria can also be found.

1.1 Comparison of Usability Criteria for EIS Science Planning Tool

Fig. 1 is the window from the EIS planning tool used to construct the assessment in Table 2. A larger version is provided in Chapter Two Fig. 23.

Fig. 1 EIS Planning Tool Screen

Parameter: Section Solar-B EIS Science Planning Tool	Compliance Good -5 None - 0	Reasoning
Interaction - Accessibility: 2.1.1		
place controls according to usage	3	top down placement strategy
group controls and their displays	4	mostly grouped
group controls by function or sub-system	3	individual tools
allow easy discovery of control limits	2	some provided, many not
provide contextual help for data entry & controls	1	Not context sensitive, but help in window header
use the most appropriate interface devices	3	standard display, keyboard & mouse expected
Interaction - Adaptability: 2.1.2		
allow user to customise their system view	0	not customisable
allow for different levels of experience	1	no - assumes skilled users
allow parameters to be joined algorithmically	3	allows duration calculation
Interaction - Control: 2.1.3		
ensure user is in control	5	no unexpected system actions
anticipate possible user actions	5	nothing unanticipated found
Representation: 2.2		
take cultural conventions into account	2	default international convention
make controls & indicators self explanatory	2	needs basic understanding first
avoid clutter	3	some empty space could be better used
use an appropriate information density	3	reasonable

Table 2 Solar-B EIS Science Planning Tool

Parameter: Section Solar-B EIS Science Planning Tool	Compliance Good -5 None - 0	Reasoning
give immediacy of feedback for data entry	5	good
make new control settings visible immediately	5	responsive
avoid sensory overload	5	good
create clear mental model	1	no hints or diagrams
allow graphics use	3	graphics used for timeline
allow use of audible hints	0	not implemented
imitate the physical layout of the system	0	not appropriate
include all necessary parameters	-	not known
use colour appropriately	1	uniform grey is usable but misses opportunities
use controls of an adequate size	3	tiny radio buttons, otherwise good
Consistency: 2.3		
avoid ambiguity in controls and displays	4	small amount of ambiguity
ensure layout consistency	4	good, limited use of screen elements
ensure consistency in operation of controls	5	just push buttons and menus
use accepted design practice if appropriate	-	not known
use public standards for colour & contrast	-	no use of colour, some use of contrast
Error management: 2.4		
check entries as they are entered	3	database lookup
suggest values to user if computable	5	computes raster length, etc
avoid dialogue boxes if possible	2	dialogues used extensively

Table 2 Solar-B EIS Science Planning Tool

Parameter: Section Solar-B EIS Science Planning Tool	Compliance Good -5 None - 0	Reasoning
correct errors easily	5	simple re-entry
Operator Aspects: Workload: 2.5.2		
provide ready filled data fields	5	good
avoid reliance on user memory	5	no need found to take notes

Table 2 Solar-B EIS Science Planning Tool

Note: Compliancy in this table has been checked against EIS planning tools release available on 20 Feb 06.

1.2 Comparison of Usability Criteria for the CDS Science Planning Tool

The EIS instrument is related in several ways to the eleven years earlier CDS instrument on the ESA SOHO mission, including the overall design behind the science planning tool. The implementation of the CDS science planning tool, illustrated in Fig. 2, is compared in Table 3 in the same manner as the EIS tool against the criteria in Chapter Four.

The CDS interfaces were designed in ~1995 and were limited by the tools and machines available then.

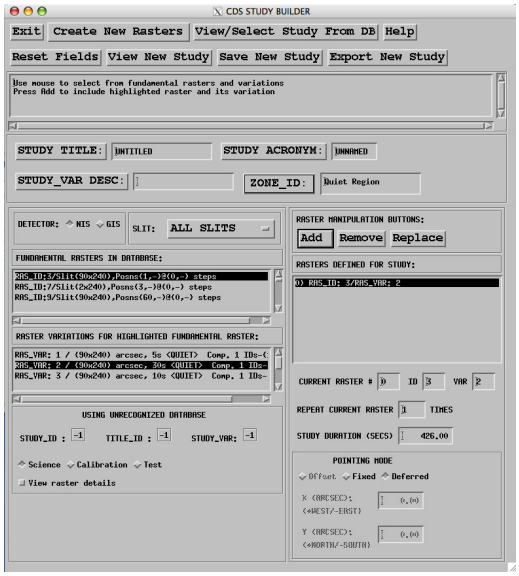


Fig. 2 CDS Planning Tool - Study Builder

Parameter: Section SOHO CDS Science Planning Tool	Compliance Good - 5 None - 0	Reasoning
Interaction - Accessibility: 2.1.1		
place controls according to usage	3	controls are scattered
group controls and their displays	2	no close grouping
group controls by function or sub-system	3	done by sub-system
allow easy discovery of control limits	0	only from documentation
provide contextual help for data entry & controls	0	none
use the most appropriate interface devices	3	standard display, keyboard and mouse
Interaction - Adaptability: 2.1.2		
allow user to customise their system view	0	not customisable
allow for different levels of experience	1	assumes skilled users
allow parameters to be joined algorithmically	0	not apparent
Interaction - Control: 2.1.3		
ensure user is in control	5	good control
anticipate possible user actions	5	nothing unanticipated found
Representation: 2.2		
take cultural conventions into account	2	English only
make controls & indicators self explanatory	1	need lot of familiarity
avoid clutter	4	reasonable
use an appropriate information density	4	good
give immediacy of feedback for data entry	5	good
make new control settings visible immediately	5	responsive
avoid sensory overload	5	good
create clear mental model	1	no attempt at hints

Table 3 SOHO CDS Science Planning Tool

Parameter: Section SOHO CDS Science Planning Tool	Compliance Good - 5 None - 0	Reasoning
allow graphics use	3	used for timeline
allow use of audible hints	0	not implemented
imitate the physical layout of the system	0	not appropriate
include all necessary parameters	-	unknown
use colour appropriately	1	very little advantage taken of colour
use controls of an adequate size	3	most controls are a good size, but some are tiny
Consistency: 2.3		
avoid ambiguity in controls and displays	1	command lines may only differ by one character
ensure layout consistency	4	very predictable
ensure consistency in operation of controls	5	all similar in operation
use accepted design practice if appropriate	-	not appropriate
use public standards for colour & contrast	-	cannot assess
Error management: 2.4		
check entries as they are entered	3	database lookup
suggest values to user if computable	0	not done
avoid dialogue boxes if possible	2	boxes used
correct errors easily	3	command line retyping
Operator Aspects: Workload: 2.5.2		
provide ready filled data fields	5	good
avoid reliance on user memory	5	no need for notes

Table 3 SOHO CDS Science Planning Tool

1.3 EIS Engineering Interface

The previous examples show analyses of the science planning interface for EIS and CDS. Many of the criteria of Chapter Four are not relevant to a planning tool and so the same

process is applied instead to an engineering interface. The EIS instrument is chosen again as it is of recent design (early 2006), and available to the author for operational evaluation. This is important as many of the criteria adopted are concerned with action rather than just appearance. The EIS operation is designed primarily for one operator and so issues of team working are not relevant.

Fig. 3 is the command window, reproduced in a small size here as it has already been shown in detail in Chapter Two Fig. 25. The right hand side allows commands to be sent for a specific sub-system selected by the tabs at the top right, with the operator prompted to fill in command parameter fields where necessary. The left hand side is a scrolling display of all the commands sent.

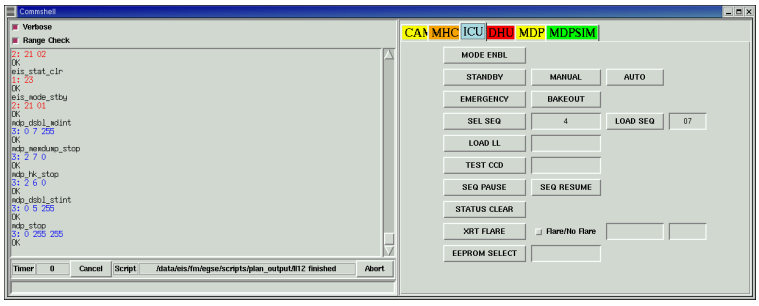


Fig. 3 EIS Command Display

Fig. 4 below is the telemetry display, also available in Chapter Two as Fig. 26. The image is intended to fill a large monitor and consequently appears very compressed here. The display shows just one subsystem selected out of several by the tabs at middle and top left. The top half of the telemetry display allows selectable parameters to be shown (for example) against telemetry packet number, whilst the lower half shows the state of all the parameters for that sub-system. These displays are assessed in Table 4.

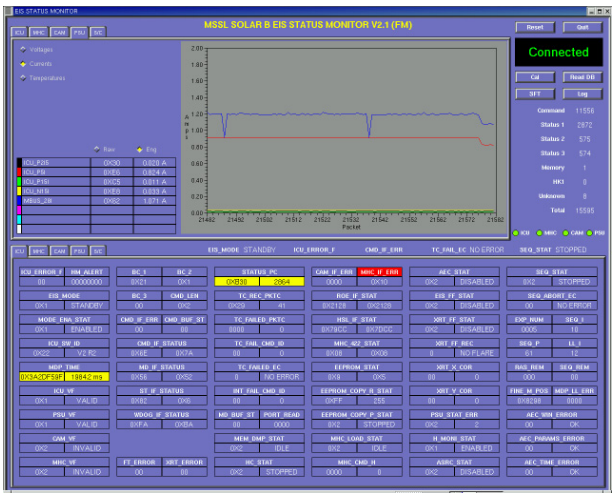


Fig. 4 EIS Telemetry Display

Parameter: Section Solar-B EIS Engineering Control	Compliance Good - 5 None - 0	Reasoning
Interaction - Accessibility: 2.1.1		
place controls according to usage	3	tabs plus hierarchy
group controls and their displays	1	two different windows
group controls by function or sub-system	5	tabbed selection
allow command authority with visibility for all	-	not applicable - built just for one operator
allow easy discovery of control limits	0	not apparent
provide contextual help for data entry & controls	0	no help provided
use the most appropriate interface devices	3	display, keyboard & mouse - no others considered
Interaction - Adaptability: 2.1.2		
consider all users	3	need to be familiar with it first
allow user to customise their system view	0	not customisable
allow for different levels of experience	2	no particular provision
allow parameters to be joined algorithmically	4	basic sub-system check, but no customisation
allow resources for extra operators	-	not required
Interaction - Control: 2.1.3		
ensure user is in control	5	no unexpected system actions
anticipate possible user actions	5	no unexpected results due to user action
keep system logs, and make them available	5	comprehensive logs kept
Representation: 2.2		
take cultural conventions into account	2	English only

Table 4 Solar-B EIS Engineering

Parameter: Section Solar-B EIS Engineering Control	Compliance Good - 5 None - 0	Reasoning
make controls & indicators self explanatory	3	some explanation needed
avoid clutter	3	busy, but few spurious items
use an appropriate information density	2	telemetry display deliberately crowded
give immediacy of feedback for data entry	5	on scrolling command display
make new control settings visible immediately	3	relies on quick instrument communications
give full feedback of delayed response entry	-	not required
avoid sensory overload	-	no audible alarms used
create clear mental model	1	no representation attempted
allow graphics use	3	used for graph
allow use of audible hints	-	not used
allow audible alarms to be distinguishable	-	no alarms used
imitate the physical layout of the system	0	not attempted
include all necessary parameters	5	yes
use colour appropriately	3	colour used for highlighting
use controls of an adequate size	4	select buttons small at 10 pixel, others 25 pixel
Consistency: 2.3		
avoid ambiguity in controls and displays	4	clear but with minimal labels
ensure layout consistency	5	good
ensure consistency in operation of controls	5	just push buttons and tabs
use accepted design practice if appropriate	-	not considered
use public standards for colour & contrast	-	not considered, use of orange violates US default

Table 4 Solar-B EIS Engineering

Parameter: Section Solar-B EIS Engineering Control	Compliance Good - 5 None - 0	Reasoning
Error management: 2.4		
minimise risk of inadvertent control operation	3	no special protection, but low risk of problems
provide protection from unauthorised users	2	unix user protection only
identify hazardous operation	5	some power supply operations identified
signal abnormal operation clearly	5	coloured highlight for out of range variable
allow limits to be exceeded in an emergency	5	limit checking can be disabled
use interlocks to minimise risk	5	power switching interlock implemented
check entries as they are entered	5	range checking
suggest values to user if computable	-	not applicable
avoid dialogue boxes if possible	4	some used, but appropriate
correct errors easily	5	no problems
implement system redundancy if possible	-	no redundancy in instrument as no resources
avoid operator over-confidence	-	no hazardous operations
Operator Aspects - Workload: 2.5.2		
provide ready filled data fields	5	for location of log and data files
avoid reliance on user memory	4	commands shown as buttons

Table 4 Solar-B EIS Engineering

1.4 CDS Engineering Interface

Fig. 5 shows one screen from the CDS engineering interface. A larger version is reproduced in Chapter Two Fig. 11. The body of the page has text lines giving a mnemonic, its value and the units. Comment text is used also. The top of the page has two bush button controls to stop and start the telemetry stream. Commands are sent using a simple text window (not shown) into which the operator types the appropriate mnemonic. Table 5 gives the result of the assessment.

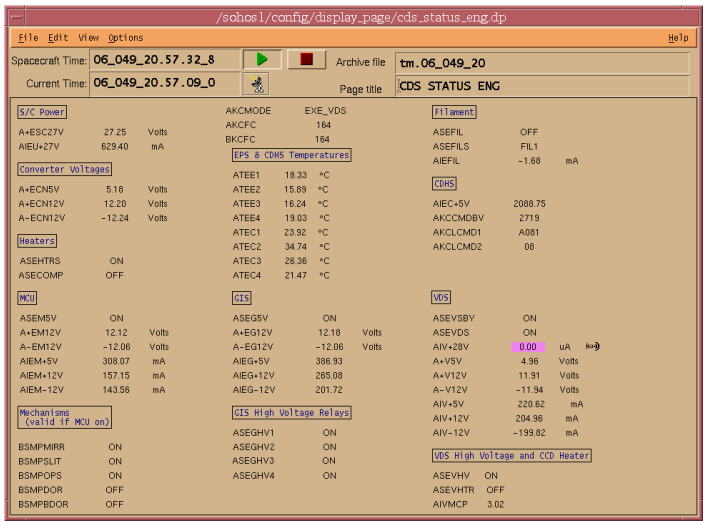


Fig. 5 CDS Engineering

Parameter: Section SOHO-CDS Engineering Control	Compliance Good - 5 None - 0	Reasoning
Interaction - Accessibility: 2.1.1		
place controls according to usage	0	command line interface
group controls and their displays	0	command line interface
group controls by function or sub-system	3	telemetry controls grouped
allow command authority with visibility for all	-	one operator with command line input
allow easy discovery of control limits	0	requires manual lookup
provide contextual help for data entry & controls	0	no help
use the most appropriate interface devices	3	only keyboard, mouse and display considered
Interaction - Adaptability: 2.1.2		
consider all users	2	need considerable familiarisation
allow user to customise their system view	3	parameters moveable, new pages can be built
allow for different levels of experience	1	not explicitly designed in
allow parameters to be joined algorithmically	-	not known
allow resources for extra operators	-	not applicable
Interaction - Control: 2.1.3		
ensure user is in control	4	command line gives good control
anticipate possible user actions	2	out of range traps
keep system logs, and make them available	5	verbose system logs kept
Representation: 2.2		
take cultural conventions into account	2	English only
make controls & indicators self explanatory	3	indicators allow units and comments

Table 5 SOHO CDS Engineering

Parameter: Section SOHO-CDS Engineering Control	Compliance Good - 5 None - 0	Reasoning
avoid clutter	4	indicator framework is clear
use an appropriate information density	3	system would not allow very close spacing
give immediacy of feedback for data entry	4	visible in scrolling display
make new control settings visible immediately	2	parameters needed response from instrument
give full feedback of delayed response entry	5	display driven by delayed response
avoid sensory overload	-	no alarms used
create clear mental model	1	no representation attempted
allow graphics use	1	elementary graphics
allow use of audible hints	-	not appropriate for command line
allow audible alarms to be distinguishable	-	not used
imitate the physical layout of the system	0	not attempted
include all necessary parameters	5	whole database used
use colour appropriately	3	coloured background and comments appropriate
use controls of an adequate size	5	telemetry controls are good
Consistency: 2.3		
avoid ambiguity in controls and displays	2	displays use short, easily confused mnemonic
ensure layout consistency	3	framework clear, but no grid layout option
ensure consistency in operation of controls	5	two telemetry buttons
use accepted design practice if appropriate	-	not known
use public standards for colour & contrast	-	not known

Table 5 SOHO CDS Engineering

Parameter: Section SOHO-CDS Engineering Control	Compliance Good - 5 None - 0	Reasoning
Error management: 2.4		
minimise risk of inadvertent control operation	1	command mnemonics need careful attention
provide protection from unauthorised users	2	location and unix permissions
identify hazardous operation	3	hazards use extra prompt, but overrideable
signal abnormal operation clearly	5	warning and alarm colours used for telemetry
allow limits to be exceeded in an emergency	-	not known
use interlocks to minimise risk	3	scripts used for mandatory sequences
check entries as they are entered	5	range checking on entry
suggest values to user if computable	-	not appropriate
avoid dialogue boxes if possible	1	used for every hazardous command
correct errors easily	5	re-type line
implement system redundancy if possible	3	in parts of instrument and communications links
avoid operator over-confidence	3	pre-tested scripts used for most operations
Operator Aspects - Workload: 2.5.2		
provide ready filled data fields	-	not applicable
avoid reliance on user memory	0	commanding needs manual database lookup

Table 5 SOHO CDS Engineering

1.5 Hubble Visual Target Tuner

The Astronomer's Planning Tool (APT) for Hubble should have one of the better sets of interfaces in the field of space science due to its long history and the involvement of the NASA SEA programme, described in Chapter Six. As it is a suite of individual tools, the visual target tuner (VTT) is chosen for examination as it probably represents the most developed part of the suite, with a mix of graphics, text and controls. This analysis is shown in Table 6 and done using the stand alone tool shown in Fig. 6, as the fully integrated tool appears to require a valid proposal on which to run it. A larger illustration with a variation on this view appears in Chapter Two Fig. 17.

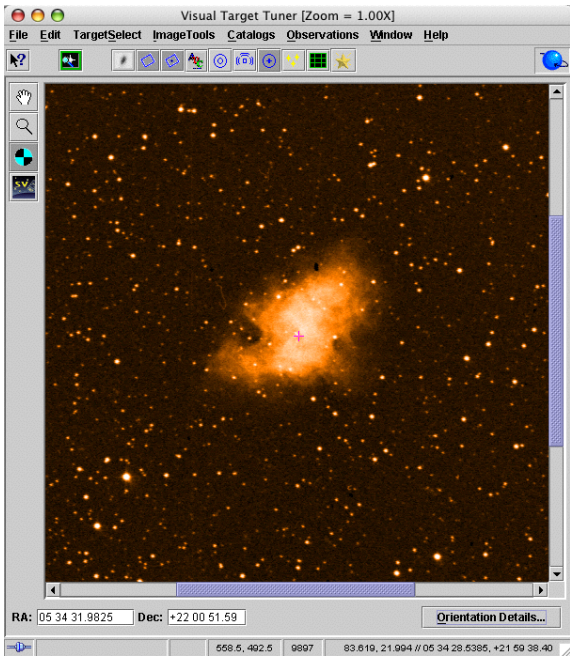


Fig. 6 Hubble Visual Target Tuner

Parameter: Section Hubble Visual Target Tuner	Compliance Good - 5 None - 0	Reasoning
Interaction - Accessibility: 2.1.1		
place controls according to usage	5	good - on main display and in dialogues
group controls and their displays	5	yes
group items by function or sub-system	5	done
allow easy discovery of control limits	4	generally clear
provide contextual help for data entry & controls	5	pop-up as well as context help window
use the most appropriate interface devices	3	keyboard, mouse and display
Interaction - Adaptability: 2.1.2		
allow user to customise their system view	4	customise colour, database, etc
allow for different levels of experience	5	easy for non-astronomer to learn, but good for expert
allow parameters to be joined algorithmically	-	no obvious provision
Interaction - Control: 2.1.3		
ensure user is in control	5	no unexpected actions
anticipate possible user actions	5	no errors discovered
Representation: 2.2		
take cultural conventions into account	2	American English only
make controls & indicators self explanatory	5	icons used frequently
avoid clutter	5	clean appearance
use an appropriate information density	5	good - including the help file
give immediacy of feedback for data entry	5	very good
make new control settings visible immediately	5	excellent to use

Table 6 Hubble Visual Target Tuner

Parameter: Section Hubble Visual Target Tuner	Compliance Good - 5 None - 0	Reasoning
avoid sensory overload	5	good
create clear mental model	5	overlays give impression of targeting
allow graphics use	5	relies on graphics
allow use of audible hints	0	not used
imitate the physical layout of the system	-	not appropriate
include all necessary parameters	-	cannot assess
use colour appropriately	5	grey background with appropriate contrasts
use controls of an adequate size	5	main buttons are 25 pixels minimum dimension
Consistency: 2.3		
avoid ambiguity in controls and displays	5	no ambiguity observed
ensure layout consistency	4	embedded applications are reasonable
ensure consistency in operation of controls	5	pushbuttons and menus only
use accepted design practice if appropriate	-	cannot assess
use public standards for colour & contrast	-	cannot assess
Error management: 2.4		
check entries as they are entered	5	including network lookup
suggest values to user if computable	5	good, limited opportunities
avoid dialogue boxes if possible	3	network error gave 'OK' dialogue
correct errors easily	5	fully interactive
Operator Aspects: Workload: 2.5.2		
provide ready filled data fields	5	good, limited opportunities
avoid reliance on user memory	5	no need for paper notes found

Table 6 Hubble Visual Target Tuner

1.6 Gemini Position Editor

The Gemini Observing Tool (OT) has a partial common heritage with the Hubble APT, but development appeared to split in later years. This example examines the Position Editor, used to preview images from databases. A larger version of the image in Fig. 7 is included in Chapter Two as Fig. 19. The analysis is shown in Table 7.

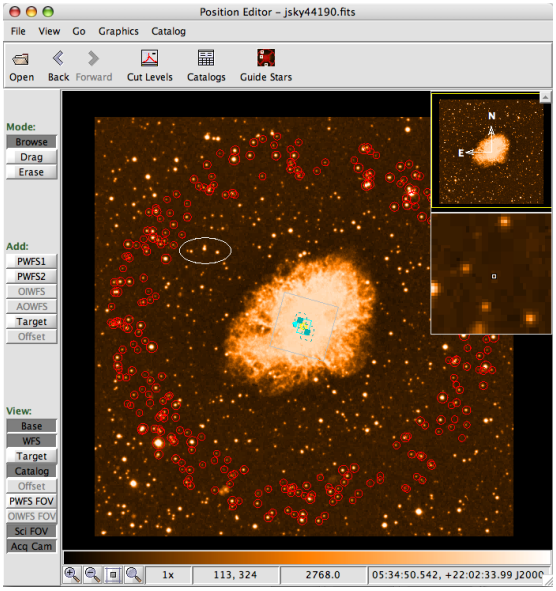


Fig. 7 Gemini Position Editor

Parameter: Section Gemini Position Editor	Compliance Good - 5 None - 0	Reasoning
Interaction - Accessibility: 2.1.1		
place controls according to usage	5	very accessible
group controls and their displays	5	grouped around display
group items by function or sub-system	5	grouped by function & clearly labelled
allow easy discovery of control limits	4	generally clear
provide contextual help for data entry & controls	3	pop-up text; main help fails needing Netscape
use the most appropriate interface devices	3	mouse & keyboard; no graphics tablet features
Interaction - Adaptability: 2.1.2		
allow user to customise their system view	4	features can be selected, colours chosen, etc
allow for different levels of experience	3	no keyboard shortcuts
allow parameters to be joined algorithmically	-	no obvious provision
Interaction - Control: 2.1.3		
ensure user is in control	4	straightforward to use
anticipate possible user actions	5	no unexpected action found
Representation: 2.2		
take cultural conventions into account	2	only single language etc; appropriate for context
make controls & indicators self explanatory	4	good
avoid clutter	5	neatly arranged
use an appropriate information density	5	appropriate use of white space
give immediacy of feedback for data entry	5	tool is responsive
make new control settings visible immediately	5	radio buttons are responsive

Table 7 Gemini Position Editor

Parameter: Section Gemini Position Editor	Compliance Good - 5 None - 0	Reasoning
avoid sensory overload	5	difficult to see how to improve it
create clear mental model	5	overlays on image give clear orientation
allow graphics use	5	based on graphics
allow use of audible hints	0	not implemented
imitate the physical layout of the system	-	not appropriate
include all necessary parameters	-	not testable
use colour appropriately	5	good for image and controls
use controls of an adequate size	5	smallest is about 15 pixels - is adequate
Consistency: 2.3		
avoid ambiguity in controls and displays	5	no obvious ambiguity
ensure layout consistency	4	simple and straightforward
ensure consistency in operation of controls	5	just buttons and menus
use accepted design practice if appropriate	-	not able to assess
use public standards for colour & contrast	-	not able to assess
Error management: 2.4		
check entries as they are entered	5	network lookup
suggest values to user if computable	5	cursor continuously tracked
avoid dialogue boxes if possible	3	network error gave ‚ÄòOK‚Äô dialogue
correct errors easily	5	no difficulty entering fresh values
Operator Aspects: Workload: 2.5.2		
provide ready filled data fields	5	database lookup fills in fields
avoid reliance on user memory	5	no occasion found where notes needed

Table 7 Gemini Position Editor

2 Results

2.1 Method

Achieving a consistent assessment on each of the criteria for each of the interface examples proved to be very difficult. The criteria are necessarily somewhat subjective, depending on the assessor, immediately previous experience, distractions and such factors. The approach initially adopted was to set up one software application, run through the complete list of criteria noting comments and putting a tentative score against each. The next application would then be set up and assessed, and the process repeated. Examining the results at the end of the exercise, and after several hours spread over a few days, revealed inconsistencies both in comments and scores. In order to present the results above, the procedure was repeated in a different manner. Some time was spent with all display screens printed on paper and spread out to be visible simultaneously. Care was taken to go through each criterion one at a time, ensuring an equal judgement was applied and that scores were consistent with the comments. The applications were run again if appropriate. Scoring one interface directly against another also allowed a much better interpolation between the extreme scores of good or none. The process makes better use of human short-term memory by enabling side by side comparisons to take place within a very few seconds.

A more thorough approach might have been to have a sufficient number of machines, or at least sufficient screen space, to allow all six interfaces to be both visible and active at once. As above, each criterion would then be tested against each interface and a consistent result arrived at before moving to the next criterion, and this process repeated to the end of the list. It would probably be wise to allow enough time for this to be attempted in one session without distractions. Lack of easily available resources prevented this from being tried.

These are important points in attempting to use these lists of criteria to assess, for example, a newly created interface. Whilst a simple testing of each point could be adopted, far more meaningful results are likely if an existing and well-tested interface is assessed at the same time and point by point. More than one additional interface would be likely to strengthen the analysis. The extra interface or interfaces can be seen as reference points, allowing the new work to be anchored to existing practice.

Some criteria are simply not implemented in the interface examples here, or are impossible to judge, and have a score shown as ‘-’. This has been treated as zero in making an overall assessment.

Several criteria are difficult for a third party to assess, such as in this exercise, because they depend on understanding the precise dynamics of the interface. One example might be

‘provide ready filled data fields’. This is relatively easy for the development team at the time to judge, which is what one should plan to make happen.

2.2 Scores

Points that cannot be assessed are marked as ‘-’ and counted as zero. Quantities of zero assessments occur in specific pairs, thereby not invalidating the specific comparisons made.

The science tools are tested against 36 criteria, giving a maximum score of 180.

The engineering interfaces are tested against 50 criteria, giving a maximum score of 250.

Table 8 summarises the results, and includes an indication of the date the software tool became available or the period over which it has been updated. An estimate of related costs would also have been very useful, but the data are not readily available.

Interface	Score	%	Unassessed	Date Implemented
VTT	140	78	5	2001 - 2006
Position editor	134	74	5	2001 - 2006
EIS science	101	56	3	2006
CDS science	84	47	3	1995 - 2005
EIS engineering	132	53	11	2006
CDS engineering	102	41	11	1995

Table 8 Interface Scoring

These results are in line with the expected ranking described above in section 1. The risk of scores unintentionally biased to show what was postulated is real, but was mitigated as far as possible by not computing these totals until the individual scores were determined. The author was therefore at least partially blind to the developing totals. The two tools of the VTT and the Position Editor derived with user analysis show significantly higher scores than the EIS or CDS tools, with the VTT leading slightly. The newer EIS tool in both cases leads the older CDS tool.

3 Limitations of this analysis

To paraphrase Cooper as described in Chapter Two section 1.5, standards should be treated as guidelines or rules of thumb and should be followed unless there is a clear alternative (Cooper 2003).The criteria used here are based on a combination of published research work, material from generally accepted experts in the field, and the case studies in this thesis. That information has then been selected to optimise it for the context of space flight instrumentation. The most likely error in this process is simple omission of what another observer would consider as important. Another limitation is that no weighting has been attempted for each of the criteria; they are all presented as though they were of equal importance. This latter approach has been taken as it is difficult to see how a meaningful grading could be applied without it being extremely dependant on the precise subject matter used for the grading exercise. To prove stability of these values (or otherwise) was seen as more work than could be attempted here, and the risk of the effort being nugatory was seen as unacceptably high.

Some of the criteria in Chapter Four are more applicable to situations where there is a team rather than an individual in control of real-time commanding, and the near loss of the SOHO spacecraft covered in Chapter Three was one reason for including them. Some other criteria are only relevant for instruments that could be damaged by inappropriate commands. Examples of this might be pointing an astrophysics telescope directly at the Sun and burning the detector, or applying too much voltage to a photon multiplier and causing breakdown. Many instruments do not allow a wide degree of pointing or do not have high voltage systems. Even fewer systems currently have sets of mechanical controls, but for those that do have them the risk of problems such as parallax errors is important, and hence this factor is included.

The criteria are deliberately general in nature. For example, ‘use colour appropriately’ is a reminder to think about colour, and to refer to the guides referenced earlier to achieve the desired effect. It is also a reminder to refer to recommendations for dealing with colour deficient vision.

In presenting these examples of interface analysis in this chapter, it has been a useful check of the criteria themselves and some refinement of the wording and selection has been made. It seems reasonable to state that continued use in this manner on other interface implementations should be the best way to create and refine a set of criteria that is focused and robust. Ideally judgement on the precise form of these criteria should be made by a group of people with different backgrounds to avoid the inevitable bias of a single observer.

The use of a group however begs the question that would need to be resolved beforehand, of whether any decision should be based on a majority vote, unanimity, or simple consensus.

4 Visual Presentation of Results

It is useful to take the results from Table 2 to Table 7 and plot each of them on a polar graph, also known as a ‘spider plot’, as in Fig. 8 to Fig. 13. This leads to a series of compact diagrams that allow a quick visual judgement to be made after assessing a given interface design, and also allows an easy comparison to be made between two or more sets of results. A score of 0 is plotted in the centre, and a score of 5 is plotted on the outer circle. The resulting shape of the plot is clearly somewhat arbitrary, as the parameters are simply presented in the same order as on the tables of results above. These are grouped loosely by some common areas, but there is no other organisation. However, since the same order is used in all the plots of one type, the overall plot shape can be used as a means of comparison. Cross comparison of the science tool interface plots against the engineering interface plots is not particularly useful, as the latter have fourteen more parameters variously inserted amongst the parameters of the former.

The EIS and CDS science tool interface plots in Fig. 8 and Fig. 9 show the evolution of one from the other, retaining the same overall shape. The EIS plot shows that some extra parameters are covered with its interface. The Hubble and Gemini plots in Fig. 10 and Fig. 11 also show their common heritage by their overall shape. The significantly greater area covered by the Hubble and Gemini plots compared to the EIS and CDS plots is an indicator of their greater design maturity.

The EIS and CDS engineering interfaces in Fig. 12 and Fig. 13 show only a superficial similarity to each other. EIS shows a greater area covered by the plot, which should be expected of the later design and a greater awareness of user interface requirements.

Appendix D has unused copies of these diagrams.

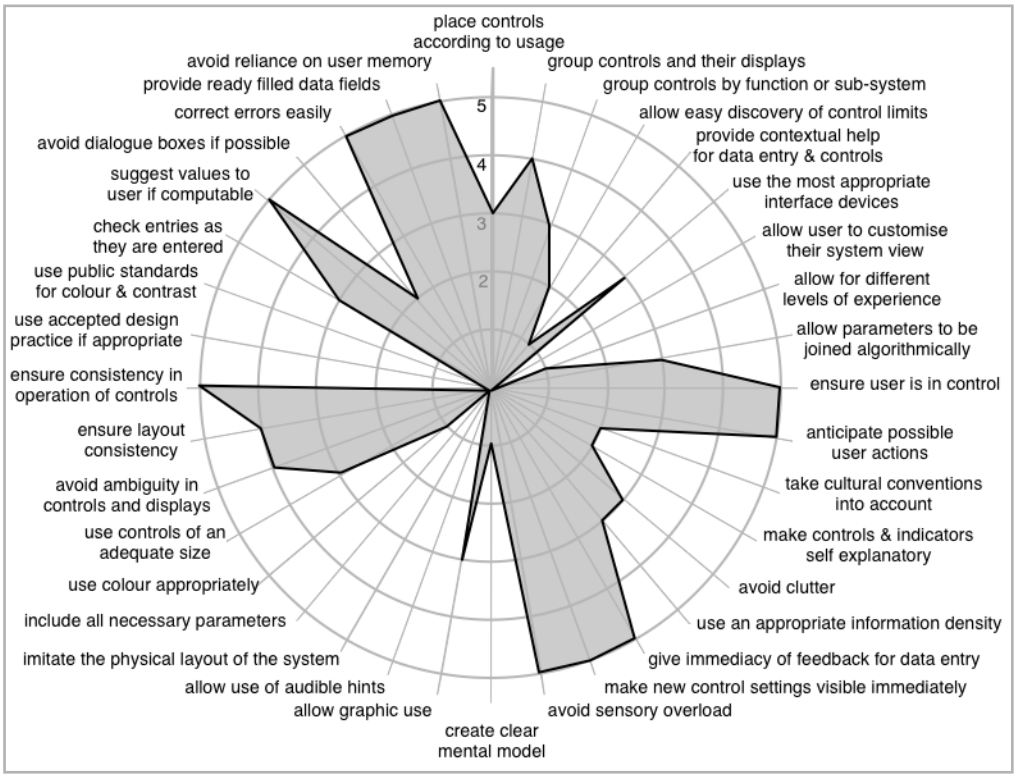


Fig. 8 Solar-B EIS Science Planning Tool (2006) Interface Results

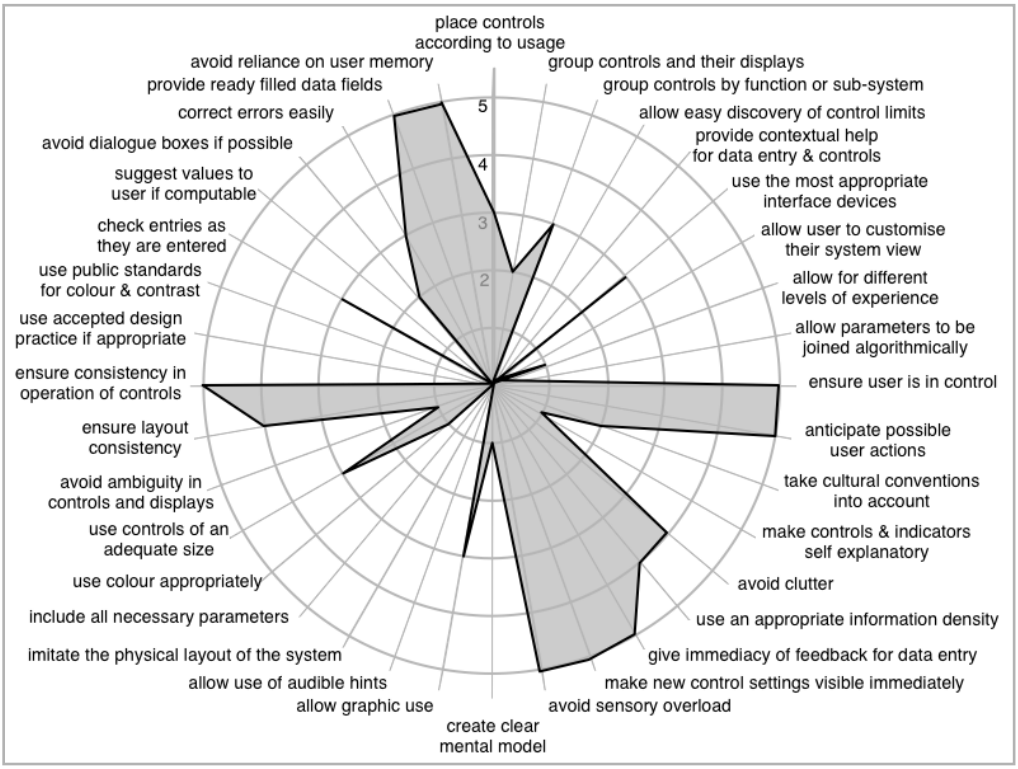


Fig. 9 SOHO CDS Science Planning Tool (1995 - 2005) Interface Results

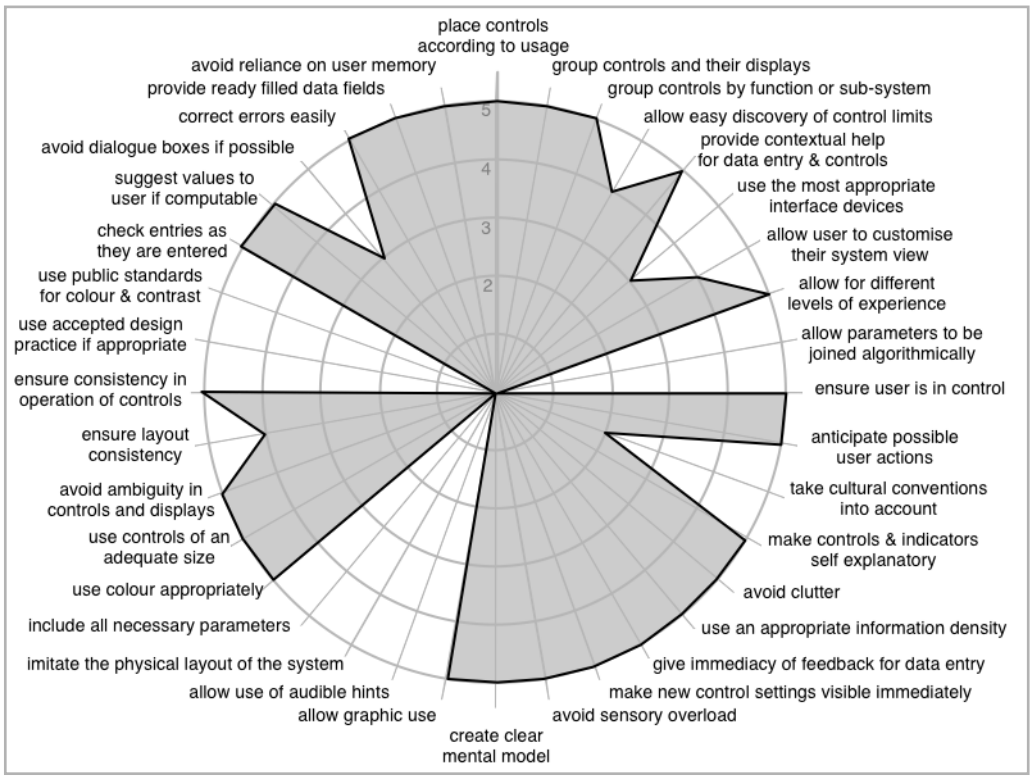


Fig. 10 Hubble VTT (2001 - 2006) Interface Results

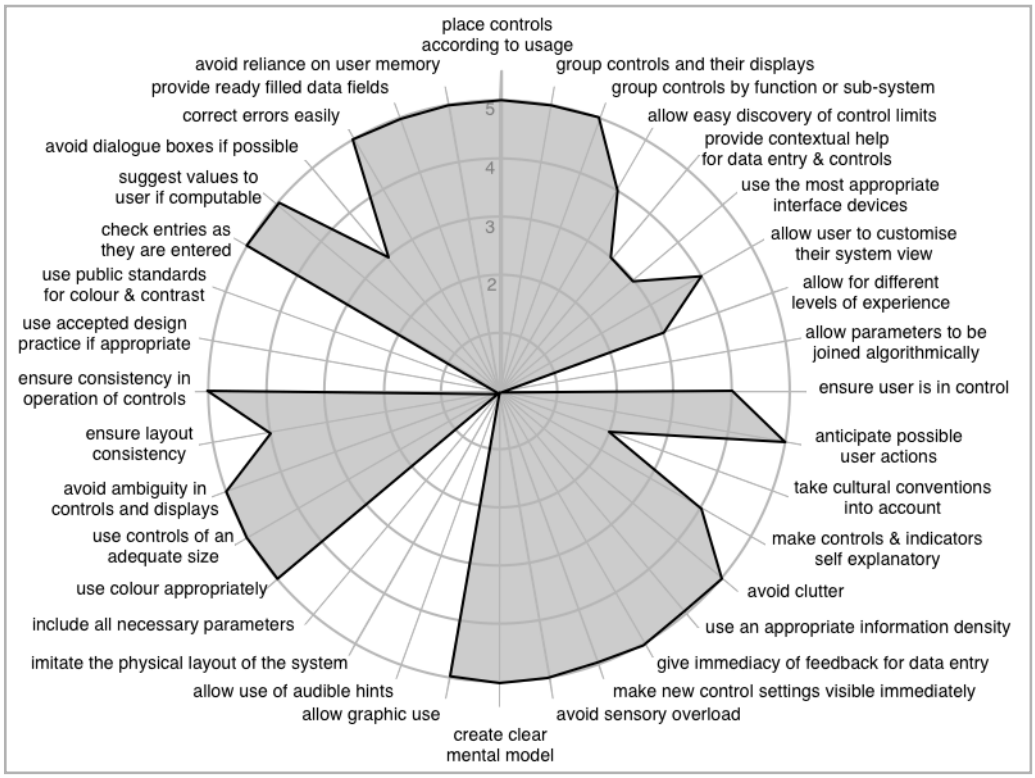


Fig. 11 Gemini Position Editor Science Tool (2001 - 2006) Interface Results

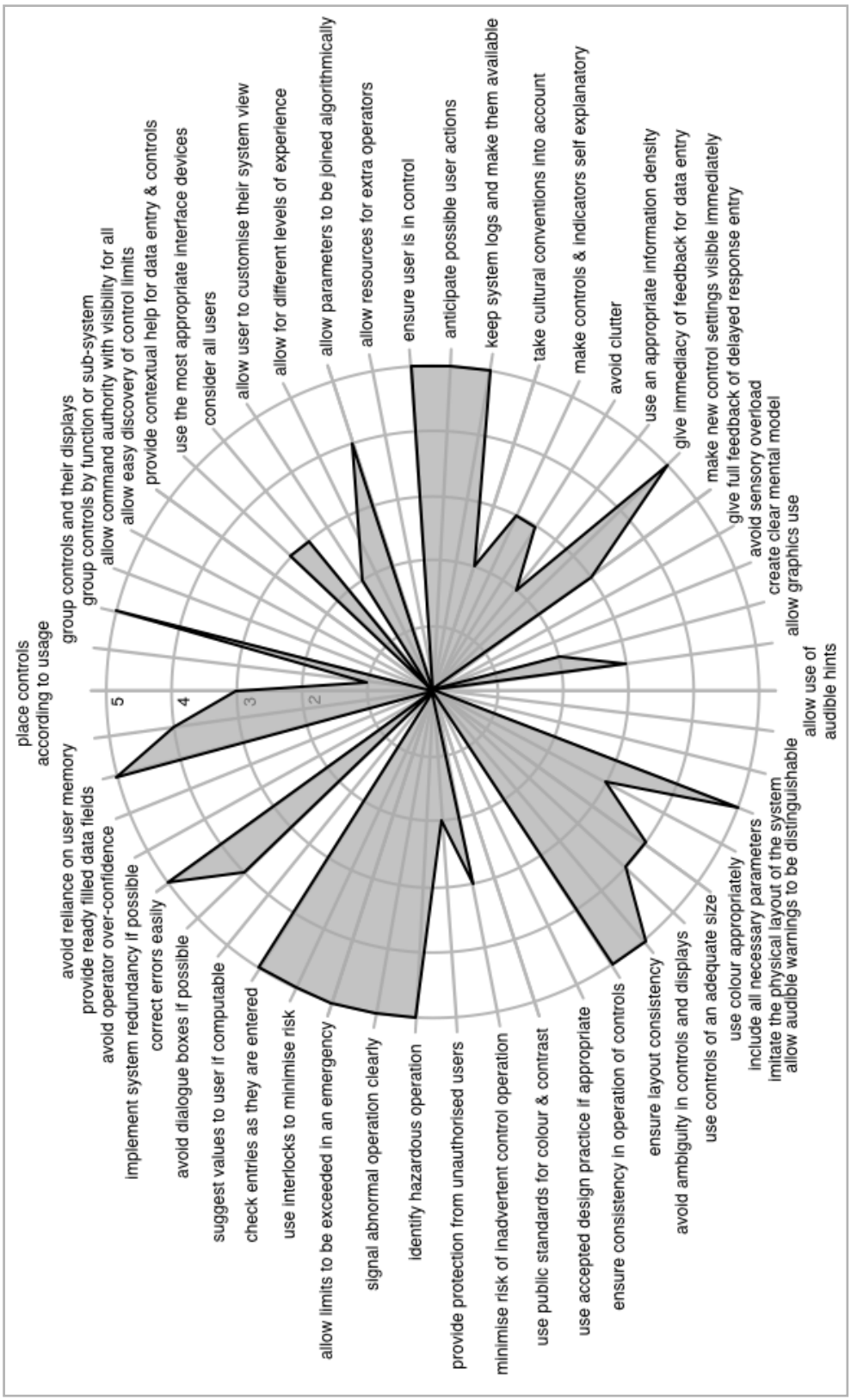


Fig. 12 Solar-B EIS Engineering Interface (2006) Results

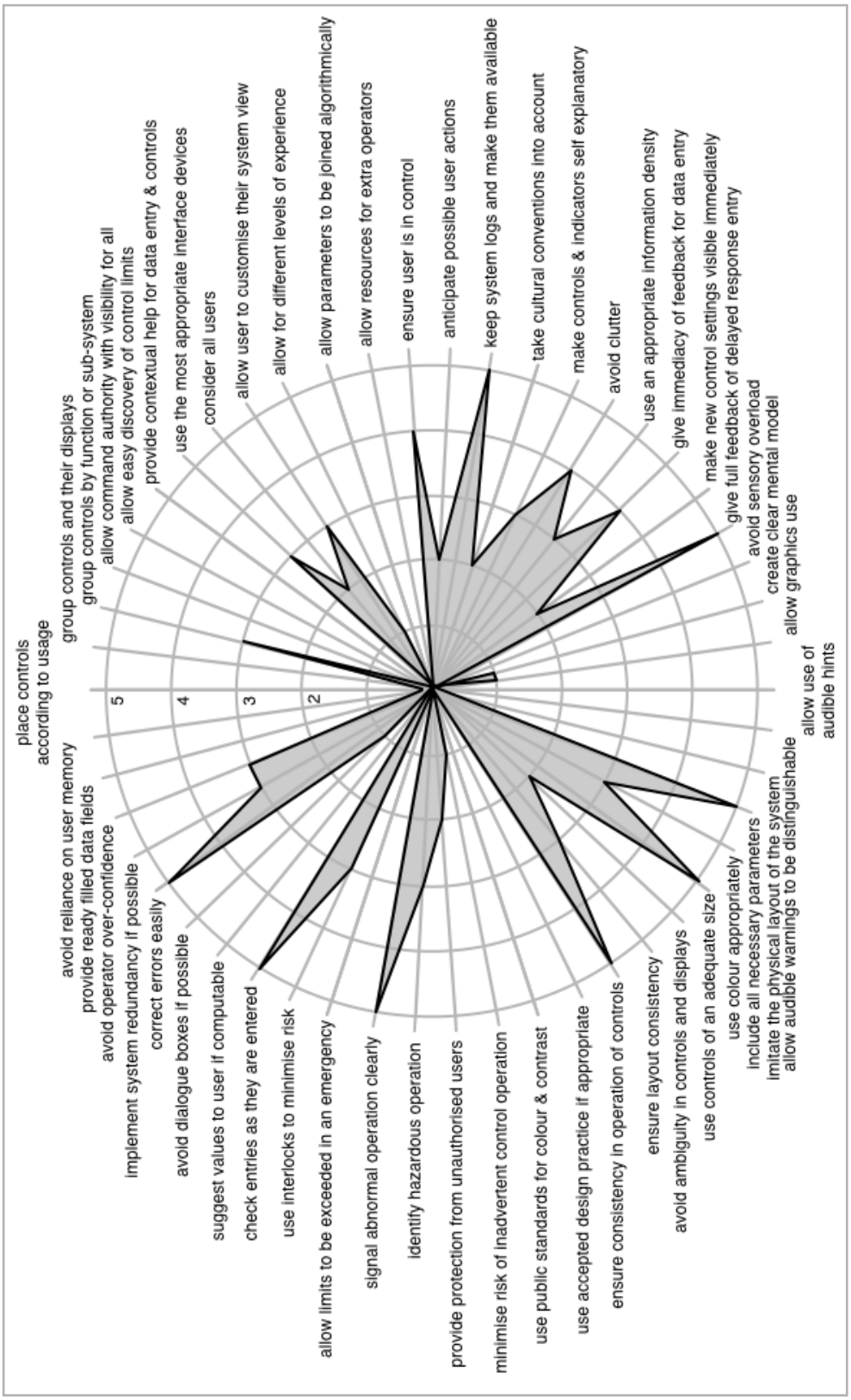


Fig. 13 SOHO CDS Engineering Interface (1995) Results

5 Conclusions

The method of testing the proposed usability criteria against six existing interfaces appears to confirm the hypothesis in section 1, in that they present a means of assessing an interface in the space science context and ranking it in comparison to another. They are not put forward as approaching perfection, but they do present a starting point. Continual refinement by using them in diverse situations and applying modifications should enhance their accuracy.

The polar plots present a novel method of quickly comparing two functionally similar interfaces against each other. Some standardisation of assessment criteria would be necessary before such an approach could become widespread.

Chapter Six: A Common User Interface?

Previous chapters show the analysis behind the design of a user interface for space science. This chapter examines the concept that a high degree of consistency for the user interface might be achievable across a range of otherwise unrelated astronomical instruments, leading to the idea of a common user interface. In particular it includes a detailed synopsis of related recent work on the 'Scientist's Expert Assistant' and the 'Science Goal Monitor' at the Goddard Space Flight Center (GSFC) and at the European Southern Observatory (ESO). That work is then analysed to see what might be learnt from it to assist the realisation of a common user interface for future space instrumentation.

1 The Holy Grail

Many astronomical instruments, both ground-based and those intended for space, have operations in common. There may be for example mechanisms to point the instrument, procedures to make an exposure, and protection systems to avoid accidental dangerous levels of illumination of the detector. There has been some recent collaboration between institutes or design teams to give a limited degree of uniformity to the manner in which a user performs these operations, although each instrument tends to have a bespoke interface designed for it.

To take an example of different practice from the commercial world, an aircraft manufacturer will try to standardise the flight deck layout for their own designs. It gives advantages such as economies of scale in manufacturing, reduction of the maintenance inventory and much reduced design costs. To their customers, the airlines, it also means that crew trained on one aircraft model can fly different models in the fleet with minimal retraining. The aircraft may be physically rather different, but the presentation of the controls and instrumentation allow the crew to become familiar with the aircraft quickly. The airlines see this as reduced costs and therefore greater profitability.

An astronomical instrument tends to be built as a one-off, unlike aircraft, and with the goal of optimising its performance rather than minimising its cost. If a way of controlling such instruments in a common manner led to trade-offs, these might be acceptable if it were simply details of presentation. It would be unlikely to be satisfactory if these trade-offs led to the instruments being reduced to the lowest common denominator of performance.

It seems that a level of abstraction from the actual instrument may be called for. For example, instead of having direct visibility of the mechanics and controls of a pointing mechanism, the user should be able to enter the desired coordinates and the instrument

would find a route there. In doing so, it would take into account hazardous operations and mechanism limitations. This is a concept of operation that is *goal seeking*, rather than *task oriented*. It implies considerably more work in producing robust software than has been typical for instrumentation, but the reduced operational cost due to the reduction in the technical support necessary should make it economically worthwhile. At the same time the performance is likely to be enhanced; any trade-offs should be careful to ensure no degradation in performance. Current and future missions should be in the position to benefit from sufficient processing power to ensure that any instrument is not limited in this respect. The bonus is that the actual details of the instrument implementation are not normally needed by the user, and therefore the same user interface design can service multiple instruments.

Some specific points can be made about consistency:

1.1 Natural Areas for Consistency

- Use the same software tools for as many instruments as possible.
- Use a common appearance for icons, and use same colours and size for screen objects between different instruments.
- Use the same custom graphics for decorations and names.
- Use a consistent screen placement for common screen objects.
- Use a standard method of enabling use by a user who has difficulty with a mouse, or who has a deficiency with colour vision.
- Use common software building blocks to reduce application developer effort.

1.2 Benefits of Consistency

- Familiarity between different instruments for users.
- Standardisation minimises the learning time of a user for a new instrument.
- It reduces the risk of incorrect operation.
- Reduction of the design and programming time for the user interface as some decisions are already made, reducing discussion about endless fine detail.
- It allow re-use of the software written for the interface, and consequently gives reduced software support costs.
- It reduces barriers to dissemination of ideas and makes cooperation between different groups easier.
- It makes verification more straightforward.

1.3 Drawbacks of Consistency

- Work has to be put into defining and policing standards definitions, which implies a body with sufficient international authority. The alternative is to be outstandingly good and lead by example. The problem is part carrot, part stick.
- Can threaten the emergence of new ideas. There needs to be a method of ensuring democracy in the developer and user communities to ensure new ideas are heard.
- Any level of standardisation implies resistance to change and the risk of failure of natural evolution.
- Extra layer of organisation required.
- Need to be able to convince others that the work is worth the gain.
- The interface may grow to be more complicated than otherwise needed.
- Every additional constraint generally makes it more difficult to achieve a high level of optimisation for any given instrument.

1.4 Industry Comments

Human interface consultant Jakob Nielsen in 1989 coordinated a set of papers from industry experts on the theme of consistency in user interfaces, which was published as a book (Nielsen 1989). It included many general purpose points that were additional to those above, such as those paraphrased here:

- If a good degree of consistency is achieved between different applications, then there is a greater burden on the developer to keep a good sense of consistency throughout the whole interface. The user will come to expect it, and will not be so vigilant in being aware of inconsistent operation which might lead to unwanted events.
- All the organisations involved, and their managements, must implicitly support the idea of consistency, as its implementation may appear to put more work in the development programme. In reality, it may actually save costs.
- Some inconsistencies may be allowable if they actually enhance usability, and conversely a slavish adherence to consistency may impair usability if it takes precedence over users' tasks.
- When examined by the author of the article, several examples of software writing showed the user interface to be as much as 35% of the total application. It illustrated the fact that the potential work in achieving consistency, and also the gains to be achieved, are not trivial. Modern software development tools allow the repetitive bulk of work behind defining a graphical interface to be automated, reducing this percentage somewhat.

- Many examples of software writing show inconsistencies between the application itself, the interface and the documentation. One contributing author, Gary Perlman, points out that all three items are the same information represented in different ways, and that one way of responding to this is to use a relational database to hold the results of the original problem analysis. The application, user interface and documentation are then built using the database parameters. Any changes are implemented in the database and are correctly reflected in all the built products.
  This approach would enhance the traceability and consistency of the finished products, although it requires considerable discipline in setting up the methodology. It may particularly aid user interface development where the required result is uncertain and a "rapid prototyping" approach is used to give flexibility.
- The philosophy is one of "design for redesign".
- "There are no simple answers, only trade-offs" (Norman 1983).
- Consistency principle: Do not deviate from published guidelines unless that change will *clearly and distinctly* improve the performance of the product while causing the least possible confusion to the user.
- Style guidelines must be present at the start of software design, not later.
- Plenty of user testing and face to face meetings are essential for interface design.
- To be 'consistent' implies being predictable, dependable, habit forming, transferable, and natural.
- To assist consistent programming, ensure user interface code is separate from other software.

These are all general purpose guidelines for consistency between implementations of interfaces in non-specific contexts.

2 Research towards Consistency

Work from NASA in particular over the period 1998 - 2004 focused on the idea of consistency in the context of observatories, and this resulted in some ideas which are likely to influence space instrument design for a number of years.

2.1 NASA Scientist's Expert Assistant

In 2000 NASA's GSFC reported (Koratkar 2000) on a two year project which had set out to make the science proposal preparation process dramatically less time consuming and costly. It was targeted specifically at the James Webb Space Telescope, due for launch in

about 2013, and became known as the 'Scientist's Expert Assistant' (SEA). It used an interactive visual and expert system approach. One motivation for the work was the realisation that there were real savings to be gained by establishing a common core of observing tools across "*all major observatories, ground or space, inside or outside NASA*". In the context of this thesis, this clearly addresses the idea of a common user interface, although the work has been concentrated on the software architecture and means of development rather than the fine details of the presentation to the user. It is therefore nicely complementary to, rather than overlapping, the work elsewhere in this thesis. This section of the chapter sets out to summarise the collection of papers about the SEA, emphasising the parts relevant to the concept of a common user interface.

The work started from the Hubble telescope science proposal process, which was text or form based, had a significant learning curve and lacked any form of embedded documentation. The goal was to produce a set of tools for scientists to use, each with the following concepts (quoted from (Burkhardt 2000)):

- Easy to use for users of varying degrees of expertise
- Provides reduction in manual support from observatory staff. This implies that the tool decreases complexity of instruments/processes, provides easy access to up-to-date reference information, is flexible and helps guide the user towards completion of that tool's process.
- Allows exploration of the observing parameter space
- Allows visualisation and is as interactive as possible
- Allows documentation to be an integral part of software tools
- Is aesthetically pleasing
- Is common for both observatory staff and observer

In addition, the system "should use artificial intelligence to guide the user, the user interface should be intuitive, the computing power should be distributable across computing systems, it should be adaptable, it should be capable of integration with other tools, and it should be flexible".

The artificial intelligence aspect was based on '**expert systems**' methodology, with the aim of "changing expert human support from the routine to the innovative and extraordinary". This was in line with the recognition that the financially profligate days of twenty four hour expert support teams were coming to an end, and future operations would need to rely on much smaller general support teams with expert help no longer permanently present but at least a phone call away.

The system was built using Java as the programming language, which allowed platform independence and an object-oriented approach. It made full use of modern interface features such as drag and drop and context sensitive help, and was inherently multi-threaded. It was developed using a rapid prototyping method, with a rapid repeated design, build and test cycle rather than trying to define all features beforehand with more formal and extensive written specifications. The reactions by those involved in using this method were strongly positive. The choice of Java gave excellent library support, allowing fast development of complex features such as networking and image manipulation. Automatic update of the application from a server was also implemented. Java included a robust security model, useful for helping implementation over open networks. The features of the language allowed tools to be built with a very extensible form of architecture, and object orientation was a major contribution to this.

The SEA tools were also written with the ideas of improving code sharing and collaboration between groups (Koratkar 2001), as a prerequisite for implementing a **virtual observatory** (VO). The same tools that might have been used to control a physical observatory were also to be used to extract data from a collection of databases of astronomical information, otherwise known as a VO. To achieve this, the software would have to become more capable and less expensive than that already in existence, single mission use would be unacceptable, and parallel development by related groups would need to take place. Code sharing implied an extra workload defining the programming interfaces and producing documentation to a higher standard, often with little management appreciation of the extra work involved. The concept of 'extreme programming' (XP) was looked at, where flexibility for late change is built into the programme in order to lower the cost of changes. Development is driven by the requirement to test frequently, programmers work in pairs, and the method is not without controversy.

An important decision was adoption of the **Model-View-Controller** (MVC) design architecture, which "describes a way to separate the system's data from the various user interfaces to the data, while ensuring that changes to the data are propagated properly through the application". It gave one model which many views shared, allowing changes to be seamlessly propagated. More details can be found in (Fowler 2006). Parameter changes should be reflected immediately in the displays. The SEA evaluation results (Koratkar 2000) pointed out that sometimes the user wanted to enter several parameters at once without being slowed down whilst the displays were updated after each entry. Overall immediate responsiveness of the tools was seen as important.

In order to help meet the goal of making it easier for a user to create an observing proposal, it was considered important to embed expert knowledge about observing into the actual observing tools. This used an expert systems method, where the knowledge was encoded into a collection of rules or '**rulebase**', and followed an 'IF-THEN-ELSE' structure of a test followed by a decision. Processing was carried out by a 'rule engine'. The rulebase was found to be easier to construct if divided up into smaller 'rulesets'. The concept was that the rules could be written and understood by scientists directly, without needing to change the underlying Java software. In practice, programmers were still needed to write the rulebase and rulesets. In the paper on 'lessons learned' (Jones 2000), the claims that rulebases were easy to build and maintain by domain experts such as astronomers were seen as idealistic. Many experts were found to make use of personal 'rules of thumb', and experts disagreed with other experts. Rulebases were best tested with scenarios, and unexpected results were likely. They did find that expert systems catered well for situations where there were both domain experts and novices, such as in astronomy.

The functional requirements overview from the NGST SEA Design Document (Jones 1998) stated that the "user will be encouraged to express their proposal in terms of the science they wish to achieve, rather than the instrument parameters to achieve that science".

Initially the interaction with the user was in the form of an interview where the expert system would present one or more questions. Depending on the answers given, a path through more questions would be constructed until a conclusion was reached. In user tests, this was not a successful approach as it hindered exploration, often failed to match the route of the user's thinking, and was inappropriate if all the user wanted was to set one parameter.

The second approach scaled the interaction back to no more than an optional "assistant" button, which would bring up a window of information relevant to the context. It did not control the tool operation. Although it worked, the developers felt they could do more.

The third approach was that of '"the helpful observer", part way between the first two approaches. The expert system was active all the time, watched the user's actions and made appropriate comments and suggestions in a text panel on the tool. It did not force itself on the user, and could be disabled completely if required. In the period between the first approach and this the underlying rulebase technology had improved greatly as well, and so the helpful observer was adopted, whilst also retaining the assist button and window of the second approach.

The user interface part of the tool was built using standard screen elements from the Java 'Swing' class (a class is roughly equivalent to a library) and was modelled on the current Microsoft Windows design. It included a tree diagram for navigation, as in Fig. 1, which the researchers admitted that few people liked but nothing better had been constructed. There was no evidence that the design of the user interface took into account ideas such as formally identifying user goals of different groups of users, or of layout techniques discussed in Chapter Four, and such work was probably outside the original project goals.

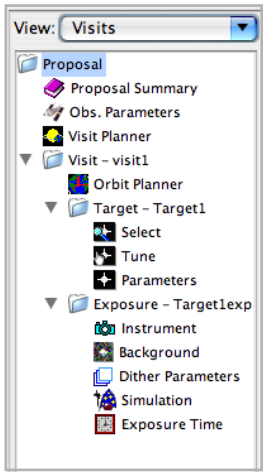


Fig. 1 SEA Tree

An assessment exercise was organised with 15 principal investigators (PI s) of various astronomical instruments, who gave detailed feedback on using the tools. User ranking of guidelines that were useful from the user's perspective are shown in Table 1. A grade of 1 is very important, and a grade of 5 is useless.

Feature	Grade
Easy use - To accommodate user expertise and preferences, proposal information can be accepted in multiple formats.	1.5
User orientation - All components of the system will use terms and concepts which are meaningful to astronomers.	1.9
Responsiveness and speed - Whenever possible, results of user actions will be available immediately. Instantaneous graphical updates when changes are made will be presented to users whenever possible.	1.9
Uniformity - All tools will use consistent terminologies and have a similar "look and feel" to reduce the learning curve	2.0
Scientific feedback - The system will provide information needed to make scientific trade-offs. The impact of a choice will be shown in a meaningful way, and thus users will be made self-sufficient.	2.1
Easy installation - Straightforward web-based installation with a highly portable, platform-independent implementation.	2.1
Interoperability - Tools will be able to share information, alleviating users from having to manually enter and re-enter data and re-process information.	2.1
Common environment - Observers will have access to the same tools environment and configuration as observatory staff.	2.1
Useful documentation - Documentation will be an integral part of the toolset and will be structured to allow efficient access by humans and software tools.	2.5

Table 1 Assessment Exercise Results (Burkhardt 2000, section 6.22)

The PIs found that the tools “allowed them the ability to visualize their scientific needs better and they did not have to concentrate on observatory dependent technical aspects of observing”.

Some existing software tools for astronomical work were reused by building a software wrapper in Java around them. This allowed a continuity of use coupled with the ability to easily swap the old tool for a new one if required.

One of many points realised during evaluation was that users need to be able to pull off networks in one session all the on-line information they might need in order to be able to plan an observation. The astronomical community spans the world and is often travelling.

Continuing interaction between potential users and developers throughout the development process contributed significantly to the quality of the final result. The flexibility of using the tools meant that although some users appreciated it, others were frustrated by no clear sense of direction.

New technologies that were to be considered included real time 3D rendering, as at the time this ability was just reaching the average desktop. Initial plans were made for research into a ‘natural language interface’, where an interview technique between tool and user would allow the user to express their proposal in the simplest form possible. There might be a mixture of natural language processing with a question and answer format.

There was also speculation that high quality tools could be adapted for **education** and **outreach** work. Multi observatory tools were seen to hold great promise for cost reduction, efficiency and quality of observation.

Storage of tool data and tool preference files for the SEA was done using extensible mark-up language (**XML**) files, a simple flexible format used internationally that is both machine and human readable. This idea was taken rather further by Ames *et al*, in discussing using XML and Java for astronomical instrument control (Ames 2000). The software was driven by a description using the Instrument Markup Language (IML), which was derived from XML, and defined using a Document Type Definition (DTD). Using the XML ‘Schema’ standard was a future consideration. It was part of the NASA Instrument Remote Control (IRC) project and was planned for use on the Stratospheric Observatory For Infrared Astronomy (SOFIA) as well as the Spectral and Photometric Imaging Receiver (SPIRE) on the ESA Herschel Space Observatory. The following points are the project goals from the XML paper:

- Provide as much platform independence as possible;
- Create a system that is easy to develop, maintain, and extend;

- Explicitly promote reuse by design and utilize emerging technologies that facilitate software reuse;
- Greatly reduce the implementation time for facility instruments, which must be reliable, robust, state-of-the-art, and easily used by scientists other than the instrument's designers;
- Clearly define the interface between hardware and software engineers;
- Facilitate multiple iterations of the instrument description during design and implementation by means of a software architecture that is readily adaptable to such changes;
- Cleanly separate implementation from description.

The elements of the user interface were dynamically derived at run time from the IML file, and further work was planned to extend this from simple placement of graphical objects in a window. The IML approach allowed deferment of some instrument design decisions during development as the final configuration could also be defined at run time. The data analysis used a pipeline method, and the dynamic class loading of Java allowed this to be defined at run time as well. It conceptually allowed placement of processing at the point of data generation, giving the idea of a smart sensor, and at the point of control, giving the idea of a smart actuator.

There were attempts in the astronomy community in 2000 - 2001 to coordinate further development of this work on software tools. Generally it appeared to stall, apparently due to organisations being unprepared to give sufficient priority to it. One aspect that has worked is the **JSky** project. This started at ESO as the Skycat project using the Tcl/Tk software architecture. Then it evolved into the JSky project, which used several Java classes from the NASA SEA work with the aim "to build a collection of reusable Java components for use in astronomy" (Brighton 2000). Skycat was rebuilt as JSkycat using Java. The ASTROVIRGIL tool for the Chandra x-ray telescope was built on top of JSky and "displays images, spectrums and lightcurves and allows photons to be filtered on sky position, energy or time of arrival". In turn, NASA adopted the SEA tools of Visual Target Tuner (VTT) and Exposure Time Calculator (ETC) for operational use on the Hubble Space Telescope (HST). These evolved into the Astronomer's Proposal Tool (**APT**), which is the method in use in early 2006. It consists of two major components: the APT integrated environment that unifies the individual components and ensures they are interoperable, and the APT tools set. This tools set in turn includes the VTT, the ETC, the Exposure Planner, the Bright Object Checker, the Visit Planner, the Orbit Planner, Starview and the Phase 1 submission form ((Blacker 2000) and the Space Telescope website (HST 2006)).

From about 2000 JSky was further developed as part of the Gemini Observing Tool (OT), which is still in active development in early 2006 (Puxley 2006). The stated intention is that

the OT is to be usable by other observatories. The James Clerk Maxwell Telescope Observing Tool (JCMT-OT) was based on early versions of the Gemini OT (Barnard 2004). The Large Millimeter Telescope Monitor and Control system included the catalogue and image support tools from JSky, which were extended to customise the system.

2.2 NASA Science Goal Monitor

The SEA led also into the idea of the **Science Goal Monitor** (SGM) which had a target of capturing higher level goals, then using automation to translate them into a flexible observing strategy without user intervention (Koratkar 2002). For example, an existing observation might have defined tasks of exposure time, target and filter, whilst an appropriate high level goal might have been “to observe a particular target to give a certain signal to noise ratio”. Autonomy of the systems on board a space instrument was seen as important as future missions were liable to generate more data than could be feasibly downlinked. On board selection, prioritisation and compression would need to be used. Autonomy should allow the observer to pre-define the actions of the instrument according to the quality of the data, just as hands-on use of a ground telescope would allow decisions to be made as the data was received. Appropriate system software for space instruments was already available at the time in the form of embedded Linux and Java 2 Micro-Edition, and development of these products has continued since.

Maths markup language (**MathML**) was seen to provide a means of communicating equations between machines and allowing equations to be readily displayable.

The inbuilt security mechanisms of Java opened up the possibility for scientists to edit on-board software for data processing without the risk of affecting critical instrument systems.

The approach allowed different mission needs. For example, observations for Earth science could be ignored when the target was obscured by clouds, or for solar science could recognise a flare and reconfigure to track it. It gave the observer, not the operator, the opportunity to specify contingency plans.

The tools created for the SGM also appeared to be very appropriate for archival analysis, supporting the idea of a virtual observatory.

Two follow up papers (Koratkar 2004a) and (Koratkar 2004b) discuss some success in prototyping these SGM ideas for dynamic scheduling on the Small and Moderate Aperture Research Telescope Series (SMARTS). Further work was carried out with the NASA Earth Observing-1 (EO-1) satellite and the Earth Observing System Aqua/Terra satellite with the Moderate Resolution Imaging Spectroradiometer (MODIS) instrument. This involved autonomous detection and reconnaissance of forest fires, floods and volcanic eruptions. The

original user interface for this work used visual programming concepts but whilst it was very flexible it was found to be too difficult for science users to construct science goals. The project moved to the idea of adjustable observation templates, which were much easier to use with the drawback that some science goals were not covered.

Interviews with scientists were used to capture domain knowledge and then develop it into a story of how the observation was carried out, allowing for many alternative branches to cover various criteria. Scientists were found often to disagree on the same procedure, and these differences were reconciled by a mixture of discussion and of provision of variable parameters in the template. There was a need to strike a balance between automation and manual operation. As much as possible was automated, but the human was able to interact with the control loop and intervene if necessary, such as to make a subjective decision on the scientific worth of an observation. This allowed scientists to be confident that the automation was only replacing the tedious parts of the job, rather than replacing the human.

Concerns over data handling were found to vary by subject area. Astrophysics is concerned with vary faint objects and scientists are typically insistent that the raw detector data are downloaded to ensure the processing of every last photon. Disrupting an existing observation for an autonomously selected new one would require great care in defining the criteria, as it might wreck hours of valuable observing time. By comparison, Earth scientists have very large amounts of data, and are much more amenable to dynamic observation strategies.

Autonomous systems were found difficult to graft onto existing systems not designed with autonomy in mind.

The SWIFT mission, launched late in 2004, included some SGM ideas in the design. On receipt of a gamma ray burst, the mission would use a fixed on board analysis to obtain a figure of merit for the burst intensity. If appropriate, the satellite would autonomously slew to make an observation of the burst afterglow. Human intervention would have meant that the response time would have been too slow and the observation wasted.

2.3 Other Methods

The use of Java in the SEA and JSky programmes lead to the start of one particular library of useful astronomical tools. The SOuthern Astrophysical Research telescope (SOAR) took a different route and used the commercial **LabVIEW** package from National Instruments for instrument control tools. However, the SOAR Observation Planning Tool (SOPT) was not integrated with the control tools, but was based on the Gemini Observing Tool described earlier. The South African Large Telescope (SALT) followed the encouragement of SOAR

and used LabVIEW as well just for instrument control. SALT used its own generation of tools written in Java for observer planning.

The solar physics area has typically used Interactive Data Language (**IDL**) from Research Systems Inc. (RSI Inc. 2006) for the construction of planning tools.

Various space instruments have used computer languages such as 'C', 'C++' and ADA for instrument control, with little evidence that there has been any degree of consistency or code re-use sought.

3 Analysis

3.1 What Can We Learn from the Scientist's Expert Assistant?

The SEA design is described in sufficient depth to be able to extract many useful guidelines for a common user interface for astronomy. The SGM is described in reasonable detail too, whilst the detail available for the other tools is limited. As several of the later tools have a strong heritage from the SEA and Skycat (ESO) work, the limited detail should not be restrictive. The SEA work is particularly good from the viewpoint of user interaction with the tools, which is very relevant to this thesis.

The initial impetus was financial, as it was perceived that funds available for instrument support personnel would become more difficult to find. This is likely to apply to all branches of astronomy, making it a worthwhile route to follow. The original work was targeted in concept at both ground and space based instruments. Apart from the Hubble Space Telescope with the APT, most users of the SEA work seem to have been ground based observatories. Public outreach and education fields do not seem to have benefited yet, either. This may simply reflect flight instrument timescales and possibly better publicity in the ground observatory groups. It may also reflect a closer working relationship between scientists and technologists for ground based instruments compared to space instruments, but this is only speculation. It may also show how difficult it is to install a common idea into the working practices of disparate organisations with different languages which are distributed around the planet. Any grand solution based solely on technological expertise is unlikely to succeed widely; the technology is necessary but not sufficient. Organisational strength will be necessary too.

Both a space instrument and a ground based instrument that is remote controlled appear to have very similar requirements for a science planning tool, as the costs of an operator locally correcting any problems are high, if not actually unaffordable. This consideration seems to blur any distinction between tool requirements for the two cases. A tool specified for an

hypothetical space instrument should be valid for the less demanding case of that same functionality when ground based.

A number of **concepts** seemed to be particularly appropriate choices, according to the SEA papers' authors (Jones 2000), (Burkhardt 2000). In no particular order these included:

- Use of visualisation in the tools, illustrated in Fig. 2
- Ensuring that the tool was immediately responsive to user input
- Documentation as an integral component
- Allow exploration - the tool should be discoverable
- Use of Java as a code base giving extensive libraries, platform independence and object orientation
- Adoption of the Model-View-Controller software architecture
- Use of an expert system for user guidance
- Use of rapid prototyping as a software development method
- Evaluate work in progress with potential users on a frequent basis
- Express the proposal in terms of science wanted, rather than instrument parameters to be used
- Same tools used by scientists and by technical staff

To elaborate on these in more detail:

Visualisation was a major change from an electronic form filling approach, allowing better communication between tool and user. A visual approach allowed, for example, clicking and dragging to define parameters of the observation.

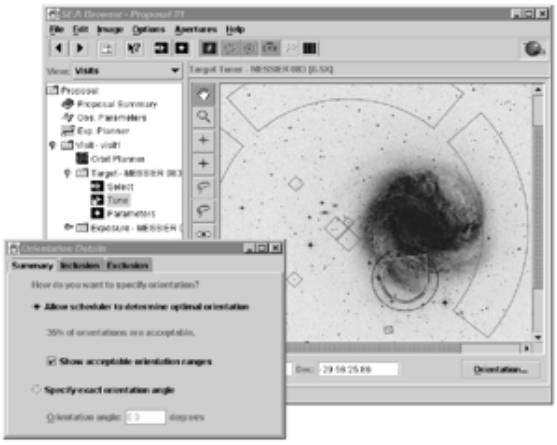


Fig. 2 SEA Visual Target Tuner
(Jones 2000)

Responsiveness of an observing tool was not elaborated in the papers, probably because the result was taken as desired. Experience shows that a responsive tool allows the user simply to work faster, without multiple recurring delays that add to inefficient use of time. It makes better use of a user's short term memory as well. In certain situations, where multiple parameters are changed one after the other by the user, processing by the tool of

one new entry needs to be interruptible by the next entry, otherwise responsiveness will be lost as the tool refuses new data until the last change is processed.

Integral documentation allowed the documents to be installed with the application, and the context-sensitive implementation saved time in performing lookups automatically. The authors made the point that a lot of time can be consumed looking up items in conventional instruction manuals.

Exploration of the SEA parameter space was ranked highly in the user evaluation testing (Jones 2000). Related to this, the SEA by design did not force users to develop an observation program in any particular order. Some users appreciated the flexibility, whilst others were frustrated by the apparent lack of direction of the tool. The provision of a default mode of working is probably the solution to this problem.

Java as a choice for programming language gave the user community the advantage of usage on virtually any computer platform. Initial plans for deployment as an applet on a web page were discontinued due to unrealistic file sizes, but standalone applications did not pose any such problems. The extensive collection of Java libraries (classes) from both standard distributions and from independent developers significantly shortened development time, and hence potential costs. The language features of object orientation which give an extensible infrastructure allowed new features to be added without the need to modify existing components.

Inherent support for a **model-view-controller** architecture allowed the separation of the data from the user interface software, whilst ensuring that data changes were fully propagated throughout the application.

An **expert system** was implemented in only two parts of the SEA, the observing dither module and the instrument configuration. The reason is not described but speculation suggests caution by those involved, as the lack of implementation may have been due to the amount of work involved in establishing the data behind an expert system. Where implemented, the rulebase was easier to define when broken up into multiple smaller rulesets, each containing a small number of the decision making algorithms. The goal of making rulesets accessible to non-programmers for requirement capture was found difficult, with programmer assistance often needed. It was necessary to run the expert system software as a background task (in its own program thread) to achieve the program flow. The ‘user interview’ approach was too invasive and restrictive, whilst the ‘assistant’ and ‘helpful observer’ concepts both worked. The authors saw two future areas of investigation in expert system technology. The first was ‘enhanced assistance’ based on intelligent but unobtrusive monitoring of the proposal as it is being developed, and adaptable to the particular expertise

of the user. It could extend to a knowledge of the relevant merits of particular observatories and instruments. The second was 'functional' where on demand the expert system could analyse the entire observing plan and provide suggestions for optimising it. At the same time it could check for instrument or observatory health and safety.

The use of **rapid prototyping** as a software development method was regarded by the SEA team as a resounding success (Jones 2000). It allowed the test and evaluation of multiple new visual concepts whilst keeping a sound engineering structure. A loose coupling between specific requirements and the resultant system gave a method of working that adjusted to constantly evolving concepts as new ideas were tried and either kept or discarded.

Frequent formal **evaluation** of work in progress with potential science users was also seen as an important point; waiting until the end of a development cycle was not adequate. The SEA team also brought in the idea of an 'alpha user', a practising astronomer who became a full member of the team. The development team came from a computer science background with little detailed knowledge of astronomy, and the alpha user brought in the astronomical knowledge whilst having little formal computer training. This seems an important part of a 'rapid prototyping' scheme of working, in order to produce the frequent user feedback that the method requires.

Using **science goals** for input parameters implied embedding technical knowledge in the tools. For instance, a user could specify the coordinates for an observation and did not need to know how to control the various mechanisms that would point the instrument correctly. It was realised that this facility would allow the tool to be used across multiple instruments, with only the instrument-specific parts being changed. The interface that the user saw would remain the same, reducing development work and easing the transition for the user from one instrument to another.

Tools that contained embedded knowledge meant that the technical staff could use the same tools as the science users, and also it reduced the need for specialism amongst the technical staff. This opened up flexibility in the deployment of people, reducing costs by allowing a small team on duty at an installation each capable of carrying out most of the routine work. Only for special cases or in emergencies would system experts need to be called in.

3.2 What can be learnt from the Science Goal Monitor?

The SWIFT mission incorporated some SGM ideas about autonomous response and, a year after launch, is widely considered to be a major success. It is just about conceivable that a

manual decision making process could have been implemented, but the round the clock instant response necessary would have meant large personnel and other support costs. Almost certainly it would have been significantly slower, degrading the quality of observations. The automation has been extended to the support teams, with messages sent using the internet and mobile phone technology without direct human intervention.

Central to a generic autonomous observing strategy are sufficient onboard processing power in the hardware, an efficient operating system, and the development of event recognition algorithms. One estimate of minimum processing power is 80MIPS (Koratkar 2002)(section 2.1 for JWST), and a suggested operating system route is embedded Linux with Java 2 micro edition. The processor must have access to sufficient fast access storage to be able to compare detector images against a reference or each other in a short time. The scientist must be able to specify the goal analysis algorithms in near real time to permit the analysis to be modified as the observation progresses.

The longer-term target is to capture science goals using natural (human) language. In 2002 this was perceived to be beyond the capabilities of machine techniques available, and effort was put into capturing a subset of the science goals using visual techniques. Further research in the field of artificial intelligence is likely to make a natural language interface feasible. On-board computers with a choice of personality (Adams 1977) are outside the scope of this thesis.

The SGM concept is to allow users to influence the actual execution of their observation by capturing plans for actions to take if certain conditions occur. For example, the object under observation could be too dim for reasonable data, or an anomaly in the instrument systems could still allow some data to be salvaged.

In a resource constrained situation, the SGM could monitor the data received against pre-defined scientific goals. Good data would be kept, whilst poor data might be automatically resurveyed. This could make large differences to the science return of a mission, and is particularly appropriate where the data volumes are large and any manual checking could take a long time.

Unless costs are constrained by use of automation such as the SGM approach, ambitious plans for missions involving multiple spacecraft and consequent huge data volumes will not be economically feasible. A major task will be to evolve a process and technologies that gain the confidence of observing scientists. They will need to feel sure that use of automation will result in the gathering of more and better quality science data, at essentially no risk that good data will be lost.

In the 2004 paper (Koratkar 2004b) on the SGM, the authors reinforce this point by noting that a frequent pitfall in striving to automate processes in the astronomical community in particular has been trying to accomplish too much too soon. They emphasise that developers have to build both expertise in themselves and confidence of the method in the users, and have approached this problem by automating just the more mundane and static tasks, so that when a dynamic event occurs the scientists have their time freed up to study it properly.

The capture of goals by the SGM using a set of graphical building blocks was found to be too difficult to use (Koratkar *et al* in (Koratkar 2004b)), although very flexible. A set of customisable observation templates were adopted, where the template structure is fixed whilst parameters can be customised. This met with very positive responses from scientists, and seems to be a good compromise between flexibility and ease of use. The templates themselves were based on XML, Java servlets and the Java Struts interface, and generated a web-based form to be filled in. The same approach can be seen in complex commercial software, for example Microsoft Word, where the confusion of program detail is tempered by the provision of multiple templates that allow even unskilled users to produce useful results.

3.3 What can be learned from use of LabVIEW?

The experiences of both SOAR and SALT with using LabVIEW for control and data collection from their telescopes seems to have been very positive (Ashe 2000) (Cecil 2002) (Cecil 2004). Remote control systems have been implemented, potentially demonstrating the use of LabVIEW for flight as well as ground instruments. One limitation in this respect in that a flight application would require a real time system and the software is only available for a very limited number of machine architectures, which need to be built with suitable technology for space flight. Even this lack of availability may be changing with recent announcements from National Instruments about embedded technology. Use of the package for support and observation planning is valid in both ground and space applications.

LabVIEW uses a graphical interface for programming, rather than the text base of most other methods. In use it typically gives a fast generation of basic working software, and an immediate functional interface which can be restructured and finalised later. The graphical representation of the software gives an automatic partial documentation, and it is fast to learn compared to text-based languages. It supports interfaces to software written in other languages, and it is available for most operating systems. Library support for complex tasks such as networking and image analysis is very good. All these comments, and others, are put forward by the authors of the papers referenced above as reasons for being very satisfied with their choice of software for their sophisticated control systems.

None of the papers describing the use of LabVIEW have any details of analysis of user requirements. The work on the SEA and SGM from NASA appears to be the only place where user needs have been taken into account.

3.4 General Points

3.4.1 Evolution

Looking at the evolution of the tools discussed here, from the SEA and Skycat to the Hubble APT and Gemini OT, the GSFC in America and the ESO in Germany were considering observing tool design around 1998 - 2000. Some common parts were used in the ESO JSky project, which in turn donated parts to the GSFC Visual Target Tuner. But then the collaboration seemed to drift apart again, with the Astronomer's Proposal Tool for HST and the Observing Tools for Gemini and JCMT moving in different directions. Fig. 3 illustrates this.

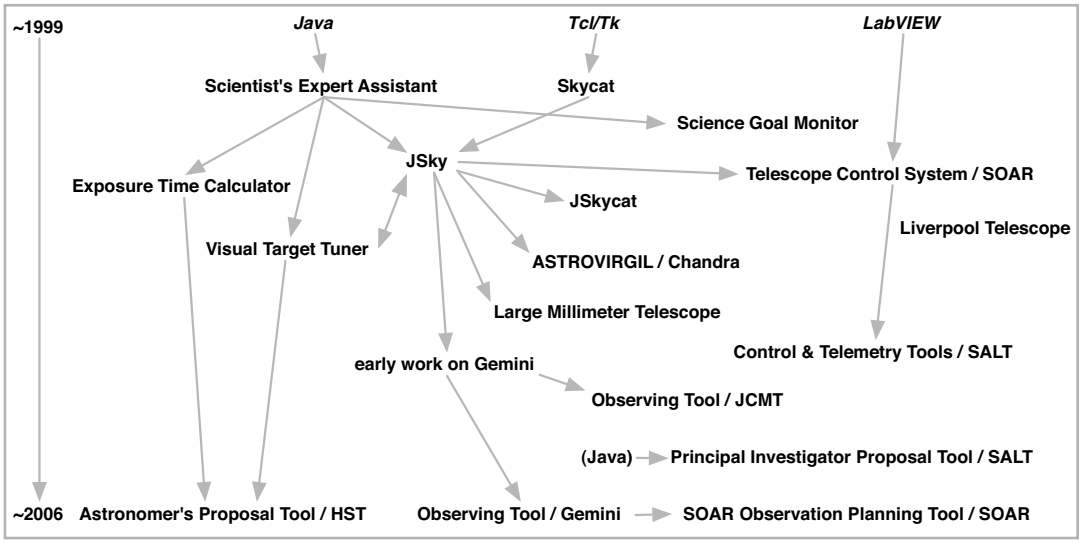


Fig. 3 Evolution of Astronomical Tools

The current tools have a common ancestry, but appear to be diverging in design. Despite this, current versions of the APT for Hubble and the OT for Gemini have some resemblance. The two images below are of the APT (Fig. 4) and OT (Fig. 5) set up for defining a new observation. Larger images of these tools are shown in Chapter Two Fig. 16 and Fig. 18.

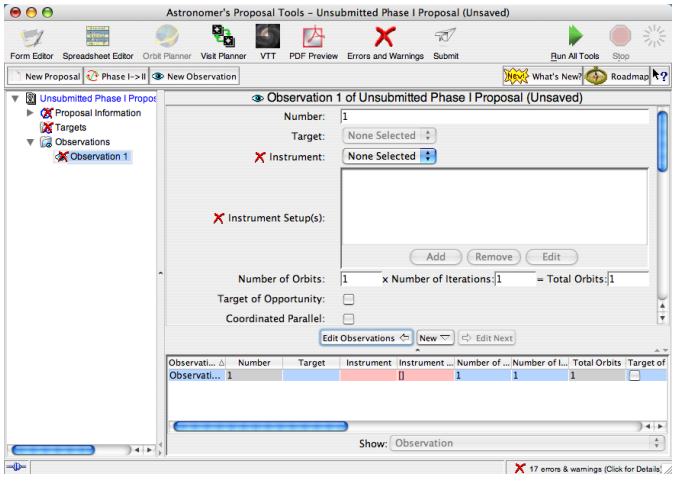


Fig. 4 Hubble APT

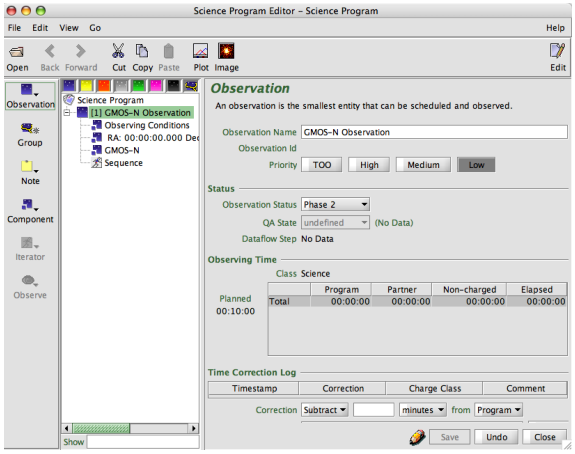


Fig. 5 Gemini OT

The underlying design, rather than the user interface that is seen here, may have more in common. For example, both access groups of on-line databases and require minimal setup to do so. Both the Position Editor for Gemini and the VTT for Hubble have similar image presentation controls that allow different scaling and selection of colours. It is clear that both have a substantial body of work behind them.

3.4.2 User Research

The main body of user research seems to be over the period of the SEA development, with some additional work during the SGM project. These pieces of work give a good foundation, particularly with general insight into the area of interest and the specific requirements of

instrumentation for scientists. The full-time inclusion of a scientist into a computer-oriented development team was probably a large factor in this, and would be worth repeating for future work. The other tools shown above do not appear to have had such a background of user research, or at least any such work has not been committed to publication. Whilst the SEA and SGM authors considered and even interviewed science users and observatory staff, no work seems to have been done in considering use of such tools over the entire lifetime of an instrument, or considering who else might consequently benefit from being considered a user during the design phase of the tool. Examples of such users might be hardware system designers and calibration engineers. These points are referred to in Chapters Four and Seven of this thesis.

The specific design of the user interface does not feature in any of the papers to date. Generally it seems accepted that what is referred to as a “GUI” (graphical user interface) will somehow happen, and that it will consume a significant amount of processing cycles, but there is no concept of rational design of such a creation. This may simply reflect the priorities of the programmes where other aspects were considered more important. One of several opinions in user interface design is that instrument design starts with the user and aims to produce an interface that closely associates with user goals. In other words, the best results occur in instrument design by having a functional interface first, and building the tool functionality as that interface requires. Chapter Seven deals with this topic in much more depth.

One distinct bonus of a common set of observing tools used over a wide selection of ground and space based instruments would be the ease of communication between those instruments. A campaign requiring coordinated observations from a group of instruments with a wide physical separation might be much easier to set up if an entry field or icon on one instrument observing tool mapped precisely to those on the tools of other instruments.

3.4.3 Plug-in Architecture

A common user interface might be better thought of as a set of unique objects for carrying out specific tasks, each of which is an activity often performed in an astronomical context. Such objects might be “make an observation”, or “compute an expected visual result”, and each would form a building block for a more sophisticated interface. This would give a modular interface, where each major component of a display was the product of an individual tool. Each tool would be built as a separate application capable of executing independently, and would easily plug in to a software framework and run as a combined tool if required. A separate requirement would then be to define the presentation of the interface for a given combination of individual tools. Each component of that interface would have a

high degree of consistency with implementations for other instruments, whilst the overall combination of those components would vary to suit the individual instrument.

The very simplistic diagram in Fig. 6 illustrates the point. The user interface is represented by the overall solid rectangle and the individual tool as the jigsaw pieces A to E. Communication between the tools happens through channels, represented by the interlocking parts. Obviously the diagram has its limitations, as for example communications may also happen between non-adjacent parts such as A and E. There may also be multiple channels which would make an accurate diagram clumsy to draw. One of the early advocates of using communication channels between different parts of software was Tony Hoare with his work on Communicating Sequential Processes (CSP) (WoTUG 2003) which gave rise to the 'occam' programming language (INMOS 2002). An interesting development is continuing work at the University of Kent at Canterbury by Peter Welch and Paul Austin who have developed a Java class library implementing the CSP approach (Welch 2004). The process robustness of CSP may make this and Java potential candidates for on-board processing in future space instruments.

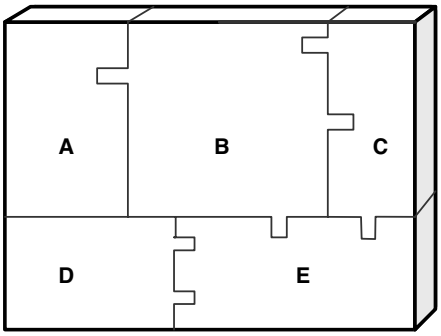


Fig. 6 Plug-in Interfaces

The idea of a plug in architecture is not new in commercial software, although the concept of a single display window containing multiple executing applications is more unusual. Historically, OpenDoc (Apple Computer) was the first commercial realisation of this in 1992 although it was not a commercial success. However, it lead to a consortium, the Object Management Group, being established out of which came the Common Object Request Broker Architecture (CORBA) which is a relevant and currently maintained standard in this field. Peñataro (Peñataro 2000) has a description of the use of CORBA in an astronomical context.

This is in many ways the approach taken with aircraft flight deck design quoted earlier in this chapter. Different aircraft look different but each flight deck has a similar range of displays and controls, deployed according to the specific number of engines, active air control surfaces and other factors. This way of thinking fits well with the role of the developers, as each user's task also forms a well-defined task for the developer. It requires an understanding and definition of the smallest usable interface unit that can be foreseen.

3.4.4 Teamwork

One comment on ways of organising work structures comes out of the SEA experience at the GSFC. The authors acknowledge (Koratkar 2000) the gain in insight from having an astronomer on their team, and also talk about other ad hoc discussions with potential users. There was an advantage in people physically working together. If the field of instrumentation moves further towards an industrial model where the science group writes a specification and a contractor builds the instrument at a physically remote location, explicit care will need to be taken to ensure that communication is easy in both directions. This is particularly true for the software content. Possibly new forms of contract may need to be explored where the specification is flexible to a certain extent whilst the cost is still kept under control.

3.4.5 XML

Ames *et al* (Ames 2000) present some far reaching ideas on the use of XML for an instrument remote control (IRC) framework. Their concept of structured documents supports instrument development from early design through to operations and maintenance, with the target of maximising flexibility and minimising costs. They foresee for example automatic generation of software component interfaces, documentation and graphical interfaces, and an ability to use the structured documents to allow changes to be planned in an incremental manner, rather than a fresh start. It would allow simulations to be developed quickly of any part of the instrument to an appropriate level of fidelity. This suggested approach would probably be a large change in the way of working for many research groups, and that would be a reservation towards the authors' prescient suggestions. For such a method to work well, the whole instrument team would have to be committed to it and would all need to be familiar with XML so that each could contribute to the effort. It would be wise to refine the techniques on a small instrument first.

Any potential implementation of XML would probably benefit from familiarity with the Resource Description Framework (RDF) proposals from the World Wide Web Consortium (W3C). This framework (W3C 2004) suggests a standard means "to support the exchange of knowledge on the Web".

4 Considerations for a Common User Interface

Having looked in some detail at existing work on commonality in interfaces, it is appropriate to examine what broad decisions are required to make use of this work in the future.

4.1 Interfaces

From typical practice, and also as shown in Chapter Four, the interfaces under discussion divide into two groups for any given instrument. One is the science planning tool, used for defining science operations typically by compiling a script that later is added into a composite observing programme. The other interface is that used to directly control specific functions of the instrument, such as a pointing mechanism, and to display the telemetry fed back. The two are often known as the science and engineering interfaces respectively. In typical current instruments they may have very little in common in their implementation as they may be built by people with different backgrounds.

4.2 Ground or Space

Astronomical instrumentation also divides into ground based and space flight units. Earlier we described how it was appropriate to group both together as both were likely to utilise remote control, and so subsequent references to an 'instrument' will simply mean one where physical human intervention is difficult. There is one limitation here, as the ground based instrument will only have significant environmental constraints of temperature, whilst the space instrument will be constrained in mass, electrical power and radiation tolerance as well. In practice this means that space instruments have been very limited in choice of technology. and the processor sub-system may have been a custom design. This would have made it unlikely that a standard operating system such as Linux could have been used easily in the past. Current technology may be adequate to overcome this limitation. The ground based support equipment will of course have no such restriction.

4.3 Software Language

The technology choices for computer language seem to break into three areas. One is the use of relatively low level languages such as 'C' or 'C++' with support from a scripting language such as Perl or Tcl/Tk. This has been a popular choice for space instruments with their technical constraints as it allows good functionality from the available means. The ground engineering support has often been built with the same software language. The science planning tools have typically used a higher level language aimed at numeric processing, such as IDL. The engineering and science packages are likely to have been written by different teams with technology and science backgrounds respectively. This approach has also necessitated the creation of a middle area where science related information can be seen in real or near real time with instrument operation, and known as the 'quick-look' facility. Conceivably this could have been built seamlessly into the operation of the engineering package.

The second area is the use of Java. It has an excellent set of library software both from Sun Microsystems Inc. (the originator) and third parties including JSky in particular. Also embedded is a very widely adopted documentation system (JavaDoc) giving access to consistently structured software comments. Java includes standard libraries for user interface layout, but a third party tool is necessary for this to be done with a 'drag and drop' metaphor. Laying out interfaces with text specifications can be time consuming. Java appears to have established a significant amount of use in ground based astronomy, and a simple inspection of the JSky package shows a large volume of well documented work.

The third area is LabVIEW which has been used for engineering purposes only, and just by SOAR and SALT in the limited survey carried out. The implicit generation of a user interface when constructing a program, and the graphical manner of linking up functions (virtual instruments in LabVIEW parlance), make it easy to learn and become productive. This ease of use was one factor in the decision to use it (Ashe 2000) and ease of reconfiguration was another (Cecil 2002). Like Java, it has good library support. In the examples reviewed, LabVIEW has not been used for the science tools. The reason is not stated, but the reason for SOAR was probably because modifying the Gemini tool gave a common heritage. A recent development is explicit interoperability between IDL and LabVIEW.

4.4 Platform

Languages such as 'C' are not specific about the hardware platform to use, assuming it is well defined. Java requires either a known platform such as Linux or MS Windows to be used, or a 'Connected Device Configuration' (CDC) software framework to be installed as well. This CDC can be a software package from Sun Microsystems that is configured to suit the hardware. LabVIEW requires either a known platform to be used, or an embedded development module used to couple to an unknown platform.

This is a very limited analysis of the situation, and would require experimentation to clarify the trade-offs of one choice against another. What are the likely consequences of the choice of software? Programmer productivity is likely to be better using high level languages such as Java and LabVIEW compared to the other approaches considered, and the main reason in this astronomical application is likely to be the good library support. If huge volumes of embedded processor equipment were about to be produced, an argument could be made for less reliance on libraries and more on writing specific software for those functions. Possible hardware savings might result. This is a weak argument though, and is not justified in the astronomical field where equipment volumes may be single units.

4.5 Common Architecture

What is there to choose between Java and LabVIEW if one is looking for a common approach? Although their approach to creating working software is radically different - one text, the other graphics, for example - both appear to be capable of doing the job. Also, in early 2006, both appear recently to be able to do *all* the jobs of space and ground, instrument and support equipment. One rational approach therefore seems to be to suggest an astronomical instrument software framework, where the functions of the system are achieved by a number - say 50 to 150 - of modules with clearly defined functions and interfaces which are the same whatever language is used. The decision can be taken at the developer level as to which language to use, and the field is left open for other languages to be defined and used in the future. The number of modules is a trade off between sufficient resolution of different functions and the number of objects of which one person can comfortably be aware. This is expanding the idea of a common user interface to that of a common architecture, which would include the interface. Conceptually there should be no reason why more than one language should be in use at any one time. A modular approach also allows any distinction between science and engineering data stream handling to blur as the technology to implement each comes from the same pool of resources.

4.6 Organisation

Any common resource needs a means of organising it. Two models of doing this come to mind. One is that of a library and librarian, which implies one institute taking on a specific coordination role to keep the software modules categorised and updated. The other is that of the ‘SourceForge’ software forum (SourceForge 2006) which provides an on-line framework for collaboration on software authoring for multiple users. One point of contact is still required, but the administrative load should be small. The success of the Open Source software movement illustrates the strength of the latter idea.

As mentioned before, organisation of disparate astronomy groups to achieve a consensus for commonality and consistency will be the difficult task. There will be fixed ideas and the ‘not invented here’ syndrome. By making a start with a small set of research groups who agree amongst themselves and refine ideas, others may be persuaded to join. A major milestone would be the ability to show significant economies over several projects. The terms of reference would include the idea of a common architecture as above, plus the ideas explored in other chapters of a formal definition of the user groups, interface layout guidelines, and the need to plan these for the whole lifecycle.

5 Conclusions

We started off with the concept of consistency in user interfaces and evolved, with the influence of the work on the Scientist's Expert Assistant, to the idea as well of consistency in all the development work that is normally hidden to the user. Consistency in the development process is shown to lead naturally to the much more significant idea of a common architecture for astronomical tools. This would be a step full of inter-organisational difficulties, and might be best tackled in small increments. A natural first step could be common user interface libraries. As explained in section 1.3, there are also risks with following the lure of consistency, with for example care needing to be taken to ensure that new ideas are not lost.

The work on the SEA and SGM in particular has a real wealth of excellent detail on user interaction, based on direct feedback from scientists. Refining the expert system concepts and the other work on capturing science goals has still to be brought to a point where the vision of a natural language interface is functional. The implementations on satellites such as SWIFT and EO-1 showed that autonomous operation could be achieved. In mid-2006, NASA is about to upload a major software change to the Mars Exploration Rover Mission to implement a goal-seeking, rather than task following, mode of operation. Future refinements to autonomy should allow scientists to modify flight software to cope with changing science requirements without any risk to the mission safety. Future space missions are likely to find a telemetry bottleneck as volumes of data spiral upwards and the capacity of radio links only grows slowly, forcing the need to autonomously select, filter and compress data on board. The ideas on XML, with dynamic configuration of space instruments and their interfaces, could imply a sea change in the planning of space science projects.

Many ideas from this chapter are taken forward to Chapter Eight, which explores the notion of a new approach to space science projects.

As ever people are the key to these challenges. Any attempt to coordinate for consistency amongst disparate groups will need superb organisational and diplomatic skill. To implement the ideas on autonomy talked about here, confidence will have to be built in the science community that every last photon will continue to be collected where necessary. The upside is that there is promising potential for releasing funds for more accurate instruments through more efficient ways of working.

Chapter Seven: Use of an Instrument Simulator

1 Introduction

In researching this thesis two concepts kept coming to mind. One was the range of people that have an interest in a space instrument over its lifetime, where that lifetime includes not only the period in active service but also that taken by the other stages such as design and development. *Lifecycle* seems a better term to describe the whole period from gestation of the idea for the instrument, to end of service and possible de-orbit. The other concept was the difficulty in communication that seems to happen from time to time between those user groups. Maybe this should not be unexpected, as the background of someone interested in cataclysmic variables is different from that of the person trying to understand anomalous results in a spacecraft thermal vacuum test. Nobody at all understands the embedded software author. It seems reasonable to speculate that if these people each could operate the instrument in a manner that was close to the final intended configuration, then that shared experience could help establish a common implied language.

In a typical development programme however, no representative instrument exists until at least most of the way through the final instrument build. This might be several years after the start of the programme. Interim and unrepresentative test equipment might be used in the meantime.There might be little consideration of different users and groups and no clear plan for user interface testing. Hence the idea of an instrument simulator for the whole lifecycle was born. It would be a means from instrument cradle to grave by which anyone with an interest in the instrument could explore, test and calibrate its facilities, and by a natural extension of the idea use the simulator for routine control and planning during normal service as well. By promoting team communication, a better instrument might be produced in a more efficient manner.

2 Context

The distortion and absorption of electromagnetic radiation by the Earth's atmosphere leads to a need to put astronomical instruments in orbit beyond it. The development of instrumentation for this purpose requires a structured process similar to that employed for other complex high reliability technologies (Stevens 1998), (Eisner 2002), (Reilly 1993). The space environment introduces hostile factors such as mechanical, thermal and radiation stresses, whilst the resources of mass and electrical power are typically seriously restricted by the available technology. Each instrument tends to be a new design, with possibly a limited heritage from previous missions. Once launched, the opportunity for maintenance is

virtually zero and therefore there is much emphasis on high reliability and on testing in simulated space environments.

A typical instrument begins life with a call by a space agency for mission concepts. A response is made by the science community taking into consideration national and international science priorities, available resources, earlier results and technical feasibility. Following a period of scientific peer review and associated technical studies, a mission objective is defined together with an outline of the scientific payload required to fulfil it. The agency may then issue an 'announcement of opportunity' to which individual consortia of instrument development groups propose payload elements. The successful consortium may then spend several years developing an instrument, qualifying it to ensure survival in the space environment, calibrating it, and finally delivering it to the industrial contractor selected by the agency to provide the flight platform for the science payloads. In this process the developing instrument may go through multiple iterations, trade-offs and refinements. Programme philosophies vary from building a prototype, an engineering model, a flight model and a spare to a minimal approach of technology proving followed by a single 'proto-flight' instrument.

The question of achieving best utilisation of the instrument resources available often does not appear to be considered, at least not before final commissioning into active service. Some work has been done in accounting for and minimising time wasted during tasks such as maintenance and slewing during normal operations, and the observing queue operated for many instruments is typically optimised to maximise the time available for astronomy use. Users might not be given their first choice of observing time slot if moving it allows more efficient operation. Etherton, Rees and Steele (Etherton 2000) have an analysis of such usage ideas applied to a ground-based telescope. There is little evidence of consideration being given to achieving best utilisation over the whole instrument lifecycle. Best utilisation during development for example may give a significant positive influence to the quality of performance of the final instrument. The following discussion suggests that use of an instrument simulator over the whole lifecycle may be one way to achieve this.

3 Existing Practice

The following is somewhat generalised, but checks show it to be representative. Whilst space instruments vary a great deal in their design, implementation and operation, a common approach by a consortium is to break the work down into topic-related work packages which are apportioned to various consortium groups and then to individual people. Typical packages might include system design, hardware design, software development and

environmental test. The consortium then makes a bid to the agency for their instrument to be flown as part of the mission. If the bid is successful, and once the programme is running, packages are typically started as late as possible, to avoid the overhead of employing people to do the work before the situation is ready. Funding agencies plan a spending timeline based on this breakdown and once established there is little opportunity to change this plan. Technical difficulties during test and flight platform integration can also cause funding problems, which can rapidly consume management time. The situation can be described as one of a nominally fixed finance structure with a technical programme that has large uncertainties occurring late in its execution.

As described in Chapter Four section 3.1.2, control of an instrument by a technical operator has become the normal means of operation. A scientist uses a science planning tool to generate a set of requests to use the instrument in a particular manner, and these are passed to the operator. These are merged with other requests, attempts are made to minimise overheads such as instrument slewing, and detailed command scripts are created which are uploaded for immediate or deferred execution. On board data analysis, selection and compression are also defined at this time. Other routine tasks such as calibration and maintenance are interleaved by the operator with the science tasks. Instruments are frequently over subscribed by a factor of up to six times and occasionally more (from personal communication), and so maximising efficient operation is seen as important.

The resultant science data are then telemetered back to a ground network point and eventually held in an instrument file store. The science user may be able to look at the data immediately at a data and operations centre, or the data may be sent to a science institute for analysis there. The current practice for these subsequently-generated ‘data products’ is for processing and calibration of science data to be applied prior to distribution to science institutes.

There are a number of risks in the above scenario that could reduce the science return. Here are a few examples and typical mitigations that are employed at present:

- **Configuration**: The configuration may not be optimal for the science objectives. For example, an inappropriate filter may have been selected.
  Guidelines for configuration can be used to manage this, along with assessment of the observation proposal and the experience and track record of the proposer.
- **Calibration**: The particular instrument configuration may be poorly calibrated, limiting the data quality through unappreciated systematic errors.
  Systematic calibration against standards, cross calibration in orbit and prohibition of uncalibrated configurations can control this.

- **Exposure**: The exposure time may not be optimal. It may be too short giving a poor signal to noise ratio, or too long, potentially saturating the detector and wasting time that could be used for other observations.
  This risk can be managed by attention to the instrument configuration, and by using systematic feedback from earlier observations to maintain an assessment of current instrument capability.
- **Pointing**: Positioning errors may not be noticed leading to wasted observations.
  Detailed checking of proposals by reviewers or operators can reduce the possibility of this happening.
- **Redundant Work**: The observation may be redundant in the light of previous results.
  Careful peer review of the proposal should catch this problem.
- **Data Processing**: Inappropriate post processing of data on-board may remove the very data that was the purpose of the observation.
  This can be controlled by simulation on similar data sets.

Examples are given for the operational science programme, but during the development programme the instrument and support systems are also used extensively. Risks can arise during system level phases in particular. For example, calibration can only take place after virtually all assembly and testing has taken place and this puts calibration quality at risk if there is insufficient time left near the end of a programme. The result is an inadequately calibrated instrument, although that fact may not be immediately apparent. The calibration programme is also an excellent opportunity for the team to become familiar with the idiosyncrasies of the instrument they have just built, and this familiarity may pay dividends in maximising the science return. If the calibration programme is truncated for lack of time, this opportunity may be lost. Another example is that of part way through the build or test programmes for the instrument, its safety systems are often deactivated in order to test it thoroughly, or they may simply not be working fully due to incomplete development. The instrument is left at increased risk of damage. Both sets of risks can be reduced by ensuring operators are familiar with the intended aims and operation of the instrument, and a simulator may be of significant help here.

The length of the lifecycle of some instruments poses real problems in terms of continuity of team knowledge. An instrument might be five years in a planning stage, take five years to develop through to launch, and be in active service for ten years. Over the total of twenty years it is likely that the entire instrument team will change. As well as printed and verbal information, a simulator may have a useful part to play in propagating instrument information

to new team members. It would allow self-paced discovery of the instrument's capabilities without risk to the real item.

4 Simulator Use

Use of some sort of simulator in science planning is becoming more common, for example in judging correct exposures. One of the best so far discovered in researching this topic is the VTT part of the Hubble Space Telescope planning tool, illustrated in Chapter Two section 2.3, which is very sophisticated. The motivation behind the development of this is described in a little detail in Chapter Six. The tools that exist address parts of the science exploitation phase of an instrument's life, but no evidence has been found of any group taking this further to include the whole lifecycle for an individual instrument. Some work has been done in the area of integrating early requirements analysis through to design costings by ESA (Bandecchi 2000), (ESA 2006) under the title of 'concurrent engineering', and aims to produce an environment to enable multi-disciplinary design teams to work in specially built concurrent design facilities.

The suggestion made in this thesis is that a high fidelity simulator that emphasizes user interaction and responsiveness be developed in parallel with any instrument intended for space science use. This approach incidentally could also be valid for other fields of work, in particular where it is perceived as difficult accurately to capture users' goals, but these are not discussed here. The simulator would use common off-the-shelf computers and readily available software tools, and not require facilities any more special than a room with a quiet environment conducive to software development. The fidelity and maturity of the simulator would lead that of the instrument as both are progressively developed, in order to provide a platform for the investigation and proving of design ideas. At the very least it would be designed to:

- Accurately model the scientific response, such as point spread function, spectral resolution, noise, defects and background. The accuracy should evolve with the instrument from a concept to a calibrated implementation.
- Replicate the mechanism functions, and track both consumables usage and mechanical wear.
- Provide formatted data to allow data analysis with high fidelity.
- Have access to astronomical databases for accurately simulating observations.
- Have an effective and responsive user interface.

The reported usefulness of concurrent engineering research can be regarded as evidence for the desirability of a simulator that improves communication and interaction between members of an instrument team. Extrapolating the focus of the work from the early design phase to the whole lifecycle would appear to be a reasonable next step.

4.1 Timeline

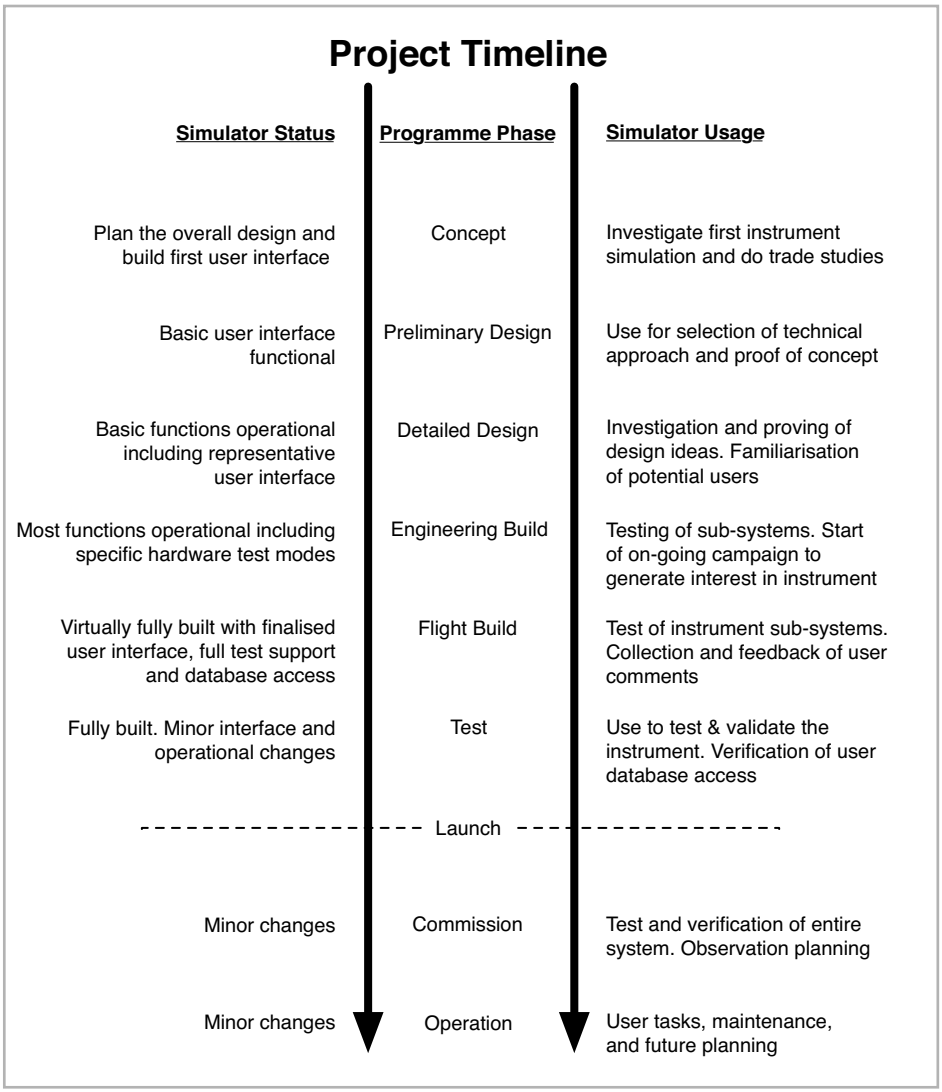


Fig. 1 Timeline

An example of the manner in which a simulator could be developed and used is summarised in Fig. 1. The effort required to achieve this is not trivial and would require a re-consideration of the typical implementation of work packages. In particular, the development programme should be lead by the development of software for the simulator, rather than be lead by the mechanical or electronic systems as has been observed in the past. Predictions of financial

catastrophe by doomsayers should be countered by pointing out the financial benefits of making correct choices and eliminating developmental dead ends in such a programme as early as possible - referred to as a 'left shift' in systems design nomenclature.

Such a simulator would become a liability if it failed to keep track of changes in the instrument, such as changing quantum efficiencies or unavailable modes of operation. The development and maintenance of the simulator must therefore be considered as an essential part of the instrument lifecycle rather than just a simple demonstrator of expertise at the beginning of the programme. The adoption of a simulator strategy implies commitment, resources, planning and the enthusiasm to implement it fully. It requires belief in the concept from all involved, from the funding body right through to the science users. Most importantly, it must not be seen as part of an experiment that can be discontinued if at first it is difficult to make it work or if there are problems in allocating sufficient resources.

4.2 Acceptable Accuracy

Some people will state that they already have a simulator which can calculate the response due to every single photon incident on the instrument detector, and that there is no need for another one. If such a simulator can calculate the effects from a realistic photon flux in real time, whilst at the same time convolving this with the current detector performance, the positions of optical surfaces, stochastic variations of the telemetry stream and have a realistic and responsive user interface, then the task is done. However, with the current and readily foreseeable state of the art and the typical availability of computing power to such a programme, it is unlikely that the simulation of the physical processes involved concerning the photon flux would be achievable in real time. Other parts have rarely been considered.

The goal of the simulator proposed here is that it gives the user a virtual instrument of an *acceptable* level of accuracy in order to decide the trade-offs in operation and performance that are an inherent part of any real world technology. At the same time it will perform actions in a time scale which is comparable to the intended performance of the actual instrument. Radio propagation delays for commanding and telemetry are ignored for this purpose. It will also have a user interface designed to industry principles, which itself is allowed to evolve as it is used in order to eliminate deficiencies. By the time the instrument is launched, not only will it have been thoroughly optimised for performance by many people including those who originally had the vision for it but the user interface will also have been thoroughly debugged, allowing maximum efficiency of use from the first moment of service.

A simple example might clarify the point. Imagine a telescope for deep space imaging. On setting initial coordinates on the simulator, an image from an existing on-line database is

presented to the user. This image will have been modified from the original database image to include the effect of instrument parameters. The user optimises the image to clarify the detail that they need by changing for example the simulated exposure, or filter, or pointing, and this is all reflected on the displayed image in real time. Not quite satisfied, the (simulated) control voltage for the detector is adjusted slightly and by minimising the effects of system artifacts turns a weak image into one full of rich detail. All the simulated parameters track the best knowledge of the instrument performance. The observation is then planned, queued, performed and a result obtained that has made the best possible use of the entire system. Then the final step is that the parameters from this observation are used to update the simulation database for the instrument, to ensure that simulation continues accurately to track performance.

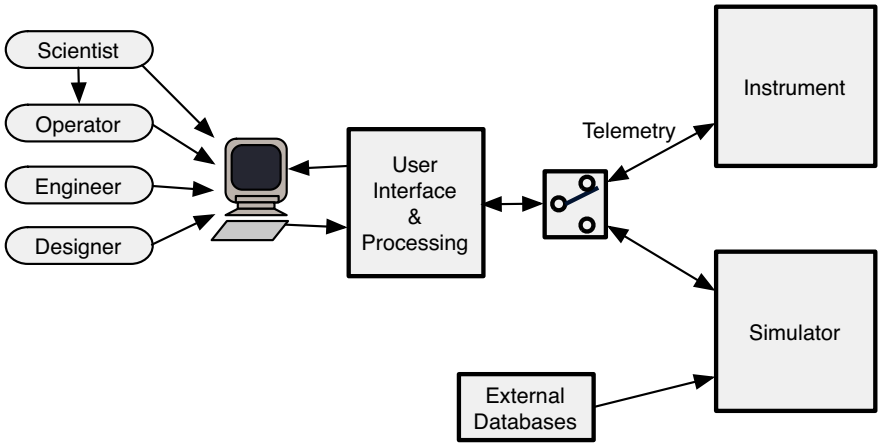


Fig. 2 Block Diagram

The outline block diagram in Fig. 2 illustrates the principal points. A user interacts with some form of display and data entry device. The data in both directions are processed as appropriate. A two way switch then controls whether the simulator or the real instrument is at the end of this chain. The simulator receives its science data from on-line external databases, whilst the instrument generates the data from its optical system. Ideally the user can discern little difference between the two modes of operation, except that sometimes the real instrument gives new and unexpected results.

4.3 Mitigating Risks

The use of a simulator in this manner directly addresses the risks referred to earlier:

- **Configuration**: Operation of the simulated instrument allows the optimal configuration to be iteratively derived.
- **Calibration**: Calibration checks can be handled as a continuously active background task by the simulator. The risk of incorrect calibration is not eliminated completely, but is much reduced.
- **Exposure**: The results of varying exposure time can be seen immediately, allowing deliberate under or over exposure if thought necessary.
- **Pointing**: Once the instrument is calibrated and kept so, pointing errors become less likely.
- **Redundant work**: A comprehensive interface will make use of the available databases to present previous results to check for duplication.
- **Data Processing**: Post processing on board can be simulated precisely on the ground, reducing the risk of data errors.

Once a simulator is developed, the replication cost is essentially zero. By careful choice of software, the simulator can be made capable of running on most computer platforms. Use of a web browser page as a display is another option. There is therefore plenty of opportunity for many potential users to explore the developing instrument and pass comments back, thus extending the scope of the informal review that tends to be associated with anything new. At the operations stage too there are gains. The process of peer review becomes easier, as the reviewers have access to the same tools as the proposer and can vet the proposal more thoroughly. An instrument operator can easily run a final check on the entry of an observing plan before committing it to the real instrument, reducing the risk of unwanted operation.

A potential problem with many instrument programmes is a continuing series of requests for extra features to be added on top of the original specification. This process, known colloquially as ‘specification creep’, is hazardous as specification change late in a development programme has a high chance of causing unexpected consequences unless very carefully analysed. The original reason for a limitation in the specification may be forgotten downstream, particularly if there are personnel changes, and only many months after a change is accepted does that original reason become apparent again. The use of a simulator could help here, as it gives a platform on which to assess potential changes before agreeing them. It also becomes easier to ask a potential user the more demanding “What

are your priorities?" rather than the less specific "What would you like?", as ideas can be tried out before committing to them.

The widespread use of such simulators could radically affect the approach to instrument calibration. Rather than deliver calibrated data sets to the science community, uncalibrated data plus access to a simulator could be provided. The response of the simulator would be managed to reflect the true performance of the instrument at the time of collecting the data. The advantage would be a greater visibility of the calibration process for the user, allowing more accurate estimates of error margins.

4.4 Programme Organisation

There appears to be a continuous tension from a space agency viewpoint as to which style of programme organisation provides the best science return. One option is when instruments are built directly by a consortium of research groups (such as in universities) and supplied to the agency for integration with the space platform, typically by an agency-appointed prime contractor. Another option is that the agency appoints a prime contractor to handle all the interaction with research groups, and instruments are procured under contract by the prime from the consortiums. The reasoning in this chapter is largely from the view that the first option has been taken, as that has been a more typical way of working. There is some evidence recently (in early 2006) that the second option may be in the ascendent. This is not the place to express preference for one or the other as both have strong and weak points, but one consequence of the second option may be that the prime splits up complete instruments into sub-systems which are procured individually. This potentially separates the science users in those research groups from the people implementing the instrument. This may make it more difficult to construct a simulator that leads the design as communication between the two groups in more difficult, ironically at the very time when the simulator is needed more than ever *because* of the potentially poor communication. It may be appropriate for the agency to take the lead and require that a simulator is produced, and then use it themselves to judge the maturity of the ongoing design. At the very least it argues that the consortium must keep a core group of software developers capable of understanding the issues and constructing the simulator. If one also takes the point (made below in section 6) that at a minimum the HCI should be a contractual deliverable, then the dilemma is that of who is placed to be able to do it?

Riggs and von Hippel (Riggs 1994) made an interesting contribution to this area of discussion. In analysing the causes of innovation in general science instruments, they concluded:

> *"We find that innovations with high scientific importance tend to be developed by instrument users, while innovations having high commercial importance tend to be developed by instrument manufacturers."*

This finding may have some relevance at the agency level in deciding the degree of involvement of a prime contractor in the design and manufacture of the science instruments for a forthcoming mission. It implies that the approach involving a consortium of research groups may give the better science return.

5 User Groups

Users are an implicit part of the operation of an instrument, so it's worth looking at the effects the way of working described above might have on the different groups. In Chapter Four Table 5 we show that the user groups actively operating an instrument could be reduced to two sets, giving a total of five groups:

Pre-launch technologist
Post launch technologist
Instrument scientist
Observing scientist
Public network user

In general, system developers - the technologist categories above - are not necessarily application domain experts (space science, in this case), whilst science end users tend to have relatively little interest in the technologies embedded within the systems that they use. This lack of knowledge outside a particular domain is a well known systems engineering problem, illustrated in Fig. 3. The lack of understanding is typically handled through the creation of written specifications, whether it be for example for the optics, or software, or mechanical structure. Formal specifications can easily become too difficult to understand for non-specialists, and there is the risk that they are read without realising that a good comprehension has not been achieved. This path can lead to blame and counter accusation, which does not make for a good team environment. To take an example of the embedded instrument software, it is an axiom that in order to be

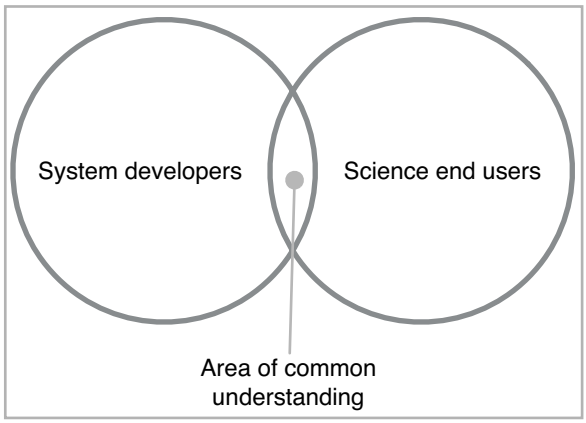


Fig. 3 Understanding between Groups

able to write detailed specifications for software you have to do a significant part of the work in writing a software architecture first. This architecture needs to include details such as memory handling and stack management, but these are details that most end users would never see let alone need to understand. The end user is interested in how the instrument performs, not how it achieves that performance.

The problem can be summarised as one of communication between the scientist, who has a reasonable idea of what is wanted and some ideas of how to go there, and the technologist who has some ideas on what is required and a good concept of how to make the journey. An instrument simulator that pays particular attention to usability should be able to provide a common language between the two groups based on goals, operability and results. Early on in the programme in particular, it may be very unclear how technically to achieve certain goals. A simulator that provides basic functionality at this point is likely to help to bring convergence between the various groups. Only once this has begun to happen should any commitments to hardware be made.

A useful extension to this thinking is to consider the use of simulators in education, particularly at university level. It can be difficult creating the opportunities for students to influence instrument design, as lengthening programme timescales and correspondingly fewer missions reduce the opportunities to contribute. Widespread availability of instrument simulators could re-create these opportunities.

6 User Interface

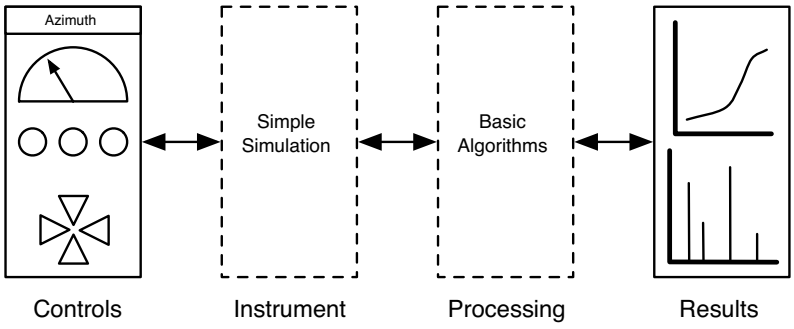


Fig. 4 First Implementation

If good communication is to happen between users and the instrument, and between different user groups, then a well designed user interface is essential. Chapter Four describes a route to achieve this, taking all users into account. The interface is likely to be one of the first tasks to be tackled in a programme run as described in this chapter, and can progressively evolve with the rest of the simulator. As illustrated in Fig. 4, initially the

interface could consist of some controls and displays on a single screen. The instrument and data processing are absent at this stage. There is no need to be able to control anything - what is necessary is to define the manner in which the user communicates with the technology. As discussed in Chapter Two, one particular goal is to establish a good mental model in the mind of the user. The analysis in Chapter Four Table 5 shows that a typical space instrument will have an engineering interface and a science planning tool. Both are regarded as part of the instrument interface here.

The interface needs to be regarded as a system element in its own right in order to be accorded sufficient priority in what can be a hardware dominated world, and to help achieve this needs to be an explicit part of the proposal for the instrument. As Fig. 2 illustrates, it can be the same whether the simulator or the actual instrument is being considered. There is nothing to be gained, and the potential for effort to be expended needlessly, if simulator and instrument were to be different.

Widespread use of simulators in this manner could lead to pressure for some standardisation of interface between different instruments. Chapter Six discusses the concept of commonality of interfaces in some depth.

7 Simulator Generic Requirements

Whilst the design of a simulator as described here would have to be closely tailored to the particular mission with which it was associated, there are several aspects of the design which would be common to many missions. By identifying these areas early in the design process, appropriate scaling and allocation of system resources can be carried out to allow the work potentially to be applicable for more than one mission. The common areas are likely to include:

- general network communications
- specific network protocols to access on-line astronomical catalogues
- database structures to hold command and telemetry databases
- tools for common requirements such as the display of graphs and data trends
- a rigorously designed user interface
- support by most computer platforms

8 Conclusions

Potential gains in utility, quality and science return exist by re-thinking the traditional approach to the organisation of an instrument programme, and it is likely these gains apply

to other fields as well. The lifecycle simulator concept makes use of affordable computing technology to help establish a close rapport between all the potential users of a system, and the system itself, at all phases of its development. In other words, the users are an integral part of the system and the design must take account of this. Real gains will happen when the understanding that this implies brings about spontaneous ideas for ways of solving intractable problems and fires the imagination for new styles of instrument.

Chapter Eight: A Possible New Approach

This chapter takes the material developed in the preceding chapters and recompiles it as a suggested sequence of events for a potential space instrument. A small number of carefully considered additions are included as well. The intention is to show how parameters such as interface standards, user analysis, goals, tasks, future speculation, and use of a simulator can be employed to develop a possible flow for the lifecycle of such an instrument. The material from the previous chapters is then considered again and used to build a list of requirements for specifying and designing the interfaces for a space instrument. References for further information are included for each requirement. The sequence of events and the list of requirements form templates that can be followed - and refined - by groups involved in space science and elsewhere.

1 Sequence of Events

A suggested sequence of a new approach to a space instrument build is shown in outline in section 1.1, described in detail in section 2, and illustrated in detail in Fig. 1, Fig. 2 and Fig. 3. The process includes a division into the phases recommended by the NASA System Engineering Handbook (NASA 1995), introduced in Chapter One. Markers such as **(Fig. 1[a])** are included to link the detailed text and diagrams together.

1.1 Outline

Pre-phase A - Advanced studies

- set mission objectives
- submit bid for mission

Phase A - Preliminary Analysis

- assign roles inside consortium
- initiate simulator work
- decide on software platform

Phase B - Definition

- distribute interface prototype
- start work on science tool
- finalise proof of concept

Phase C - Design

- review team structure
- decide on technological additions
- prototype uncertain design areas

- achieve initial simulator operation

Phase D - Development

- implement full simulator interface
- build & test sub-systems
- integrate engineering model
- build & qualify flight model

Phase E - Operations

- commission & operate instrument
- use simulator for mission planning

2 Instrument Concepts

This sequence of events builds on the historically common model where a space agency solicits and accepts the science instruments, rather than an alternative model for example of industry taking this role.

Initially an agency such as ESA or NASA may define a set of potential mission concepts, based upon a forward look of science goals and likely technological advances in a certain timeframe **(Fig. 1[a])**. This period might be anywhere from five to twenty years. In response to publicity, members of the science community will discuss, modify, and refine these concepts to the point where the ideas are mature enough for a formal 'announcement of opportunity' for an instrument platform to be made by the agency. The change from typical practice suggested in this thesis is that this is an early opportunity for the agency to maximise the usability of the instrument by specifying that a representative simulator, or a well-defined user interface, is a deliverable package along with the instrument itself. This would be following a precedent already set for most missions, where thermal and mechanical simulations are mandatory deliverables. Normally every instrument would have a development programme consisting of a set of progressively more complete models, and the model programme adopted here by way of example has a limited prototype, followed by a full engineering model and then a proto-flight model. The term 'proto-flight' implies that environmental qualification is carried out on the actual flight model.

In response to the announcement, consortia of interested groups apply for funding from national sources. If successful, they then bid for accommodation on the platform, and include details of how they will use a simulator to ensure best instrument design and functionality. The successful consortia then commence the detailed planning of their design and development programmes, and start a liaison with both the agency and the prime contractor on an ongoing basis to set and maintain the detailed technical interfaces such as

the mechanical, electrical, thermal and now also the user. Amongst other written material, a set of formal specifications detailing the technical agreements with the agency is required by the development team. This suggests one ambitious variation of the schedule described so far; given sufficient time, a speculative instrument could be simulated by an interested group early in the process, and the outline design and specification optimised by simulation before making a formal bid to fly on the platform.

2.1 Initial Development

One of the first acts of a successful consortium is then to assemble an initial simulator development team, and to obtain the commitment of the management that a scientist with the relevant understanding of the type of mission is a full-time (or at least a substantial part of their employed time) developer for the simulator. This commitment is a critical conclusion from the SEA team described in Chapter Six section 3.4.4. HCI training should be provided if necessary to all those involved with the simulator development, as it is likely to be a new subject for most.

In considering development of a simulator, reference should be made to the significant library of software available from the resources described in Chapter Six Fig. 3, particularly the JSky Java classes. There is potential here for saving a large amount of work compared with developing everything from new. These resources may drive the choice of software language for the work **(Fig. 1[b])**. At the least, the significant user interface component would seem to mandate an object-oriented language, examples being Java, C++, and LabVIEW. The 'model-view-controller' structure should also be adopted for the reasons in Chapter Six section 3.1.

The general user analysis in Chapter Four provides a methodology for analysing most arbitrary situations involving users and technology. Space science and the control of remote instrumentation appears for most instances to be covered by the general case in Chapter Four section 7.8.1 and Fig. 8. If this is still true for the instrument under discussion, then the existing deduction can also be kept. This means that for the technologists the operator, engineering and quicklook interfaces are sufficient, and for the science user the science planning tool is the required interface. This assumption needs to be confirmed beyond doubt before moving past this stage, and the analysis reworked if not true.

At this point, well before any hardware commitments, the first versions of the instrument simulator and user interfaces are built. Their purpose is to aid selection of the technical approach for the instrument, and to provide a way of establishing a proof of concept, as described in Chapter Seven. They should accept user input, synthesise detector and

engineering data and store science data in the intended mission file format (for example, the 'fits' format). The user interfaces are built using the procedure in Chapter Four section 6.7, which is summarised in Fig. 3 of this chapter as a flow chart. At this stage their level of maturity is adequate to give to a user a general concept of the intended functionality of the instrument, but not so detailed that the work becomes entirely speculative and is completely changed at the next review. The interface will be revised to a major extent on several occasions during the development programme, following the broad idea of rapid prototyping that was highly recommended by the SEA team in Chapter Six section 3.1. By following a clear procedure each time for defining the interfaces, the engineering team will build skills in interface design in a subject area that is likely to be unfamiliar.

Current practice appears to be to use the science planning tool to generate information that can subsequently be read by an operator and manually entered as instrument commands. The opportunity should be taken to generate machine and human readable information from the planning tool, with for example an XML format, to allow a direct connection between planning tool and command interface. Operator intervention will still normally be required but errors should be reduced, and at this early stage in the programme there is time to resolve the issues. It opens up the option of operation direct from the planning tool, which may aid instrument development.

The software should be constructed so as to be readily distributable to other consortium members, using local installers or automatic network update packages. Java is particularly good at the latter. At each installation, modern network ideas should be explored to allow load sharing of the software on different machines and automatic discovery of those machines by each other by 'zero configuration networking' (Steinberg 2005). This latter term is currently (2006) implemented in similar but not identical methods across several different computer platforms.

This first build of a simulator and interfaces is then used by people from as many different disciplines as possible, modifications made in response to comments, used further, and the design iterated until stable. Re-design must involve the full user interface procedure in Fig. 3 to ensure traceability of the work. This point of stable initial design can be regarded as a project milestone **(Fig. 1[c])**.

2.2 First Operations

At this place in the programme an outline instrument exists, albeit as a simulation, with simple but realistic engineering and science interfaces, and which can easily be widely distributed to consortium members for comment and criticism. This is potentially up to

several years ahead of what might happen using current practice, where the first time many potential users might experience the instrument is during testing of the flight model and where many parameters are fixed for life. A review can now be made of the consortium and individual work packages in the light of these simulator results. and decisions made on future team structure and training needs. It is an appropriate point to look critically at the on-board software requirements, and judge whether the same language and even the same tools can be used for both on-board and ground-based system development. There is potential for gaining team flexibility here by only having one system to learn for all the software requirements, encouraging closer and more efficient working.

2.3 Possible Inclusions

The SEA and SGM work from NASA described in Chapter Six detailed several system-level features which were either implemented or recommended. They also noted that in the astronomical community in particular a frequent pitfall in striving to automate processes has been to try to accomplish too much too soon. At this point, with the experience of creating an outline instrument but without commitment yet to hardware and in the light of this caution, it is appropriate to review these topics and decide whether to include or omit them. Some important examples are:

Integrated help system: This was the major goal of the SEA (Scientists' Expert Assistant) work and methods investigated ranged from a simple context-sensitive message about a potential operation, through to an assisted form-fill approach (also known as a wizard), to the complexity of a background task continually monitoring the user input and optimising the whole observing programme. Notably they also found that it was necessary to include the option to turn the help off to avoid it being intrusive. Expert systems are a possible method of organising contextual help, but demand care and expertise in establishing the rulesets that control the information delivery. It might be easy to waste a great deal of time in establishing a working system if the expertise did not already exist, and so the decision may rest on the availability of those skills. Regarding a help system simply as embedded documentation which travels with the software tools, on the other hand, may lead to good returns on usability for the work required.

Science goal templates: The SGM (Science Goal Monitor) work was largely concerned with applying autonomy to space missions, and a major part of that was ensuring the goal of any given observation was determined clearly. After experimenting with different methods, the SGM team concluded that a customisable template-based approach for the science planning tool was a good balance of flexibility and usability. This may well be appropriate even for a mission with few needs for autonomy **(Fig. 1[d])**. The templates were based on

XML with Java programming, and similar template techniques can be seen in commercial software for such tasks as word processing.

Networks and Interoperability: The CORBA communications architecture is briefly described in Chapter Six section 3.4.3 and more fully in a paper with an astronomy context (Peñataro 2000). It is one answer to the question of ensuring interoperability between different software packages, different platforms and over networks. It may be seen as unnecessary for a small instrument, but the issues raised should be actively considered even if they are subsequently discarded.

Some other topics that require decisions early on in a project programme became apparent in compiling this procedure, and are discussed briefly here:

Use of a personal digital assistant (PDA): If use of a PDA as a display or controller to the engineering or science interfaces is proposed, the definition procedure in Fig. 3 must be used to design interfaces specifically for this purpose as well as for a desktop-based solution. The use of a comparatively small touchscreen and stylus is so different in terms of user interaction that it requires its own design. Simply scrolling a large content area backwards and forwards on a small screen, and using a tiny keyboard with a stylus to control it, is unlikely to work effectively. The decision preferably needs making at this point and not putting off until later as subsequent decisions are likely to flow from it. Retrospective decisions are likely to involve significant rework; this is the systems engineering situation of a ‘left shift’ referred to earlier in Chapter Six.

Remote Alerts: The recent SWIFT spacecraft mission has highlighted a system of operating that involves fast responses world wide to unpredictable events - in this case gamma ray bursts. Automated text messages to mobile phones are a key part of this. They represent another aspect of the interface to the instrument, and involve the new aspect of automated response requiring fallback to a second or even third respondent if the earlier one fails to answer. The procedure is the same as with the case of the PDA above, and similarly the decision needs to be taken at this point in the instrument programme, not later.

Language and alphabet: Despite the international nature of many space projects, the English language appears to have dominance for use in instrument interfaces. Other languages often appear to have translations that involve more characters than English and take more room on a screen. Therefore if multi-lingual operation of an interface is intended, layout schemes must take this into account at an early stage. Similarly, other languages may invoke alphabets with different character sets including diacritical marks, or even phonetic characters such as Kanji. Characters may flow left to right, or top to bottom, or a reverse of one or both.

Technological advance: The pace of technological change is such that one should take a forward look at the likely state that computer architectures and system software will be at the time of deployment, rather than just at the time of design.

These uncertainties should be discussed over the consortium and firm decisions made before proceeding further **(Fig. 1[e])**.

2.4 Instrument Design

At the same time that the software planning aspects are being considered, there is an opportunity to start the detailed technical planning of the instrument design. In particular, areas of uncertainty in hardware or software can be modelled or built to prove a concept, and these tasks can be carried out concurrently if the resources are available. The purpose here is again to manage the risks of the project so that the areas with the greatest uncertainty are resolved as early as possible. With a basic simulator functional at this point, as there should be by the previous milestone, there is an extra opportunity to merge the modelled or built areas into the operation of the simulator and prove the combined system. The modelling, building, substitution and simulator design are iterated until the areas of uncertainty are resolved.

The two streams of software and hardware planning can be brought together again at this point **(Fig. 1[f])** and potential users familiarised with the developments, and feedback collected. The definitions of the instrument data streams of commanding, engineering and science are frequently completed late in an instrument development programme as the sub-system requirements take time to think through. In the situation of using a simulator however, most of the functionality required of a sub-system will have been understood by now, and so the definitions of the data streams can usefully be made now and the simulator revised to include the defined streams in its operation. There are frequently problems with the databases holding these definitions due to their complexity, and an early compilation will assist the development programme.

Any extra requirements for the simulator and interfaces are implemented and a full user consultation made. This principle of frequent user involvement is also taken from the SEA experience in Chapter Six section 3.4.4, where it was found to contribute significantly to the quality of the final result. The process is iterated again until a stable definition is found for the design and operation of the simulator with exploratory hardware, and a set of representative interfaces is agreed upon. This is then the second milestone **(Fig. 1[g])**, and is a critical definition of the early framework of the instrument. By using the procedure here, many of the difficult strategic decisions have already been made by now and there is involvement across

the consortium in those decisions. This general approach tackles some of the personal goals of users raised in Chapter Four and particularly those about involvement.

2.5 Engineering Model

The work now can move to the engineering model. This is intended to be a functionally identical version of the flight model, but with few environmental constraints allowing it to be built with off the shelf components as far as possible. Its purpose is to prove the design to allow the flight model to be built with confidence of correct operation as far as possible first time. The sub-systems mentioned earlier are divisions of the instrument at natural separation points, such as a mirror mechanism, or an imaging detector, and frequently are best tested as individual parts before integrating together. Specific test routines are required for this and should be implemented in the developing simulator as this will maximise their availability. There are likely to be common areas of existing software as well, for example the telemetry definition databases, thereby minimising the work required.

The first stage **(Fig. 2[h])** is to take the simulator and interfaces as far as possible towards their final design. Once again using the procedure in Fig. 3, plan the full user interfaces, do mock-ups and check with a range of users. Then implement and code the full interfaces, check with a wide range of consortium members and iterate the process until the designs are stable. Then build up the capabilities of the simulator until as many functions as possible are fully operational. At this time the science data file store can be added to complete the set of ground support equipment.

At the same time as the sub-system test routines are being built, the actual sub-system design can take place **(Fig. 2[j])**, given sufficient resources. Ideally, the design and build process of each sub-system can be done in parallel with another. Once built, the systems are individually tested and then integrated to form the engineering model instrument **(Fig. 2[k])**. Then the simulator can be used to test this integrated model, taking care that if errors are found that they are correctly attributed to the simulator or the instrument build. Once stable correct operation has been achieved, a reference design will then exist for both the instrument and the simulator, and this point forms the third milestone **(Fig. 2[m])** in the project.

2.6 Flight Model

If progress has been as expected, there should be little if any development left to do for the simulator, instrument interfaces, and science data file store, giving a complete and tested set of ground support equipment. It can be used now to support the flight instrument build, test, calibration, environmental qualification, spacecraft integrated test, launch and operation. The

advantage of following the programme detailed here should now become apparent, in that the whole team are familiar with the operation of the instrument and useful service can happen at the first opportunity after launch. A robustly designed and thoroughly debugged interface also means that new observers are able to be productive with the minimum of delay, maximising the utility of the instrument.

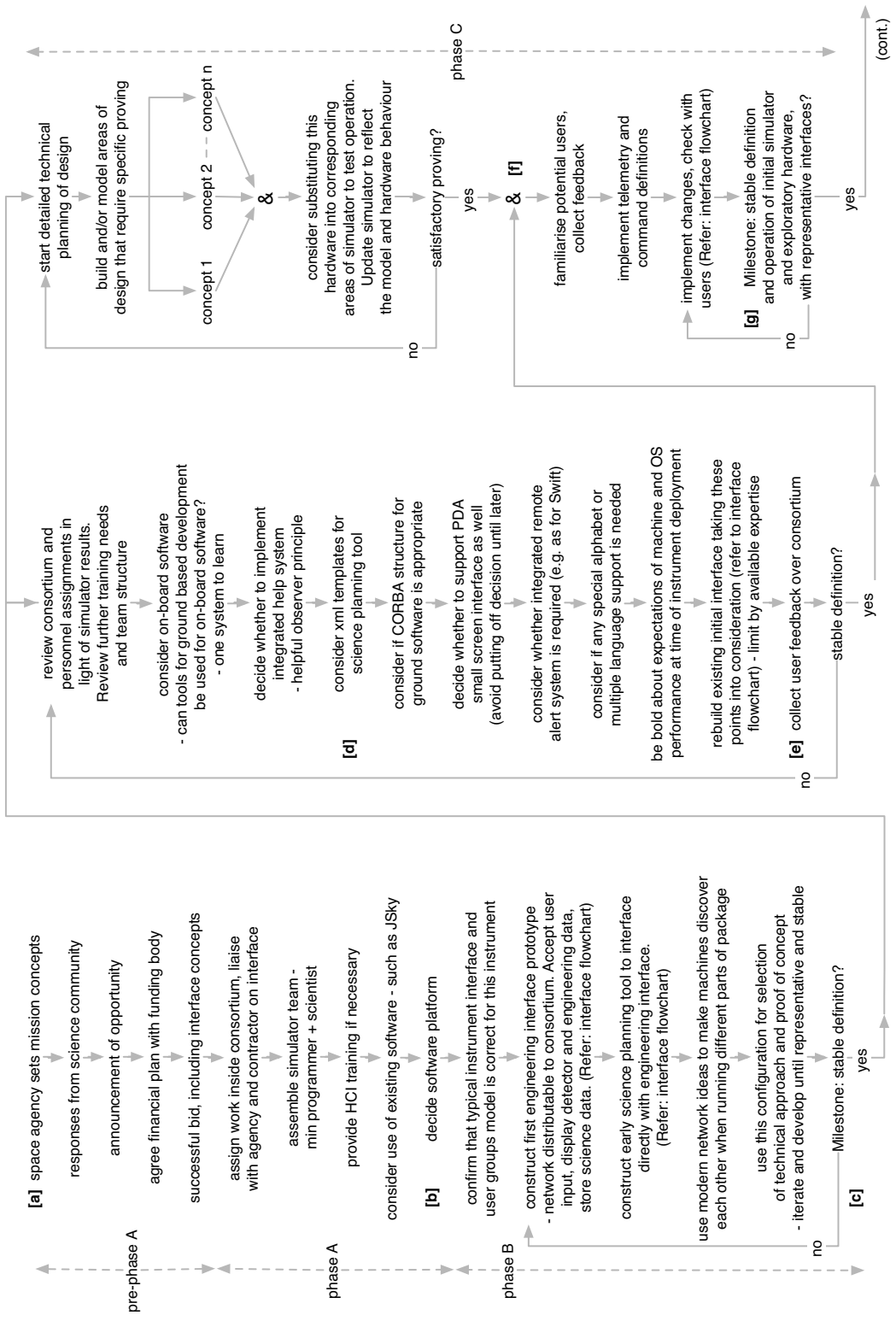


Fig. 1 Flow Chart (part 1)

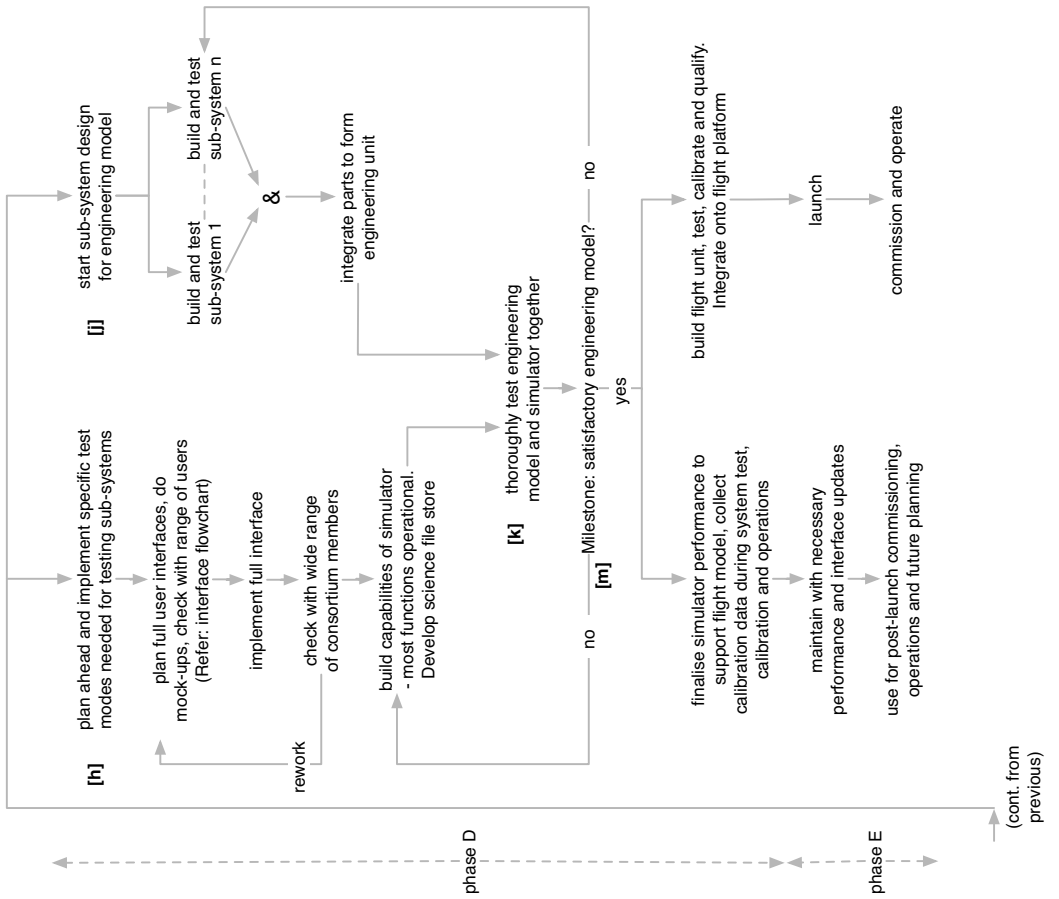


Fig. 2 Flow Chart (part 2)

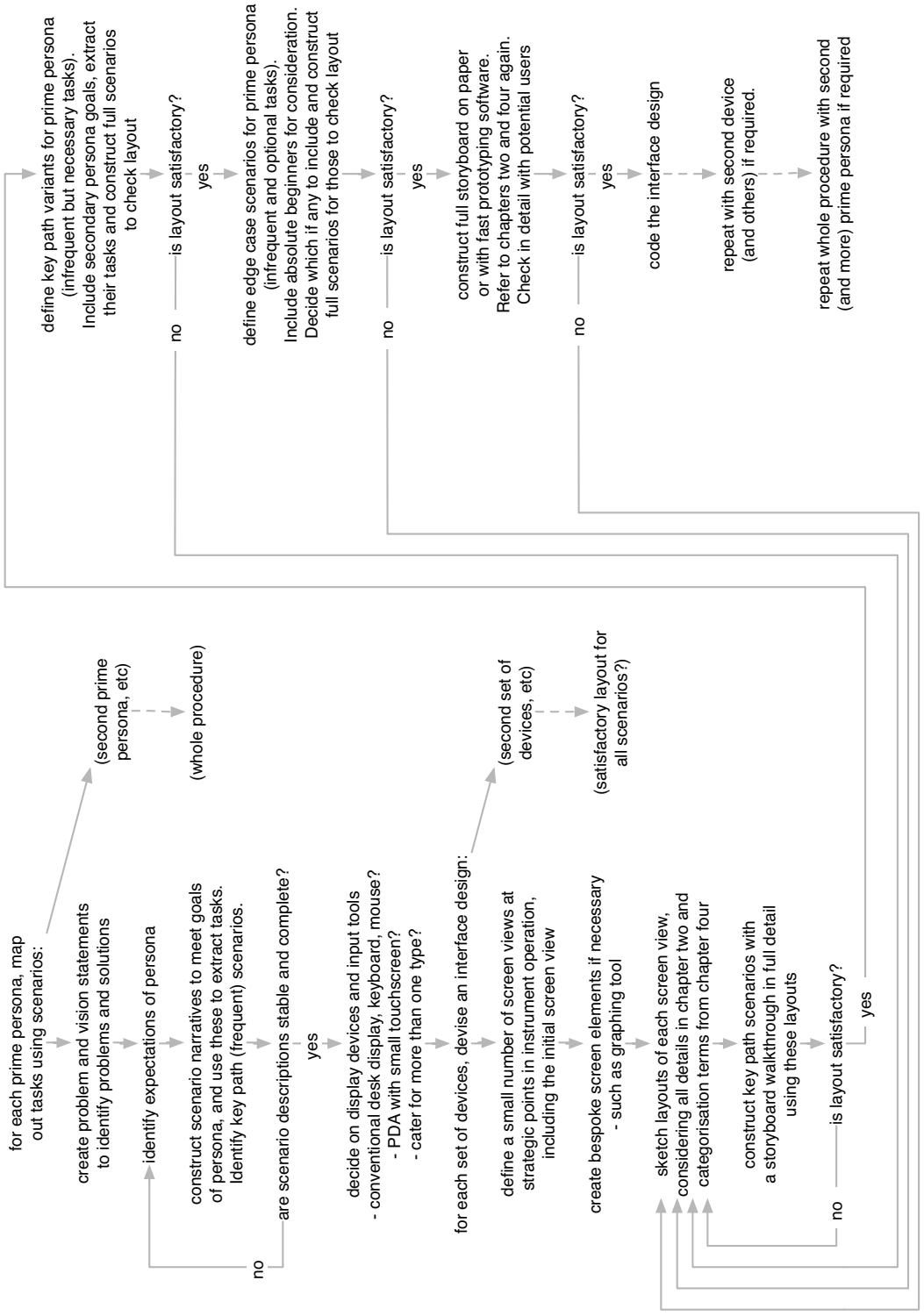


Fig. 3 Interface Flow Chart (sub-chart of Fig. 1 & Fig. 2)

3 Usability Requirements

The sequences above show a suggested plan against time for a space instrument, but by its very nature it is difficult to include all relevant material in a clearly presented manner. The list of usability requirements shown in Table 1 below attempts to show every aspect relevant to the usability of such an instrument but without implying any sequential information. The aspects are presented in the approximate order of development in this thesis and include suggested references for further information. A compliancy box allows the table to be used as a check sheet. Use of both the sequence diagrams and the requirements table should give a comprehensive approach to maximising the usability of a space instrument. There is some overlap between this table and the interface tables presented in Chapter Five, but the emphasis is different; this table is intended for use during the planning stages, whilst the others are intended to ease examination of an interface once built.

Usability Requirement	Compliant	References for information (ch 'n' is in this thesis)
consider layout standards & guides		ch2; (Apple Computer 2005a); (Microsoft 2004); (Sun Developer Network 2005); (BSI 1998); (BSI 1999b)
encourage mental models		ch2; (Norman 1983a); (DiSessa 1986); (Johnson-Laird 1983); (Cooper 2003)
use mapping & alignment		ch2; (Norman 1988)
look at alternative devices and displays apart from keyboard, mouse, screen		ch2
give user feedback to controls		ch2; (Gibson 1977); (Norman 1988)
consider Fitts' and Hick's laws		ch2; (Fitts 1954); (Raskin 2000)
use colour appropriately		ch2; (Tufte 1990); (IEE 2004)
consider overall aesthetics		ch4; (Burkhardt 2000)
look at previous examples of work		ch2
use interaction parameters		ch4
use representation parameters		ch4
use consistency parameters		ch4
use error management parameters		ch4
use operator aspects parameters		ch4
consider science group type & size		ch4
consider operation		ch4
consider timescales		ch4
understand user groups		ch4
understand technical & personal goals		ch4; (Cooper 2003)
use persona analysis to understand usage		ch4; (Cooper 2003)
define communication model		ch4

Table 1 Usability Requirements Check List

Usability Requirement	Compliant	References for information (ch 'n' is in this thesis)
define number of interfaces		ch4
be consistent for usage, including appearance, layout, disadvantaged access		ch5; (Nielsen 1989)
be consistent with software tools, and software modules		ch5
put usability before consistency		ch5; (Nielsen 1989)
do frequent user testing, face to face meetings, formal assessments		ch5; (Nielsen 1989); (Koratkar 2000)
separate the user interface from other software		ch5; (Jones 2000)
consider using artificial intelligence		ch5; (Koratkar 2002)
re-use existing software		ch5; (Koratkar 2000)
use model-view-controller software architecture		ch5; (Fowler 2006)
implement documentation system		ch5; (Koratkar 2000)
allow planning tool to be used in a flexible manner		ch5; (Burkhardt 2000)
use terms familiar to scientists for planning tool		ch5; (Burkhardt 2000)
ensure interfaces are responsive		ch5; (Burkhardt 2000)
provide enough information to make science trade-offs		ch5; (Burkhardt 2000)
create simple installers, preferably by web page		ch5; (Burkhardt 2000)
make software platform independent		ch5; (Burkhardt 2000)
ensure information only requires entering once		ch5; (Burkhardt 2000)
use same tools for scientists and technical operators		ch5; (Burkhardt 2000)

Table 1 Usability Requirements Check List

Usability Requirement	Compliant	References for information (ch 'n' is in this thesis)
group access of any on-line information into one session		ch5; (Burkhardt 2000)
use XML for miscellaneous file storage		ch5; (Burkhardt 2000)
use adaptable software architecture to allow multiple iterations of instrument description		ch5; (Ames 2000)
for instrument autonomy, consider data selection, prioritisation, compression, goal-oriented operation and auto re-survey		ch5; (Koratkar 2002)
use visualisation in tools		ch5; (Burkhardt 2000)
use rapid prototyping method		ch5; (Jones 2000)
consider natural language interface		ch5; (Jones 2000)
ensure available skills match intended technology		ch5; (Koratkar 2004b)
ensure scientist is part of software development team		ch5; (Jones 2000)
ensure good communication in development team		ch5
consider one software language for instrument and for support tools		ch5
ensure simulator closely models scientific response		ch7
ensure simulator tracks instrument calibration		ch7
ensure simulator replicates all instrument systems		ch7
use simulator to track consumables usage and mechanism wear		ch7

Table 1 Usability Requirements Check List

Usability Requirement	Compliant	References for information (ch 'n' is in this thesis)
provide formatted data from simulator		ch7
ensure simulator is an integral part of instrument lifecycle and its development precedes the instrument		ch7

Table 1 Usability Requirements Check List

4 Challenges of this Method

As with any change to an established process, the new approach proposed here poses certain challenges. The most prominent is that of shifting the costs for software effort to the start of the programme, whilst at the same time appearing to increase the range of software effort required by also producing a user interface simulator. Whilst the traditional approach has been to start work on instrument software part way through the hardware development programme, this has frequently ignored the potential contributions to system design that experienced software engineers could make if they were involved right at the start. In the interviews carried out for Chapter Four of this thesis, the wish to be part of the decision making process from the very start was one of the most strongly expressed opinions, particularly amongst those involved in programming. The suggested approach in the present chapter usefully includes this contribution, whilst showing that early provision of a software interface simulator is likely to lead to earlier cognition of the optimum development path to follow. In turn, this is likely to lead to fewer developmental dead ends and the resultant waste of human and financial resources. The net effect may well be that over the mission lifetime the costs of the approach in this chapter may actually be less than typical existing methods, even taking into account the loss of notional interest on the finance involved by using it early.

Verification of this reduction in cost would be difficult, as it simply would be ludicrous to run in parallel two similar instrument programmes with one as a control. Also, the first time a programme is run in this manner it would be likely to show flaws in the process, generating less than optimum benefit. In addition, it would be necessary to have a way of assessing how well the finished instrument met the goals of the mission, including those parameters that should have been part of the specification but eluded the capture process.

An associated factor is the work package structure. Typically this is agreed very early in the programme with the funding body, who then build their internal budgets based upon a collection of such work packages from various bodies. The result is a rigid structure that requires - and typically undergoes - lengthy re-costing exercises that consume valuable management effort, as the instrument is developed. The process advocated in this chapter pushes the bulk of the uncertainty to as early in the programme as possible, which should ease financial planning in the late stages. The corollary is that in the early stages of this programme, at least up to the proof of concept **(Fig. 1[c])**, the initial planning may be subject to comparatively large changes, and the work package structure must be flexible enough to easily accommodate it.

Another challenge to adopting this style of working is the inclusion of a practising astronomer as a development team member, which was strongly advocated by the GSFC SEA team (Chapter Six section 3.1 “evaluation”). They found that following this approach brought in to the team the astronomical knowledge that a typical computer science background unsurprisingly did not. This integration of technology and science, and the shift away from direct astronomical work for an extended period for one person, may be a difficult idea to accept for some groups. The right person is likely to find it an enlightening experience, by giving them the technological insight to obtain the best from astronomical instruments, but the cultural challenge may be a barrier. Some way of addressing the pressure to publish regular science papers would need to be found.

5 Summary

This chapter presents a suggested new approach to formulating a timeline for the lifecycle of a typical space instrument, taking into account the information developed in this thesis. It includes a checklist to allow referral back to the appropriate chapter or other source. By encouraging more thorough user participation in the design process and moving design decisions to early points in that process, the instrument that results should be easier to learn and use, it should be better tested and commissioned, and the science return should be correspondingly improved.

Chapter Nine: Conclusions and a Forward Look

These conclusions first consider each chapter in turn to examine what might be learnt from each sub-topic of this thesis and how each contributes to proving the initial hypothesis. Then an overall set of conclusions is made, and a forward look taken as to where this work could go next.

1 Conclusions

1.1 Initial hypothesis

The hypothesis put forward to be proven or disproven is:-

The science return of current space instruments is constrained by the user interface and could be improved by a new approach.

1.2 Review and Discussion

1.2.1 Chapter One - Science, Machines, Users and Methodology

The change from widespread use of individual mechanical controls and command line interfaces to graphical elements on a display screen has been surprisingly rapid and has opened up new ideas for control and data processing. The commercial world has had the financial motivation to embrace HCI design disciplines, as shown by the example of the mobile phone. The space science world has not perceived the need to go this route (with notable exceptions explored in Chapter Two) and therefore budgets to fulfil an HCI design role have not generally been considered.

The term ‘science return’ is difficult to quantify. Counting science papers from a given instrument, or the number of citations, or attempting to assess the quality of the science data, all have their limitations as methods. Assessing usability of an instrument is adopted in this thesis as the method of measuring science return.

Science return is also affected by the manner in which resources such as funds and time are used. Formal programme organisation and re-use of existing component parts can help with the efficient use of limited resources.

1.2.2 Chapter Two - Exploring Interfaces

Mental models that reflect user tasks are one effective way of approaching the detailed design of a user interface, and are one method of communicating the designer’s intentions to the user. Models may be improved by considering the use of physical actions required to

control the system, and by considering feedback from the system, as well as by using the more obvious senses of sight and sound. Several popular sets of heuristics exist from different authors, platform vendors and standards organisations to guide layout of interfaces. Natural physical mappings of interface components which relate to the instrument under control can help to reduce the need for labels and limit any clutter, and hence reduce confusion. Appropriate choice of interface devices, for example a hand-held display with a touch sensitive screen, may bring different concepts such as portability to science instrument use.

A body of work exists (Fitts' law, GOMS, etc) to quantify typical user reaction times for a given interface design, but is aimed at optimising highly repetitive actions in time critical situations. It may be overkill for the space science context where time is not usually a critical factor.

Appropriate use of colour can help comprehension of an interface, but care must be taken that it is used as secondary to the main design in order not to disadvantage the part of the population that have limitations with their colour vision. Consistent mid-tone themes are likely to be good starting points, using saturated colour to signal warnings and errors.

Appreciation of the workings of human memory are also important. Many interactions with a display screen require holding transient information for a short period. The design of a system should minimise this requirement and recognise that human short term memory tends to fail progressively and after about 30 seconds that transient information may be lost. The operator may have to regenerate it, reducing the usability of the instrument. Similarly, if a system already has a particular piece of information entered, or a required piece of information can be deduced from existing entries, then the user should not have to remember and re-enter it.

A wide range of styles of existing interface show approaches that range from electronic form filling to sophisticated interactive displays complete with on-line database lookup. Positioning of display items varies from apparently *ad-hoc* to being based on a workflow, and the use of colour varies from almost none to arguably too saturated. In judging an interface, it is important to note that the dynamic operation needs to be assessed as well as the static appearance.

In the work on exploring the programming of a virtual spectrometer, the use of the Eclipse development environment tool became critical to finishing the work in the time available. The iterative compilation process that was available meant that programmer errors were picked up immediately and were much less likely to be repeated as a result. The tool also illustrated in a clear manner the advantages of the iterative incremental technique suggested in this

thesis for science instrument interfaces. By calculating the new data immediately a parameter is changed, such a tool allows the user to see the results of their action immediately and to gain insight into the instrument operation. Notably the Hubble VTT and the Gemini Position Editor, and possibly others, understood this and had incorporated this principle of operation.

The use of Java showed the merits of platform independence, web browser delivery if required and the default availability of a very wide range of library support functions (classes). The excellent programming environment has already been mentioned. To complete an assessment of the merits of one software language over another, a similar spectrometer task could be attempted using for example C++, LabVIEW, and IDL. The topic of network access should be explicitly addressed in any comparative study.

The lack of specific results from the usability tests carried out showed that one needs good organisation, plenty of willing subjects, and an assessment task pitched at a correct level for those subjects. It also pointed the way towards a more analytical method for evaluating user requirements, which evolved into the persona technique described later. Personas allow the researcher to target interface design much more closely to potential user requirements, and then carry out user testing with a much more focused methodology.

1.2.3 Chapter Three - Human-Computer Interface Case Studies

A detailed study of five disparate system failures, concentrating on the human interface issues, showed that in each case these issues were important. The resultant inquiries found this also. For example, over one hundred alarms could sound at once in the TMI case, causing operator confusion. Verification of the spin state of a gyro would have been sufficient to avoid the SOHO spacecraft loss. A control labelling issue was the root cause of the Strasbourg air accident, exacerbated by the lack of a system-level backup.

The evolution in system interfaces referred to in Chapter One is particularly evident if one compares the photograph of the TMI control room in Chapter Three Fig. 2 with, for example, the SOAR control screen in Chapter Two Fig. 20.

The case studies provide a base from which to extract a set of questions that can be used to evaluate and synthesise machine interfaces. In addition to the anticipated display-related factors that are the part of the interface, the studies showed several environmental and team-based factors that are important in generating an overall system solution.

1.2.4 Chapter Four - Interface, Goals and User Analysis

A set of interface criteria from traceable sources and optimised for space science is developed and forms part of a framework of parameters for use in interface design. Other analysis shows appreciations of science user, operation, timescales and access categories. Users and goals complete the categorisation. The technique of persona analysis, based on interviews with real users, shows how interface design can be targeted at the specific people to whom it is relevant, and how to identify those to whom it is not relevant. When this process is followed for the general case of space instrumentation, it shows that interface design clearly should be carried out for two distinct groups of users. The technique gives a methodical approach to what may sometimes be a rather arbitrary process.

A recommended procedure for the process of moving from the allocation of interfaces towards a full functional interface, shows how testing of actual users against potential interface designs can safely be left to later in a design programme than might happen without using a persona method.

1.2.5 Chapter Five - Usability Criteria Verification

Confidence is built in the usability criteria developed in the previous chapter by using them to analyse a set of interfaces taken from various astronomical instruments. By making this selection carefully, predictable differentials would be expected between the samples when tested against the usability criteria. As this is what is found, and in the ranking expected, the conclusion is that these criteria are a reasonable first attempt to define a way of analysing and synthesising interfaces in the field of space science.

The assessment shows the difficulty of calibration of one's judgement when working in this field. In order to obtain a proportional measure of the effectiveness of a given interface, rather than a simplistic 'good' or 'bad', it is necessary to have similar examples at hand with which to compare.

The data are presented as tables and also as polar plots. The latter, by providing a compact, graphical presentation, allow a more detailed comparison between interfaces than a simple total points score. A grouping of high scores in one area might be enough to skew the results if based simply on a total score, but would be evident by using these plots. In the case of the results in this chapter, the polar plots agree with the expected ranking achieved.

1.2.6 Chapter Six - A Common User Interface?

The goal of a high degree of commonality between the user interfaces of different space science instruments is one that is difficult to strive for, and involves as many organisational

issues as technical challenges. It implies international collaboration, a willingness to come to a consensus and accurate technical work. A party keen on its own proprietary methods would rapidly cause lack of standardisation, analogous to the current situation with web standards and Microsoft Corporation.

The Scientist's Expert Assistant and Science Goal Monitor groups at GSFC recognised the need for reduced operational costs for orbital missions by introducing software standardisation and autonomy to instrument design. International collaboration on standards was partially successful, as illustrated in (Chapter Six) Fig. 3, but there appears to be a continuing trend to drift apart again. A useful resource of Java software for space science applications resulted from this work.

Many forward-thinking ideas resulted from the research at GSFC detailed in Chapter Six. For example, the Swift spacecraft and the Mars Exploration Rovers demonstrate functional implementations of ideas on autonomy. Many future spacecraft will be capable of collecting such massive data sets that on-board selection and compression of the data will be the only means to utilise telemetry links successfully. There are suggestions on the detail of software development programmes which are very appropriate to the space science context, such as to adopt a rapid prototyping methodology and to use embedded documentation. They found that an astronomer working as a member of the programming team was very important in order to capture requirements accurately.

There is a promising potential for releasing funds for space science by adopting the ideas in this chapter, as most are likely to result in more efficient ways of working. To achieve this, confidence must be built in the science community that new technologies will not compromise the science return and are likely to improve it.

1.2.7 Chapter Seven - Use of an Instrument Simulator

By re-thinking the typical organisation of a space instrument programme, there is potential for building an instrument that better meets users' needs. Key to this is use of an instrument simulator at the inception of the programme which evolves in capability as development takes place. The goal is to move as many decisions as possible to as early in the programme as possible by constructing a software simulator that will run adequately on easily affordable computing platforms. It would be specified to give an acceptable level of science data accuracy whilst meeting operational parameters of being responsive in time and realistic in imitating instrument systems. The result could be an instrument that has gains in science return compared to the results obtained from the typical organisation of an instrument programme.

The gains are expected to be realised by creating a situation where all users of an instrument have access to a simulator throughout the design, development, test and operation lifecycle. The term ‘users’ implies not only nominal science users, but also for example the programmers, hardware designers and system operators that also need to interact with the instrument in order for it ultimately to deliver good results. Widespread use of multiple instances of a realistic simulator should promote the detailed communication and understanding that is necessary between those groups of users.

Consideration should be given as to whether the development of instruments in a commercial or in a research environment is likely to give the better science return.

1.2.8 Chapter Eight - A Possible New Approach

By taking the issues developed in the preceding chapters, a new approach is proposed for the entire programme of a space instrument. The key factors are:-

- well-defined user interfaces with early definitions at the start of the programme
- use of a realistic simulator throughout the programme
- participation of a scientist in the writing of system software
- an early decision on strategic technology additions
- frequent feedback to and agreement of developments with the whole instrument team

Details of interface design and examples in context can be seen in Chapter Two, with detailed heuristics and user analysis in Chapter Four. This chapter also provides a specific methodology for design of user interfaces. There are a large number of technology suggestions based on highly relevant research by NASA in Chapter Six, and there are also very apposite suggestions on instrument team structure. The concept of a lifecycle instrument simulator is developed in Chapter Seven.

This new approach is seen as optimising the instrument procurement and operation processes to produce the best design and ease of operation, in order to enhance the science return.

1.3 Overall Conclusions

Can we show that the stated hypothesis is proven? Let us take each term in turn:

science return - We discussed possible means of measuring this and settled on ‘usability’ as viable and also as relevant to the work here. Parameters such as ‘takes less time to operate’, ‘tends to allow fewer incorrect configurations’, and ‘makes best use of the allocated time’ were used as examples of the meaning of improving usability.

current space instruments - A survey of the interfaces used on various instruments over the last decade shows a high degree of variability. Some of this is simple progress over time, but the latest does not necessarily represent the best.

constrained by the user interface - Many aspects of user interfaces have been shown, from fundamental operating system building blocks upwards. The results of exploratory programming work have been shown, and a proven methodology for building the actual interface has been described. A set of guidelines drawn from case studies and other research work that has been optimised for the space science context gives a means of assessing user interfaces, and a first level of confidence in their use has been provided by testing against existing interfaces.

This interface testing clearly shows large differences even between current interfaces. The best, with a pedigree of many years development and also built as part of a programme with explicit consideration of user requirements, is very sophisticated. Others show less appreciation of such practices. In testing the hypothesis, it is therefore too simplistic to say 'current space instruments'. The term should change to refer to some or most, rather than to all, instruments. Some methods to improve the situation are understood, but require to be made more widely known.

The issue of *user* has been examined by exploring as to exactly who are the users of an instrument over its whole lifecycle. The questions of timescales, operating environment and user goals have been addressed. The technique of 'personas' has been described, and then used to show the prime interface groupings for a generic space instrument. A clearer division of responsibilities than typically employed has been suggested as a result.

be improved by a new approach - The prescient work carried out at the GSFC shows many ideas for improving space instrument design in order to increase the science return, particularly in the areas of on-board data selection and compression. Some implementation of these ideas will be necessary to make new designs viable due to the huge data sets proposed. At the point of operator interaction, interface designs have been implemented with embedded documentation and using expert systems methods for operator assistance. These designs have released funds that otherwise would have been consumed by operational support.

The use of a simulator over an instrument lifecycle, and the resultant new programme flow that is suggested, are thought to be original contributions to the subject from this thesis. By adopting a method of bringing a wide range of users into the instrument process as early as possible, better attention should be paid to their needs and a more efficient instrument with an improved science return should be the result. The important concept is that the simulator

is designed not only to allow the instrument to be developed to the best possible extent, but also to allow the *usability* of the instrument to be optimised as well. Usability is treated as a design parameter.

From this discussion, the hypothesis is considered met, and an acknowledgment is made that at least some instruments are already paying attention to the user interface. Even they may well be capable of improvement; they just do not have so far to go as some others.

2 A Forward Look

This thesis has hinted at what could be, and so it is appropriate to end by attempting to look forward a little way into the future of space instrumentation and highlight desirable developments.

2.1 Autonomy

Autonomous operation of instruments and spacecraft is already being explored, and has the drivers of technical necessity or financial benefit behind it. Response to unpredictable external events, or a communication difficulty such as an orbit on the other side of the Sun, or long loop delays in real time operation modes such as planetary rovers, are three examples where autonomy is becoming necessary.

With an autonomous operation mode there is the question of what form commands should take. Instead of well defined tasks, such as ‘move forward ten metres or until front sensor looses contact and then stop’, the command is likely to be of the form of goals such as ‘find a crater likely to be formed from a low velocity impact’. What is implied here is also the use of natural language commands, rather than sub-system specific actions and limits.

2.2 Simulators

A widespread use of whole instrument simulators would be likely to lead to instruments specified much more closely to what can be used, rather than what is thought to be wanted. By offering all users - scientists and technologists - throughout the lifecycle a means of operating and hence optimising a nascent instrument, instruments with an improved science return are likely to result.

2.3 Modular Software Architecture

Following on from the ideas of JSky in Chapter Six, a shared modular architecture for on-board and ground support systems would seem to have much to recommend it. The costs of developing software from scratch are typically large, and an ability to re-use routines from

previous missions would be welcomed by many organisations. The goal of software re-use is frequently quoted in many different contexts however, and often not met because of the many pitfalls on the way.

2.4 Automated Design Generation

The SGM work suggested the use of structured documents to take some of the burden of the detailed design phase off the technical staff. A great deal of design work is taking detailed information from one point and placing it at another point, with nothing less than perfect accuracy. The creative contribution comes from joining those points of information together. An example would be taking two large integrated circuits and putting them on a circuit board with a communications connector. A structured document would have the library sources for the parts and would automate a large part of the work, leaving just those areas requiring innovation to be completed manually. The result would be more accurate work at lower cost, with the further advantage that design reconfiguration for another mission would be able to make use of most of the work carried out for the first.

The future looks exciting; there seems to be a growing appreciation of how good instrument design involves people as well as technology, and how one must look at it in the round over a whole lifecycle. This thesis has done its best to contribute to that discussion.

Bibliography

This list of books, papers, and web sites contains the details of the references in the previous chapters, plus other relevant reading material.

Adams D., 1977: *The Hitchhiker's Guide to the Galaxy*, pub: BBC. The 'Heart of Gold' had an emergency backup personality for the shipboard computer that was modelled on an over fussy matron.

Air Accidents Investigation Branch, 1989: *Report on the accident to Boeing 737-4000 - G-OBME near Kegworth Leicestershire on 8 January 1989*, http://www.aaib.gov.uk/cms_resources/dft_avsafety_pdf_502819.pdf, See particularly report appendix 2.7.

Ames T.J., Koons, L., Sall, K. and Warsaw, C., 2000: *Using XML and Java for telescope and instrumentation control*, in Advanced Telescope and Instrumentation Control Software, pub: SPIE, vol. 4009, pp. 2-12. Ed: Lewis, H.

Apple Computer, 1993: *MacIntosh Human Interface Guidelines*, pub: Addison-Wesley.

Apple Computer, 2005a: *Macintosh OSX Human Interface Guidelines*, http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/index.html.

Apple Computer, 2005b: *Interface Builder*, part of Developer Tools. Pub: Apple Computer Inc.

Ashe M.C., Schumacher, G., 2000: *SOAR Telescope Control System: A Rapid Prototype and Development in LabVIEW*, in Advanced Telescope and Instrumentation Control Software, pub: SPIE, vol. 4009, pp. 48 - 60.

Atkinson R.C., Shiffrin, R.M., 1968: *Human memory: A proposed system and its control processes,* in The Psychology of Learning and Motivation: Advances in research and theory, vol. 2, pp. 89 - 195.

Aviation Safety Network, 1992: *Strasbourg Airbus A320 accident*, http://aviation-safety.net/database/1992/920120-0.htm.

Bandecchi M., Melton, B., Gardini, B. and Ongaro, F., 2000: *The ESA/ESTEC Concurrent Design Facility*, in Proceedings of EuSEC 2000, pub: INCOSE.

Barnard V., Leech, J., 2004: *The JCMT OT primer*, http://www.jach.hawaii.edu/software/jcmtot/.

Bastien J.M.C., Scapin, D.L., 1993: *Ergonomic Criteria for the Evaluation of Human-Computer Interfaces,* in Inria Rocquencourt, vol. Programme 3. http://www.webmaestro.gouv.qc.ca/ress/webeduc/2000nov/criteres.pdf

Bastien J.M.C., Scapin, D.L., 1995: *Evaluating a user interface with ergonomic criteria*, in Int. J. Hum.-Comput. Interact., vol. 7, iss. 2, pp. 105-121.

BBC, 2006: *UK Weather*, http://www.bbc.co.uk/weather/ukweather/temperature.shtml, accessed 24 Feb 06.

Benson C., Elman, A., Nickell, S. and Robertson, C.Z., 2004: *GNOME Human Interface Guidelines 2.0*, pub: The GNOME Usability Project. http://developer.gnome.org/projects/gup/hig/, accessed: 9 July 06.

Bentley R., Hughes, J.A., Randall, D., Rodden, T., Sawyer, P., Shapiro, D. and Sommerville, I., 1992: *Ethnographically-informed systems design for air traffic control,* in ACM CSCW 92 Proceedings, iss. November 1992.

Benyon D., 2001: *The new HCI? Navigation of Information Space,* in Knowledge-based Systems, vol. 14, pp. 425 - 430.

Bergman E., 2000: *Information Appliances and Beyond: Interaction Design for Consumer Products*, pub: Morgan Kaufmann.

Bignell V., Fortune, J., 1984: *Understanding System Failures*, pub: Manchester University Press.

Blacker B.S., Burkhardt, C., Koratkar, A. and Younger, J., 2000: *A New Era for HST Phase 1 Development and Submission*, in Observatory Proceedings to Optimise Scientific Return II, pub: SPIE, vol. 4010, pp. 256 - 266. Ed: Quinn, P.J.

Boehm B., Egyed, A., Kwan, J., Port, D., Shah, A. and Madachy, R., 1998: *Using the WinWin Spiral Model: A Case Study,* in IEEE Computer, iss. July 1998, pp. 33 - 44.

Boguslaw R., 1965: *The New Utopians - A Study in System Design and Social Change*, pub: Prentice Hall.

Brekke P., Haugan, S.V.H., 1997: *CDS Quicklook Software User Manual*, pub: UiO, Oslo. http://solar.bnsc.rl.ac.uk/software/notes.shtml.

Bretfellner M.G., Ehle, M. and Dahlem, M., 2004: *XMM-Newton Proposers' Guide and Remote Proposal Submission Software Users' Manual*, pub: ESA, XMM-PS-GM-17 Issue 4.0, http://xmm.vilspa.esa.es/external/xmm_user_support/documentation/index.shtml.

Brighton A., 2000: *JSky Home Page*, http://archive.eso.org/JSky/.

BSI, 1998: *Ergonomic requirements for office work with visual display terminals*, BS EN ISO 9241-11:1998, Part 11: Guidance on usability.

BSI, 1999a: *Human-centred design processes for interactive systems*, BS EN ISO 13407:1999.

BSI, 1999b: *Ergonomic requirements for office work with visual display terminals*, BS EN ISO 9241-13:1999, Part 13: User guidance.

Bureau d'Enquêtes et d'Analyses, 1992: *Rapport de la commission d'enquête sur l'accident survenu le 20 janvier 1992 près du Mont Sainte-Odile (Bas Rhin) à l'Airbus immatriculé F-GGED exploité par la compagnie Air Inter*, F-ED920120, http://www.bea-fr.org/, Search reports by plane registration (F-GGED).

Burkhardt C., Fishman, M., Grosvenor, S., Jones, J., Koratkar, A., Ruley, L. and Wolf, K., 2000: *NGST Scientist's Expert Assistant Final Report*, pub: NASA Goddard Space Flight Center, http://aaaprod.gsfc.nasa.gov/SEA/papers/Default.htm.

Card S., Moran, T. and Newell, A., 1983: *The Psychology of Human Computer Interaction*, pub: Erlbaum, Hillsdale, New Jersey.

Cecil G., Crain, A. and Schumacher, G., 2002: *Remote Use of the SOAR 4.25m Telescope with LabVIEW*, in Advanced Global Communications Technologies for Astronomy II, pub: SPIE, vol. 4845, pp. 72-79. Ed: Kibrick, R.I.

Cecil G., Crain, J.A., 2004: *SOAR Remote Observing: Tactics and Early Results*, in Advanced Software, Control and Communication Systems for Astronomy, pub: SPIE, vol. 5496.

Chua W.F., 1986: *Radical Developments in Accounting Thought,* in The Accounting Review, vol. 61, iss. 1986, pp. 601 - 632.

Cleveland W.S., 1994: *The Elements of Graphing Data*, pub: Hobart P.

Collins H.M., 1990: *Artificial Experts*, pub: MIT Press.

Cooper A., 1999: *The Inmates Are Running The Asylum: Why high Tech Products Drive Us Crazy And How To Restore The Sanity*, pub: Sams.

Cooper A., Reimann, R., 2003: *About Face 2.0: The Essentials of Interaction Design*, pub: Wiley, Indianapolis, Indiana.

Covey S.R., 1992: *The 7 Habits of Highly Effective People*, pub: Simon & Schuster Press.

Devlin K., 1998: *Goodbye Descartes*, pub: Wiley.

DiSessa A., 1986: *Models of Computation*, in User-Centred System Design: New Perspectives in Human-Computer Interaction, eds: Norman, D.A. and Draper, S.W., pub: LEA, Hillsdale.

Dix A., Finlay, J., Abowd, G. and Beale, R., 2004: *Human-Computer Interaction*, pub: Pearson/Prentice-Hall, Harlow.

Dixon A., 2002: *Interfaces*, pub: British HCI Group, vol. 50.

Dorner D.G., Curtis, A., 2003: *A comparative review of common user interface software products for libraries*, pub: School of Information Management, Victoria University of Wellington, http://www.natlib.govt.nz/mi/whatsnew/4initiatives.html#review.

Dreyfus H., 1972: *What Computers Can't Do - A Critique of Artificial Reason*, pub: Harper and Row.

Dreyfus H., Dreyfus, S., 1986: *Mind over Machine. The Power of Human Intuition and Expertise in the Era of the Computer*, pub: Blackwell.

Dreyfus H., 1993: *What Computers (Still) Can't Do - A Critique of Artificial Reason*, pub: MIT Press.

Duric Z., Gray, W.D., Heishman, R., Li, F., Rosenfeld, A., Schoelles, M.J., Schunn, C. and Wechsler, H., 2002: *Integrating Perceptual and Cognitive Modeling for Adaptive and Intelligent Human-Computer Interaction,* in Proceedings of the IEEE, vol. 90, iss. 7, pp. 1272 - 1289.

Eclipse Foundation, 2006: *Eclipse*, http://www.eclipse.org/downloads/.

Eisner H., 2002: *Essentials of Project and Systems Engineering Management*, pub: Wiley & Sons.

ESA, 2005: *Space engineering - System engineering - Part 6: Functional and technical specifications*, pub: ESA, http://www.ecss.nl/.

ESA, 2006: *ESA-CDF Concurrent Design Facility*, http://www.esa.int/SPECIALS/CDF/.

Etherton J., Rees, P.C.T. and Steele, I.A., 2000: *Telescope Design and Efficiency*, in Observatory Operations to Optimise Scientific Return II, pub: SPIE, vol. 4010, p. 298. Ed: Quinn, P.J.

Euro-VO, 2006: *European Virtual Observatory*, http://www.euro-vo.org/pub/, accessed 26 June 06.

Federal Election Commission, 2006: *Federal Elections 2000*, http://www.fec.gov/pubrec/fe2000/2000presge.htm, accessed: 7 July 2006.

Fielding N., Schreier, M., 2001: *Introduction: On the Compatibility between Qualitative and Quantitative Research Methods,* in Forum Qualitative Research, vol. 2, iss. 1.

Fitts P.M., 1954: *The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement,* in Journal of Experimental Psychology, vol. 47, pp. 381 - 391.

Folger M., Bridger, A., Dent, B., Kelly, D., Adamson, A., Economou, F., Hirst, P., Jenness, T., Bohlender, D.A., Durand, D. and Handley, T.H., 2002: *A new observing tool for the James Clerk Maxwell Telescope*, in Astronomical Data Analysis Software and Systems Xi, pp 453-456.

Fowler A., 2006: *A Swing Architecture Overview*, pub: Sun Developer Network. http://java.sun.com/products/jfc/tsc/articles/architecture/index.html.

Fowler M., Scott, K., 1999: *UML Distilled*, pub: Addison Wesley.

Frøkjær E., Hertzum, M. and Hornbæk, K., 2000: *Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?*, in CHI 2000, pub: ACM, vol. 2.

Garvin D., 1987: *Competing on the Eight Dimensions of Quality,* in Harvard Business Review, vol. 65:6, pp. 101-109.

Gerzanics M., 2003: *Easy Does It*, in Flight International, 18 - 24 November 2003, pp. 56 - 60. Article about Dassault 'EASy' flight deck.

Gibbs W.W., 2004: *Why Machines Should Fear*, in Scientific American, January 2004, pp. 27 - 28. Article about Donald A. Norman.

Gibson J., 1977: *The Theory of Affordances*, in Perceiving, Acting and Knowing, ed: R. E. Shaw, J.B., pub: Lawrence Erlbaum Associates.

Glaser, Strauss, 1967: *The Discovery of Grounded Theory*, http://www.groundedtheory.com.

Goodwin K., 2005: *Perfecting Your Personas.*, http://www.uie.com/articles/perfecting_personas/, Kim Goodwin is part of Cooper Inc.

Goza S.M., Ambrose, R.O., Diftler, M.A. and Spain, I.M., 2004: *Telepresence Control of the NASA/DARPA Robonaut on a Mobility Platform*, in CHI 2004, pub: ACM, vol. 6, pp. 623 - 629.

Greene K.d., 1970: *Systems Psychology*, pub: McGraw-Hill.

Guba, Lincoln, 1994: *Competing Paradigms in Qualitative Research*, in Handbook of Qualitative Research, pp 105 - 117, eds: Densin, N.K. and Lincoln, Y.S., pub: Sage Publications, Thousand Oaks, CA.

Gurney K., 1997: *An Introduction to Neural Networks*, pub: CRC.

Hackos J.T., Redish, J.C., 1998: *User & Task Analysis for Interface Design*, pub: Wiley.

Hall A.D., 1967: *Methodology of Systems Engineering,* in IEEE Transactions on Engineering Management.

Head A.J., 1999: *Design Wise: A Guide for Evaluating the Interface Design of Information Resources*, pub: CyberAge Books.

Hick W.E., 1952: *On the Rate of Gain of Information,* in Quarterly Journal of Experimental Psychology, vol. 4, pp. 11 - 26.

Hopgood A., 2000: *Intelligent Systems for Engineers and Scientists*, pub: CRC Press.

HST, 2006: *Astronomer's Proposal Tool*, http://www.stsci.edu/hst/proposing/apt.

Huckle H.E., Smith, P.J., 2004: *UVOT Autonomous Operations*, in X-Ray and Gamma-Ray Instrumentation for Astronomy XIII, pub: SPIE, vol. 5165, pp. 298 - 309. Eds: Flanagan, K.A. and Siegmund, O.H.W.

Hull, Jackson, D., 2002: *Requirements Engineering*, pub: Springer-Verlag.

IEE, 2004: *Colour Vision Defects*, pub: Institution of Electrical Engineers, p. 6. http://www.iee.org/Policy/Areas/Health/cvdintro.cfm.

IEEE, 1994: *IEEE standard for information technology - X window system - Modular Toolkit Environment (MTE)*, pub: Inst. Electr. & Electron. Eng., New York, NY, USA, IEEE Std 1295-1993, p. xvi+488.

Imhof E., 1982: *Cartographic Relief Presentation*, Berlin.

INMOS, 2002: *Occam Language*, pub: WoTUG. http://www.wotug.org/occam/.

Jacko J.A., Sears, A., 2003: *The Human Computer Interaction Handbook*, pub: Lawrence Erlbaum Associates.

John B., Vera, A.H., Mantessa, M.P., Freed, M. and Remington, R.W., 2002: *Automating CPM-GOMS*, in CHI 2002, pub: ACM, vol. 4, pp. 147 - 154.

Johnson C., Holloway, C.M., 2003: *The ESA/NASA SOHO Mission Interruption: Using the STAMP Accident Analysis Technique for a Software Related 'Mishap',* http://www.dcs.gla.ac.uk/~johnson/papers/, accessed 9 July 2006.

Johnson J., Hallam, N.J., 1990: *Safety Critical neurocomputing: exploration and verification in knowledge augmented neural networks,* in IEE Digest, vol. 1990/179.

Johnson J., Nielsen, J., 2000: *GUI Bloopers: Don'ts and Dos for Software Developers and Web Designers*, pub: Morgan Kaufmann.

Johnson P., 1992: *Human Computer Interaction*, pub: McGraw-Hill.

Johnson-Laird P., 1983: *Mental Models: towards a cognitive science of language, inference and consciousness*, pub: Harvard University Press, Cambridge, Mass.

Jones J.E., 1998: *NGST SEA Design Document*, pub: NASA / Goddard Space Flight Center, http://aaaprod.gsfc.nasa.gov/sea/papers/, accessed 9 July 2006.

Jones J.E., Burkhardt, C., Fishman, M., Grosvenor, S., Koratkar, A., Ruley, L. and Wolf, K.R., 2000: *Lessons learned from the Scientist's Expert Assistant project*, pub: SPIE, vol. 4010, pp. 107-117. Ed: Quinn, P.J.

Kemeny, 1979: *Report of the President's Commission on the Accident at Three Mile Island*, pub: Nuclear Regulatory Commission, USA,
http://stellar-one.com/nuclear
http://www.pddoc.com/tmi2/kemeny/
http://www.threemileisland.org/resource/index.php?aid=00027.

Kieras D.E., Bovair, S., 1984: *The Role of a Mental Model in Learning to Operate a Device,* in Cognitive Science, vol. 8, pp. 255 - 273.

Kieras D.E., 1988: *Towards a practical GOMS model methodology for user interface design*, in Handbook of Human-Computer Interaction, pp 135 - 158, ed: Helander, M., pub: Elsevier, Amsterdam.

Kieras D.E., 1994: *GOMS Modelling of User Interfaces using NGOMSL*, in CHI '94, pp. 371 - 372.

Koratkar A., Burkhardt, C., Fishman, M., Grosvenor, S., Jones, J.E., Lucas, R.A., Ruley, L. and Wolf, K.R., 2000: *NGST's Scientist's Expert Assistant: evaluation results*, pub: SPIE, vol. 4010, pp. 225-230. Ed: Quinn, P.J.

Koratkar A., Grosvenor, S., Jones, J.E., Li, C., Mackey, J., Neher, K. and Wolf, K.R., 2001: *Code Sharing and Collaboration: Experiences from the SEA Project and their Relevance to the Virtual Observatory*, in International Symposium on Optical Science and Technology - Astronomical Data Analysis, pub: SPIE, vol. 4477, pp. 208 - 215. Eds: Starck, J. and Murtagh, F.D.

Koratkar A., Grosvenor, S., Jones, J., Memarsadeghi, N. and Wolf, K., 2002: *Science Goal Driven Observing: A Step towards Maximising Science Returns and Spacecraft Autonomy*, in Observatory Operations to Optimise Scientific Return III, pub: SPIE, vol. 4844.

Koratkar A., Jones, J.E., Jung, J. and Grosvenor, S., 2004a: *Science Goal Driven Automation for NASA Missions: The Science Goal Monitor,* http://eo1.gsfc.nasa.gov/new/validationReport/index.html#part7 http://aaa.gsfc.nasa.gov/SGM/documents/index.html, accessed 9 July 2006.

Koratkar A., Grosvenor, S., Jung, J., Pell, M., Matusow, D. and Bailyn, C., 2004b: *Science Goal Monitor - science goal driven automation for NASA missions*, pub: SPIE, vol. 5493, pp. 33-41. Ed: Oschmann, J.M.

Koratkar A., Grosvenor, S., Jung, J. and Geiger, J., 2005: *Autonomous Multi-sensor Coordination: The Science Goal Monitor*, in Enabling Sensor and Platform Technologies for Spacebourne Remote Sensing, pub: SPIE, vol. 5659, pp. 293 - 300. Eds: Komar, G.J., Wang, J. and Kimura, T.

Laurel B., 1990: *Art of Human-Computer Interface Design*, pub: Addison-Wesley.

Leveson N., 1994: *Safeware, System Safety and Computers*, pub: Taylor Francis Ltd.

Lewis H., 2000: *Advanced Telescope and Instrumentation Control Software*, in Astronomical Telescopes and Instrumentation 2000, pub: SPIE, vol. 4009, Ed: Lewis, H.

Lidwell W., Holden, K. and Butler, J., 2003: *Universal Principles of Design*, pub: Rockport Publishers Inc.

Malin D., Souccar, K. and Wallace, G., 2004: *Integrating JSky into the Large Millimeter Telescope monitor and control system*, in Advanced Software, Control and Communication Systems for Astronomy, pub: SPIE, vol. 5496, pp. 574 - 581. Eds: Lewis, H. and Raffi, G.

Malin D., Souccar, K. and Wallace, G., 2004: *Rapid prototyping of the Large Millimeter Telescope monitor and control system for effective user interface design*, pub: SPIE, vol. 5496, pp. 590-600. Eds: Lewis, H. and Raffi, G.

Microsoft, 2004: *Official Guidelines for User Interface Developers and Designers*, pub: Microsoft. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/welcome.asp.

Miller G.A., 1956: *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information,* in The Psychological Review, 1956, vol. 63, pp. 81 - 97.

Moss T.H.R., Sills, D.C., 1981: *The Three Mile Island Nuclear Accident: lessons and implications,* in Annals of the New York Academy of Sciences, vol. CCCLXV (365).

Mullet K., Sano, D., 1995: *Designing Visual Interfaces : Communication Oriented Techniques*, pub: Prentice Hall.

Myers M.D., 1997: *Qualitative Research in Information Systems,* in MIS Quarterly, vol. 21:2, iss. June 1997, pp. 241 - 242.

Myrddin Davies L., 1979: *The Three Mile Island Incident*, pub: Atomic Energy Technical Unit.

NASA, 1995: *NASA Systems Engineering Handbook*, http://ldcm.nasa.gov/library/Systems_Engineering_Handbook.pdf, accessed 9 July 2006.

NASA, 1998: *Lewis Spacecraft Mission Failure Investigation Board - Final Report*, http://klabs.org/reports.htm, Some implications on Faster, Better, Cheaper.

NASA/ESA, 1998: *SOHO Mission Interruption Joint NASA/ESA Investigation Board - Final Report*, http://umbra.nascom.nasa.gov/soho/SOHO_final_report.html, accessed 9 July 2006.

Neumann P.G., 1995: *Computer Related Risks*, pub: Addison-Wesley.

Nielsen J., 1989: *Coordinating User Interfaces for Consistency*, pub: Morgan Kaufman, San Francisco. http://www.useit.com/jakob/constbook.html.

Nielsen J., 1990: *A Meta-Model for Interacting with Computers,* in Interacting with Computers, vol. 2, pp. 147 - 160.

Nielsen J., 1993: *Usability Engineering*, pub: Morgan Kaufmann.

Nielsen J., Mack, R.L., 1994: *Usability Inspection Methods*, pub: John Wiley & Sons, New York. http://www.useit.com/jakob/inspectbook.html, accessed 9 July 2006.

Nielsen J., 1994: *Heuristic Evaluation*, in Usability Inspection Methods, eds: Nielsen, J. and Mack, R.L., pub: John Wiley & Sons, New York.

Nielsen J., 2001: *Papers and Essays by Jakob Nielsen*, http://www.useit.com/papers/, accessed: 9 July 2006.

Norman D.A., 1983a: *Some Observations on Mental Models*, in Mental Models, ed: Gentner, D., Stevens, A. L., pub: Lawrence Erlbaum Associates, Hillsdale.

Norman D.A., Draper, S.W., 1983b: *Design Rules Based on the Analysis of Human Error,* in Communications of the ACM, vol. 26, iss. 4, pp. 254 - 258.

Norman D.A., 1983c: *Design Principles for Human Interfaces*, vol. Proceedings of CHI '83, pp. 1 -10.

Norman D.A., 1988: *The Psychology of Everyday Things*, pub: MIT.

Norman D.A., 1999: *The Invisible Computer*, pub: MIT Press.

Norman D.A., 2002: *The Design of Everyday Things*, pub: Basic Books.

Norris P., 1994: *Houston! You never mentioned the boulders*, in New Scientist, 16 Jul 94.

Orlikowski W.J., Baroudi, J.J., 1991: *Studying Information Technology in Organisations: Research Approaches and Assumptions,* in Information Systems Research, vol. 2, iss. 1991, pp. 1 - 28.

Orszag P., Orszag, J., 2000: *Notes on the Florida Vote in the 2000 Election*, http://www.sbgo.com/election.htm, accessed: 7 July 2006.

Page M., 2005: *Teaching*: There is concern that not providing opportunities for students studying astronomy to actually use real telescopes in inaccessible locations will cause a failure of imagination, and a loss of interest. Personal communication.

Parsons H., 1970: *Man-Machine System Experiments*, pub: John Hopkins University Press.

Payne S.J., Squibb, H.R. and Howes, A., 1990: *The nature of Device Models: The Yoked State Space Hypothesis and Some Experiments with Text Editors,* in Human-Computer Interaction, vol. 5, pp. 415 - 444.

Peñataro R., Filgueira, J.M., Gómez-Cambronero, P., González, M. and Pi, M., 2000: *The application of CORBA to the GTC control system*, in Advanced Telescope and Instrumentation Control Software, pub: SPIE, vol. 4009, pp. 152 - 166. Ed: Lewis, H.

Perrow C., 1985: *Normal Accidents*, pub: Basic Books.

Pheasant S., 1996: *Bodyspace: Anthropometry, Ergonomics and the Design of the Work*, pub: CRC Press.

Polson P.G., Lewis, C., Rieman, J. and Wharton, C., 1992: *Cognitive Walkthrough: a method for theory-based evaluation of user interfaces,* in International Journal of Man-Machine Studies, vol. 36, pp. 741 - 773.

Preece J., Rogers, Y., Sharp, H. and Benyon, D., 1994: *Human-Computer Interaction*, pub: Addison-Wesley.

Preece J., Rogers, Y. and Sharp, H., 2002: *Interaction Design: beyond human-computer interaction*, pub: Wiley, http://www.id-book.com.

Pressman, Rogers, 1992: *Software Engineering: A Practitioner's Approach*, pub: McGraw Hill.

Puerta A.R., Cheng, E., Ou, T. and Min, J., 1999: *MOBILE: User-Centred Interface Building*, in CHI 99.

Puxley P., 2006: *Observing (Phase II) Tool - Gemini*, http://www.gemini.edu/sciops/OThelp/otIndex.html.

Quek F.K.H., 1994: *Eyes in the Interface,* in Image and Vision Computing, vol. 13, iss. 6 pp. 511 - 525.

Quinn P.J., 2000: *Observatory Operations to Optimise Scientific Return*, in Astronomical Telescopes and Instrumentation 2000, vol. Proc. of SPIE Vo. 4010, Ed: Quinn, P.J.

Rando N., 2005: *Solar Orbiter Assessment Study*, pub: ESA, SCI-A/2005/023/NP.

Rapoport R.N., 1970: *Three Dilemmas in Action Research,* in Human Relations, vol. 23, iss. 6, pp. 499-513.

Raskin J., 2000: *The Humane Interface: New Directions for Designing Interactive Systems*, pub: Addison Wesley.

Reilly N.B., 1993: *Successful Systems Engineering for Engineers and Managers*, pub: Van Nostrand Reinhold.

Rheinfrank J., Evenson, S., 1996: *Design Languages*, pub: Addison-Wesley.

Riggs W., Hippel, E.v., 1994: *Incentives to innovate and the sources of innovation: the case of scientific instruments,* in Research Policy, vol. 23, iss. July, pp. 459-469.

Robertson S., Robertson, J., 2006: *Mastering the Requirements Process*, pub: Addison-Wesley, Upper Saddle River, NJ.

Rogovin M., 1980: *Three Mile Island: A report to the Commissioners and to the Public (The Rogovin Report)*, pub: United States Regulatory Commission Special Inquiry Group, USNRC Report NUREG/CR-1250-V, http://www.threemileisland.org/resource/index.php?aid=00027.

Rosove P., 1967: *The Development of Computer-based Information Systems*, pub: J Wiley.

Rosson M.B., Carroll, J.M., 2001: *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*, pub: Morgan Kaufmann.

RSI Inc., 2006: *IDL webpage*, http://rsinc.com/idl/.

Sackman H., 1967: *Computers, System Science and Evolving Society*, pub: J Wiley.

Sasse A., 1997: *Eliciting and Describing Users' Models of Computer Systems*, PhD thesis, UCL Computer Science.

Shannon C.E., 1948: *A Mathematical Theory of Communication,* in The Bell System Technical Journal, vol. 27, iss. July, October 1948, pp. 379 - 423, 623 - 656.

Shneiderman B., 1980: *Software Psychology: Human Factors in Computer and Information Systems*, pub: Winthrop Publishers.

Shneiderman B., 1998: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, pub: Addison-Wesley.

Shneiderman B., 2002: *Leonardo's Laptop*, pub: MIT Press.

Simon H., Newell, A., 1958: *Heuristic Problem Solving: The Next Advance in Operations Research,* in Operations Research, vol. 6.

SolarSoft, 2006: *SolarSoft*, http://www.lmsal.com/solarsoft/.

SourceForge, 2006: *SourceForge*, http://sourceforge.net.

Space Telescope Science Institute, 2006: *Astronomer's Proposal Tool*, http://www.stsci.edu/hst/proposing/apt, accessed 30 May 2006.

Stanton N., 1994: *Human Factors in Alarm Design*, pub: Taylor Francis Ltd., London.

Steele I.A., 1998: *An Object Model of the Liverpool Telescope,* in Telescope Control Systems lll, Proc SPIE, vol. 3551, p. 343.

Steinberg D.H., Cheshire, S., 2005: *Zero Configuration Networking: The Definitive Guide*, pub: O'Reilly, http://www.oreilly.com/catalog/bonjour/index.html#details.

Stevens R., Brook, P., Jackson, K. and Arnold, S., 1998: *Systems Engineering: Coping with Complexity*, pub: Prentice Hall.

SunDeveloperNetwork, 2005: *Java Look and Feel Design Guidelines*, http://java.sun.com/developer/techDocs/hi/hig_outline.html.

Sutherland J., 1978: *Systems Science and Social Integrity,* in IEEE Transactions on Systems, Man and Cybernetics, iss. December, pp. 837-848.

Thovtrup H., Nielsen, J., 1991: *Assessing the Usability of a User Interface Standard*, http://www.useit.com/papers/standards.html.

Tognazzini B., 1992: *Tog on Interface*, pub: Addison-Wesley.

Tognazzini B., 2003: *First Principles of Interface Design*, http://www.asktog.com/basics/firstPrinciples.html, accessed: 14 June 06.

Tufte E.R., 1990: *Envisioning Information*, pub: Graphics Press.

Tufte E.R., 1997: *Visual Explanations: Images and Quantities, Evidence and Narrative*, pub: Graphics Press.

Tufte E.R., 1997: *Visual and Statistical Thinking: Displays of Evidence for Making Decisions*, pub: Graphics Press.

Tufte E.R., 2001: *The Visual Display of Quantitative Information*, pub: Graphics Press.

Turing A., 1950: *Computing Machinery and Intelligence,* in MIND (the journal of the Mind Association), vol. LIX, iss. 236, pp. 433-460.

W3C, 2004: *Resource Description Framework*, http://www.w3.org/RDF/, accessed 8 June 06.

Wand J.N., Shots, K.W., Sekhon, J.S., Mebane, W.R., Herron, M.C. and Brady, H.E., 2001: *The Butterfly Did It: The Aberrant Vote for Buchanan in Palm Beach County, Florida,* in American Political Science Review, iss. December 2001.

Webb J.E., 1970: *Space Age Management*, pub: McGraw-Hill.

Weinschenk S., Jamar, P. and Yeo, S.C., 1997: *GUI Design Essentials*, pub: John Wiley.

Weiss K.A., Leveson, N., Lundqvist, K., Farid, N. and Stringfellow, M., 2001: *An Analysis of Causation in Aerospace Accidents*, in Digital Avionics Systems, pub: IEEE, vol. 1, pp. 4A3/1 - 4A3/12.

Weizenbaum J., 1976: *Computer Power and Human Reason - from Judgement to Calculation*, pub: Freeman and Co.

Welch P., Austin, P., 2004: *CSP for Java*, http://www.cs.kent.ac.uk/projects/ofa/jcsp/explain.html, accessed: 15 June 06.

Whitefield A., Wilson, F. and Dowell, J., 1991: *A Framework for Human Factors Evaluation,* in Behaviour and Information Technology, vol. 10, pp. 65 - 79.

Winograd, Flores, 1986: *Understanding Computers and Cognition - A New Foundation for Design*, pub: Ablex Publishers.

Wolf K., Lee, C., Jones, J., Matusow, D., Grosvenor, S. and Koratkar, A., 2000: *The Scientist's Expert Assistant Simulation Facility*, in Astronomical Data Analysis Software and Systems, pub: Astronomical Society of the Pacific, vol. 238, Eds: Harnden, F.R., Primini, F.A. and Payne, H.E.

Wolf K.R., Burkhardt, C., Mark, Fishman, M., Grosvenor, S., Jones, J.E., Koratakar, A. and Ruley, L., 2000: *Expert system technology in observing tools*, pub: SPIE, vol. 4010, pp. 211-219. Ed: Quinn, P.J.

WoTUG, 2003: *CSP*, http://www.wotug.org/csp.shtml, accessed: 15 June 06.

Appendix A: Exploratory Programming

This appendix contains sample results obtained by running the spectrometer application described in Chapter Two section 3, and also contains a description of the contents of the CD attached to the rear of this thesis.

1 Sample Results

The path for storing log files such as the one shown here is currently fixed to be in the directory ‘alec_sim’ at the root level of the start-up disk drive. The directory will be created if it does not exist.

<info>
15:51:11, 02 Jun 2006
</info>
<user>
Carl
</user>
<control>
go
Elapsed: 0 s
</control>
<control>
cal on
Elapsed: 2 s
</control>
<control>
drawLine
Elapsed: 3 s
</control>
<control>
expose
Elapsed: 13 s
</control>
<info>
spinInitCal:4 calAdjust:2
</info>
<control>
expose
Elapsed: 23 s
</control>
<info>
spinInitCal:4 calAdjust:0
</info>
<control>
time change
Elapsed: 23 s
</control>
<control>
expose
Elapsed: 27 s
</control>
<info>
spinInitCal:4 calAdjust:0
</info>
<control>
time change
Elapsed: 27 s
</control>
<control>
expose
Elapsed: 29 s
</control>
<info>

```
spinInitCal:4 calAdjust:0
</info>
<control>
time change
Elapsed: 29 s
</control>
<control>
drawHist
Elapsed: 33 s
</control>
<control>
expose
Elapsed: 36 s
</control>
<info>
spinInitCal:4 calAdjust:0
</info>
<control>
drawLine
Elapsed: 61 s
</control>
<control>
expose
Elapsed: 66 s
</control>
<info>
spinInitCal:4 calAdjust:0
</info>
<control>
time change
Elapsed: 66 s
</control>
<control>
drawHist
Elapsed: 69 s
</control>
<control>
expose
Elapsed: 71 s
</control>
<info>
spinInitCal:4 calAdjust:0
</info>
<info>
user timeout
</info>
<info>
user timeout
</info>
<info>
user timeout
</info>
<control>
done
Elapsed: 183 s
</control>
<info>
User results
</info>
<answer>
No.   W set   H set   W user   H user
(0)   384     123     383      105
(1)   390     132     390
(2)   398     140
(3)   407     154
(4)   434     168
(5)   487     176
</answer>
<control>
restart
Elapsed: 193 s
</control>
```

2 CD Contents

The CD attached to the back cover of this thesis contains the spectrometer application described in Chapter Two section 3, including all the source and binary files created during application development. Instructions for running the application are in Chapter Two section 3.7, and are included on the CD as well. On suitable platforms simply double clicking the application will cause it to run.

Three web browser files allow a demonstration of the application in a browser window, and give convenient links to download the Java and Eclipse packages.

2.1 Description of Files

InstrAp4.jar: The runnable application (actually a Java archive).

application_information: A directory containing the experimenters' descriptive pages, and a technical note (simulator programming.rtf) containing the relevant text from Chapter Two.

eclipse_development_files: A directory containing all the files for developing the application.

eclipse_website.html: link to Eclipse web page.

java_download.html: link to Java web page.

simulator_web_page.html: demonstration of the application in a browser window. The application needs to be in the same directory as this file.

The disc also contains the Adobe Acrobat pdf file from which this document was printed. Contents and bibliography hyperlinks are enabled.

Appendix B: Interview Details

This appendix contains a model interview as an example of how the data in Chapter Four might have been gathered, followed by summaries of the results of those interviews and the review data used.

1 Model Interview

The persona approach used in Chapter Four requires interviews with a range of users of science instruments. The interview style needs to be one that lets the subject tell the situation as they understand it, with as little influence from the interviewer as possible. This is the principle of an ethnographic approach, with the interviewer immersed in the environment as an observer. With little experience of anything similar, it seemed appropriate to sketch out a possible discourse first. The following is the result of that exercise, with subsequent small additions to reflect the course of the actual interviews. Some initial speculative replies are shown (in italics) to aid the flow.

1.1 The Interview

Hi - I'm doing this work on user interfaces in space science and I'm trying to understand the work that you need to do to put a proposal together, assess another proposal, or submit a plan for observation. I've a list of questions but they're really only a starting point. What I'm interested in is understanding what you're doing, without making any false assumptions. I might well ask you to explain stuff that we both know is obvious, just to create a full picture. What I'm keen to do is to look at the problems and understand them, rather than looking at the solutions that have been used in the past. And please, if you think I'm asking something unreasonable, just say so.

I'm going to need to make some notes as we go, so please excuse me for writing whilst you're talking.

OK ...

So - how do you decide on the instrument to bid for time on?

The best would be (this one) but that's usually so busy you get bounced. So try (another one) which is almost as good, if you know how to use it.

Wait a moment - how do you learn?

By using it. Had a good teacher, also. The manual was too difficult to understand.

Why is (this one) better than (the other)?

Better resolution - more modern instrument.

So what type of work do you want to do?

I need to look at an active region of the Sun, and measure line shifts to estimate velocities of material.

Can you explain that in a bit more depth - assume I know nothing very much! Maybe why velocities are important, also?

Er... line shifts mean the line spectra from hot gases show characteristic lines at well known points. If those points move then there must be doppler shift caused by relative movement. Greater movement implies greater velocity for the mass ejection.

How do you cope when the propagation of the gases is not directly towards the observer? Won't that give you a velocity reading that is too low?

You have to image the event at the same time and work out the direction offset. It's a bit tricky...

What else do you have to do to, say to give a good accuracy to the measurements?

(.....)

What's a real challenge to make work?

(.....)

Do you enjoy it when that happens?

What are other really good spots? And bad times?

Is there an example that you can think of that's a bit unusual?

--

Let's turn to refereeing. What do you look for in a proposal?

Does it rely upon you having experience of the facility that the proposer is planning to use?

What additional resource might help you in assessing proposals?

How do you assess how much the proposal is simply repeating existing work? What sets the limits?

Do you use a template or score sheet method?

--

Can you show me the planning tool for instrument (this one) in operation?

What do you like most about it?

And what do you like least?

Is there anything which really doesn't work at all? What makes you mad?

Do you have to do any work-arounds?

Can we do the same with another couple of planning tools? (Same questions)

How do the tools compare?

How long might it take typically to plan a proposal using these tools?

Which is more interesting / enjoyable to use - a flight instrument or a ground-based one?

How do you feed back any ideas you might have to the designers of the equipment?

What feature would you most like to see in an existing instrument that you are familiar with?

Do you wish you had more influence over the design process?

What takes the most time in preparing a paper? Is it planning, background research, observation, analysis, inspiration, writing...?

What else relevant to this whole planning or review process have I missed?

(and so on)

2 Results of Interviews

The following tables summarise the results from the interviews carried out for this work. Text excerpts allow traceability back to the original interview notes, which are not reproduced here. Table 6 is the source of the review data included with the interview plots in Chapter Four.

Abbreviations used:

comm - community	instr - instrument
des - design	obs - observing
dev - development	post-com - post commission

Ref	Behavioural Variable	Interview Text	Group
4	consider rewrite of legacy software (support - resist)	opportunity should have been taken to start from scratch	des & dev
6	consortium feedback to developer (vital - unconcerned)	little feedback from most of team	des & dev
7	development programme (get job finished - add more features)	fear of big job most of which might not be used	des & dev
8	display style (cluttered if necessary - clear layout with fewer items)	specifically chose clutter	des & dev
11	interested in work (do extra if needed - do minimum)	need money for sabbaticals and training	des & dev
12	isolate technical decisions from political influence (important - unconcerned)	engineering decisions ... taken on a non-technical basis	des & dev
14	modify specification (never change once started - accept late changes)	instrument programme control undefined for ground system	des & dev
17	organisational control to stop late changes (need it - unconcerned)	changes do not need consortium wide approval	des & dev
18	personal involvement at instrument inception (strong wish - not concerned)	not asked to contribute to design phase	des & dev
20	resources for training developers in new methods (vital - unconcerned)	need money for sabbaticals and training	des & dev, dev (post com)
22	team working (collaboration - individualism)	scientists must adopt a team approach	des & dev
23	time to produce development tools (vital - unconcerned)	needed various toolkits for development	des & dev

Table 1 Extraction of Behavioural Variables - Bill

Ref	Behavioural Variable	Interview Text	Group
2	attitude to management (management as authority - management as facility)	control freak mentality from one side	design & dev
5	consideration of others (try to achieve consensus - take own view only)	system engineer made hardware dominant	des & dev
6	consortium feedback to developer (vital - unconcerned)	communications across Atlantic not good	des & dev
11	interested in work (do extra if needed - do minimum)	good things involve design, creativity...	des & dev
12	isolate technical decisions from political influence (important - unconcerned)	pressure not to show filter as a separate system	des & dev
13	keep self-motivated (big picture - detail)	good things involve design, creativity	des & dev
18	personal involvement at instrument inception (strong wish - not concerned)	not involved at start of programme	des & dev
22	team working (collaboration - individualism)	need trusted human relationship	des & dev
23	time to produce development tools (vital - unconcerned)	schedule the build of a simulator	des & dev
24	training (action - reading manual)	doing, not reading the manual	des & dev, system test
25	trust in others (full - none)	need trusted human relationships	des & dev, dev (post com)

Table 2 Extraction of Behavioural Variables - Helen

Ref	Behavioural Variable	Interview Text	Group
1	astronomy work (paper publishing - observing & research)	ground-based observing as a student	obs scientist
9	independent way of working (important - unconcerned)	way of working gives real independence	obs scientist, science comm
10	interested in design (involved - leave to others)	discussion about using, not developing	obs scientist
11	interested in work (do extra if needed - do minimum)	get excited about the subject	obs scientist
15	obtain results (simple plan, some results - complex, risk of no results)	proposal only needs to be at a simple level	obs scientist
16	operation of new instrument (take the time to learn - use existing methods)	users typically a bit conservative	obs scientist
19	resolution of data (better spatial, poorer temporal - better temporal, poorer spatial)	instruments often concentrate on spatial features	instr scientist obs scientist
21	teaching (vocation - distraction)	get excited about the subject	teaching & public
22	team working (collaboration - individualism)	collaboration can involve 4- 26 people...	obs scientist
26	usability development (spend time on interface design - rely on printed manual)	discoverable features	obs scientist
28	why do it (be first to discover something new - just a job)	chance to say ‘I did it’	obs scientist

Table 3 Extraction of Behavioural Variables - Jane

Ref	Behavioural Variable	Interview Text	Group
1	astronomy work (paper publishing - observing & research)	was name on paper, now just doing the research	instr scientist, obs scientist
3	calibration (rely on others - do it personally)	prefer raw data	obs scientist
6	consortium feedback to developer (vital - unconcerned)	planning tool would be appreciated early in programme	instr scientist
7	development programme (get job finished - add more features)	doesn't stop people wanting to change them	instr scientist
11	interested in work (do extra if needed - do minimum)	the buzz of space science...	science community
14	modify specification (never change once started - accept late changes)	appreciate need for firm specifications	instr scientist
15	obtain results (simple plan to guarantee some results - complex plan with risk of no results)	simple plan immediately after launch	obs scientist
17	organisational control for changes (need it - unconcerned)	...doesn't stop people wanting to change them	instr scientist
18	personal involvement at inception (strong wish to take part - unconcerned)	firm specifications are written in science requirements	instr scientist
22	team working (collaboration - individualism)	high degree of individualism	instr scientist

Table 4 Extraction of Behavioural Variables - Kevin

Ref	Behavioural Variable	Interview Text	Group
1	astronomy work (paper publishing - observing & research)	both instrument and science question are important	obs scientist
3	calibration (rely on others - do it personally)	always have scheduler in direct control	obs scientist
9	independent way of working (important - unconcerned)	learn to use a particular instrument, ask people	obs scientist, science comm
10	interest in instrument design (involved - leave to others)	no strong interest	obs scientist
11	interested in work (do extra if needed - do minimum)	first in discovering something new	obs scientist
26	usability development (spend time on interface design - rely on printed manual)	no preference	obs scientist
27	use of detector (best performance - longest life)	operator control	operator
28	why do it (be first to discover something new - just a job)	discover something new	obs scientist

Table 5 Extraction of Behavioural Variables - Nick

Ref	Behavioural Variable	Reasoning
1	astronomy work (paper publishing - observing & research)	(none)
2	attitude to management (management as authority - management as facility)	*funding body* - exert appropriate authority
3	calibration (rely on others - do it personally)	(none)
4	consider rewrite of legacy software (support - resist)	*instr scientist* - keen to re-use material *science comm* - prefer updated facility
5	consideration of others (try to achieve consensus - take own view only)	(none)
6	consortium feedback to developer (vital - unconcerned)	(none)
7	development programme (get job finished - add more features)	*dev (post-comm)* - after launch development, wants to finish job
8	display style (cluttered if necessary - clear layout with fewer items)	*system tester* - needs all on screen *obs scientist* - prefers clear layout *operator* - needs good selection
9	independent way of working (important - unconcerned)	(none)
10	interest in instrument design (involved - leave to others)	*instr scientist* - involved in detail *des & dev* - extremely involved
11	interested in work (do extra if needed - do minimum)	(none)
12	isolate technical decisions from political influence (important - unconcerned)	*instr scientist* - sees both sides
13	keep self-motivated (understand big picture - concentrate on fine detail)	*des & dev* - tends towards fine detail
14	modify specification (never change once started - accept late changes)	(none)
15	obtain results (simple plan to guarantee some results - complex plan with risk of no results)	(none)

Table 6 Extraction of Behavioural Variables - Review Category

Ref	Behavioural Variable	Reasoning
16	operation of new instrument (take the time to learn - use existing methods)	*operator* - will learn any new methods
17	organisational control to stop late changes (need it - unconcerned)	(none)
18	personal involvement at instrument inception (strong wish - not concerned)	(none)
19	resolution of data (better spatial, poorer temporal - better temporal, poorer spatial)	(none)
20	resources for training developers in new methods (vital - unconcerned)	*instr scientists* - tend to have poor visibility of development detail. *fund body* - rarely have contact with developers
21	teaching (vocation - distraction)	(none)
22	team working (collaboration - individualism)	*instr scientists* - prefer to agree specification and then leave alone
23	time to produce development tools (vital - unconcerned)	*instr scientists* - little visibility of need. *fund body* - no visibility of need.
24	training (action - reading manual)	*obs scientist* - by mixture of reading and doing.
25	trust in others (full - none)	*dev (post-comm)* - relies on competence of previous work
26	usability development (spend time on interface design - rely on printed manual)	*instr scientist* - usability seen as low priority. *des & dev* - usability rarely funded. *fund body* - usability not in package.
27	use of detector (best performance - longest life)	*instr scientist* - wants performance & life. *des & dev* - better understanding of how to safely maximise performance.
28	why do it (be first to discover something new - just a job)	*science comm* - generally enthusiastic. *fund body* - enthusiasm at a distance.
29	care of instrument (exceptional - adequate)	*des & dev* - reasonable, but can mend it. *operator* - not serviceable in orbit. *dev (post-comm)* - typically good understanding of operating margins.

Table 6 Extraction of Behavioural Variables - Review Category

Ref	Behavioural Variable	Reasoning
30	financial accuracy (precise - minimum necessary)	*instr scientist* - want to build the instrument. *fund body* - appreciate finite pot
31	finished instrument (best in class - minimum to meet specification)	*instr scientist* - wants best possible. *fund body* - best possible in budget. *system tester* - best in time available.
32	testing (basic test all features - thorough test selected features)	*des & dev* - accurate testing maybe limited by over familiarity. *system tester* - prioritise testing critical parts.
33	attention to detail (good - bad)	*instr scientist* - wants overall performance. *des & dev* - has to make every last bit work correctly
34	public outreach (important - don't care)	*instr scientist* - affect possible future funding. *des & dev* - neutral. *obs scientist* - appreciate some importance. *fund body* - fairly important
35	reporting & communication (thorough - minimal)	(none)
36	schedule progress (cautious - lax)	(none)
37	support astronomy funding (for - against)	(none)

Table 6 Extraction of Behavioural Variables - Review Category

Appendix C: Full Persona

1 Full Persona Descriptions

If we were to proceed to a full design of a user interface, it would be appropriate here to include a narrative style description of each persona based on the bullet points above. The rationale of this is the power of narrative to serve as a tool for generating and validating design ideas {Rheinfrank, 1996 #138}, and {Cooper, 1999 #28}. One description is included here just as an illustration; the details are not used elsewhere in this discussion. They would need to be seen to be accurate and balanced against the other personas for a full interface design. A photograph is regarded as an important part of generating the mental image that allows the persona to be treated as representative.

1.1 Pre-Launch Technologist

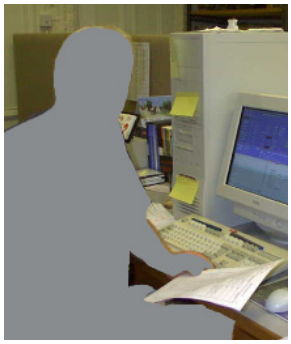


Fig. 1 Photo

David Jones is a 35 year old self-confessed computer addict who loves the opportunity to earn a living whilst doing work that interests him. Somehow his wife and young son put up with all the extra hours he works. He is keen to learn about all aspects of space flight and likes to be technically challenged. His best reward would be praise from his science colleagues for helping to produce a first class instrument and knowing that other people were keen to use his ideas in a future mission. He likes the way his boss runs the team in a friendly cohesive manner and always is open and approachable. He is also glad someone else is there to deal with the financial side of the work, which he finds particularly uninteresting. One particular wish that he has is to be able to point out to his son a moving light in the night sky and tell him that “it’s one of mine!”

Appendix D: Interface Assessment Tables

This appendix contains blank copies of the interface assessment tables used in Chapter Five to judge the science and engineering interfaces. It includes the criteria in the full list developed in Chapter Four section 2 which have been removed as they are not relevant to the science or engineering assessments. It also includes for completeness a table of those operator-related criteria that are not used in either of the other two tables. This may be useful in assessing the organisational side of an instrument programme, but otherwise is outside the scope of this thesis. Blank polar plots from Chapter Five are included as well (Fig. 1 and Fig. 2).

This appendix also contains another copy of the usability requirements check list from Chapter Eight, in order to present this set of forms in one place.

1 Science Interface assessment

(Table 1)

The following are removed from the full list in Chapter Four section 2 for the science interface assessment:

'avoid parallax problems' as it only applies to mechanical controls;

'ensure realistic workload', 'avoid distraction particularly in an emergency', 'Education', 'Environment' and 'Teamwork' groups as they are management issues outside the scope of this analysis;

the following as they are only relevant for the engineering interface:

allow command authority with visibility for all; consider all users;
allow resources for extra operators; keep system logs and make them available;
give full feedback of delayed response entry; allow audible alarms to be distinguishable;
minimise risk of inadvertent control operation; provide protection from unauthorised users;
identify hazardous operation; signal abnormal operation clearly;
allow limits to be exceeded in an emergency; use interlocks to minimise risk;
implement system redundancy if possible; avoid operator over-confidence.

2 Engineering Interface Assessment

(Table 2)

The following are removed from the full list in Chapter Four section 2:

‘avoid parallax problems’, as it only applies to mechanical controls;

‘ensure realistic workload’, ‘avoid distraction particularly in an emergency, ‘Education’, ‘Environment’ and Teamwork’ groups as they are management issues outside the scope of this analysis.

3 Operator Aspects Assessment

(Table 3)

These are the management criteria (from Chapter Four section 2.5) described above that do not appear on any other assessment table, and are tabulated here for completeness.

4 Usability Requirements Check

(Table 4)

This table is a repeat of Table 1 in Chapter Eight.

Parameter (Science Interface)	**Compliance Good -5 None - 0**	**Reasoning**
Interaction - Accessibility: 2.1.1		
place controls according to usage		
group controls and their displays		
group controls by function or sub-system		
allow easy discovery of control limits		
provide contextual help for data entry & controls		
use the most appropriate interface devices		
Interaction - Adaptability: 2.1.2		
allow user to customise their system view		
allow for different levels of experience		
allow parameters to be joined algorithmically		
Interaction - Control: 2.1.3		
ensure user is in control		
anticipate possible user actions		
Representation: 2.2		
take cultural conventions into account		
make controls & indicators self explanatory		
avoid clutter		
use an appropriate information density		
give immediacy of feedback for data entry		
make new control settings visible immediately		

Table 1 Assessment for Science Interface

Parameter (Science Interface)	**Compliance Good -5 None - 0**	**Reasoning**
avoid sensory overload		
create clear mental model		
allow graphics use		
allow use of audible hints		
imitate the physical layout of the system		
include all necessary parameters		
use colour appropriately		
use controls of an adequate size		
Consistency: 2.3		
avoid ambiguity in controls and displays		
ensure layout consistency		
ensure consistency in operation of controls		
use accepted design practice if appropriate		
use public standards for colour & contrast		
Error management: 2.4		
check entries as they are entered		
suggest values to user if computable		
avoid dialogue boxes if possible		
correct errors easily		
Operator Aspects: Workload: 2.5.2		
provide ready filled data fields		
avoid reliance on user memory		

Table 1 Assessment for Science Interface

Parameter (Engineering Interface)	**Compliance Good - 5 None - 0**	**Reasoning**
Interaction - Accessibility: 2.1.1		
place controls according to usage		
group controls and their displays		
group controls by function or sub-system		
allow command authority with visibility for all		
allow easy discovery of control limits		
provide contextual help for data entry & controls		
use the most appropriate interface devices		
Interaction - Adaptability: 2.1.2		
consider all users		
allow user to customise their system view		
allow for different levels of experience		
allow parameters to be joined algorithmically		
allow resources for extra operators		
Interaction - Control: 2.1.3		
ensure user is in control		
anticipate possible user actions		
keep system logs, and make them available		
Representation: 2.2		
take cultural conventions into account		
make controls & indicators self explanatory		
avoid clutter		

Table 2 Assessment for Engineering Interface

Parameter (Engineering Interface)	**Compliance Good - 5 None - 0**	**Reasoning**
use an appropriate information density		
give immediacy of feedback for data entry		
make new control settings visible immediately		
give full feedback of delayed response entry		
avoid sensory overload		
create clear mental model		
allow graphics use		
allow use of audible hints		
allow audible alarms to be distinguishable		
imitate the physical layout of the system		
include all necessary parameters		
use colour appropriately		
use controls of an adequate size		
Consistency: 2.3		
avoid ambiguity in controls and displays		
ensure layout consistency		
ensure consistency in operation of controls		
use accepted design practice if appropriate		
use public standards for colour & contrast		
Error management: 2.4		
minimise risk of inadvertent control operation		
provide protection from unauthorised users		

Table 2 Assessment for Engineering Interface

Parameter (Engineering Interface)	**Compliance Good - 5 None - 0**	**Reasoning**
identify hazardous operation		
signal abnormal operation clearly		
allow limits to be exceeded in an emergency		
use interlocks to minimise risk		
check entries as they are entered		
suggest values to user if computable		
avoid dialogue boxes if possible		
correct errors easily		
implement system redundancy if possible		
avoid operator over-confidence		
Operator Aspects - Workload: 2.5.2		
provide ready filled data fields		
avoid reliance on user memory		

Table 2 Assessment for Engineering Interface

Parameter (Operator Aspects)	**Compliance Good - 5 None - 0**	**Reasoning**
Operator Aspects - Education: 2.5.1		
ensure operator has appropriate education		
allow time to become familiar with system		
provide training appropriate to responsibility		
provide regular assessment of operator		
Operator Aspects - Workload: 2.5.2		
ensure realistic workload		
avoid distraction particularly in an emergency		
Operator Aspects - Environment 2.5.3		
Ensure operators are comfortable		
Allow operators to control their environment		
Operator Aspects - Teamwork 2.5.4		
Create a good team culture		
Create a process for solving disagreements		
Encourage operational transparency		

Table 3 Assessment for Operator Aspects

Usability Requirement	**Compliant**	**References for information** (ch 'n' is in the thesis)
consider layout standards & guides		ch2; (Apple Computer 2005a); (Microsoft 2004); (Sun Developer Network 2005); (BSI 1998); (BSI 1999b)
encourage mental models		ch2; (Norman 1983a); (DiSessa 1986); (Johnson-Laird 1983); (Cooper 2003)
use mapping & alignment		ch2; (Norman 1988)
look at alternative devices and displays apart from keyboard, mouse, screen		ch2
give user feedback to controls		ch2; (Gibson 1977); (Norman 1988)
consider Fitts' and Hick's laws		ch2; (Fitts 1954); (Raskin 2000)
use colour appropriately		ch2; (Tufte 1990); (IEE 2004)
consider overall aesthetics		ch4; (Burkhardt 2000)
look at previous examples of work		ch2
use interaction parameters		ch4
use representation parameters		ch4
use consistency parameters		ch4
use error management parameters		ch4
use operator aspects parameters		ch4
consider science group type & size		ch4
consider operation		ch4
consider timescales		ch4
understand user groups		ch4
understand technical & personal goals		ch4; (Cooper 2003)
use persona analysis to understand usage		ch4; (Cooper 2003)
define communication model		ch4

Table 4 Usability Requirements Check List

Usability Requirement	Compliant	References for information (ch 'n' is in the thesis)
define number of interfaces		ch4
be consistent for usage, including appearance, layout, disadvantaged access		ch5; (Nielsen 1989)
be consistent with software tools, and software modules		ch5
put usability before consistency		ch5; (Nielsen 1989)
do frequent user testing, face to face meetings, formal assessments		ch5; (Nielsen 1989); (Koratkar 2000)
separate the user interface from other software		ch5; (Jones 2000)
consider using artificial intelligence		ch5; (Koratkar 2002)
re-use existing software		ch5; (Koratkar 2000)
use model-view-controller software architecture		ch5; (Fowler 2006)
implement documentation system		ch5; (Koratkar 2000)
allow planning tool to be used in a flexible manner		ch5; (Burkhardt 2000)
use terms familiar to scientists for planning tool		ch5; (Burkhardt 2000)
ensure interfaces are responsive		ch5; (Burkhardt 2000)
provide enough information to make science trade-offs		ch5; (Burkhardt 2000)
create simple installers, preferably by web page		ch5; (Burkhardt 2000)
make software platform independent		ch5; (Burkhardt 2000)
ensure information only requires entering once		ch5; (Burkhardt 2000)
use same tools for scientists and technical operators		ch5; (Burkhardt 2000)

Table 4 Usability Requirements Check List

Usability Requirement	Compliant	References for information (ch 'n' is in the thesis)
group access of any on-line information into one session		ch5; (Burkhardt 2000)
use XML for miscellaneous file storage		ch5; (Burkhardt 2000)
use adaptable software architecture to allow multiple iterations of instrument description		ch5; (Ames 2000)
for instrument autonomy, consider data selection, prioritisation, compression, goal-oriented operation and auto re-survey		ch5; (Koratkar 2002)
use visualisation in tools		ch5; (Burkhardt 2000)
use rapid prototyping method		ch5; (Jones 2000)
consider natural language interface		ch5; (Jones 2000)
ensure available skills match intended technology		ch5; (Koratkar 2004b)
ensure scientist is part of software development team		ch5; (Jones 2000)
ensure good communication in development team		ch5
consider one software language for instrument and for support tools		ch5
ensure simulator closely models scientific response		ch7
ensure simulator tracks instrument calibration		ch7
ensure simulator replicates all instrument systems		ch7
use simulator to track consumables usage and mechanism wear		ch7

Table 4 Usability Requirements Check List

Usability Requirement	Compliant	References for information (ch ‘n’ is in the thesis)
provide formatted data from simulator		ch7
ensure simulator is an integral part of instrument lifecycle and its development precedes the instrument		ch7

Table 4 Usability Requirements Check List

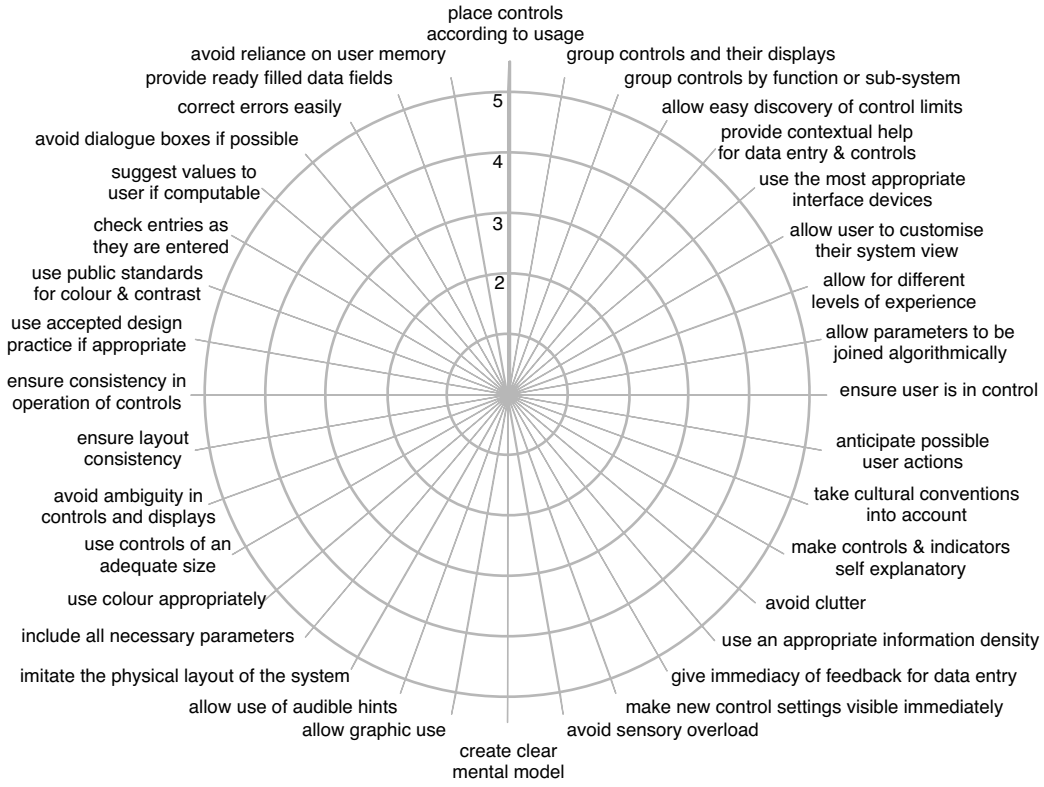


Fig. 1 Science Interface

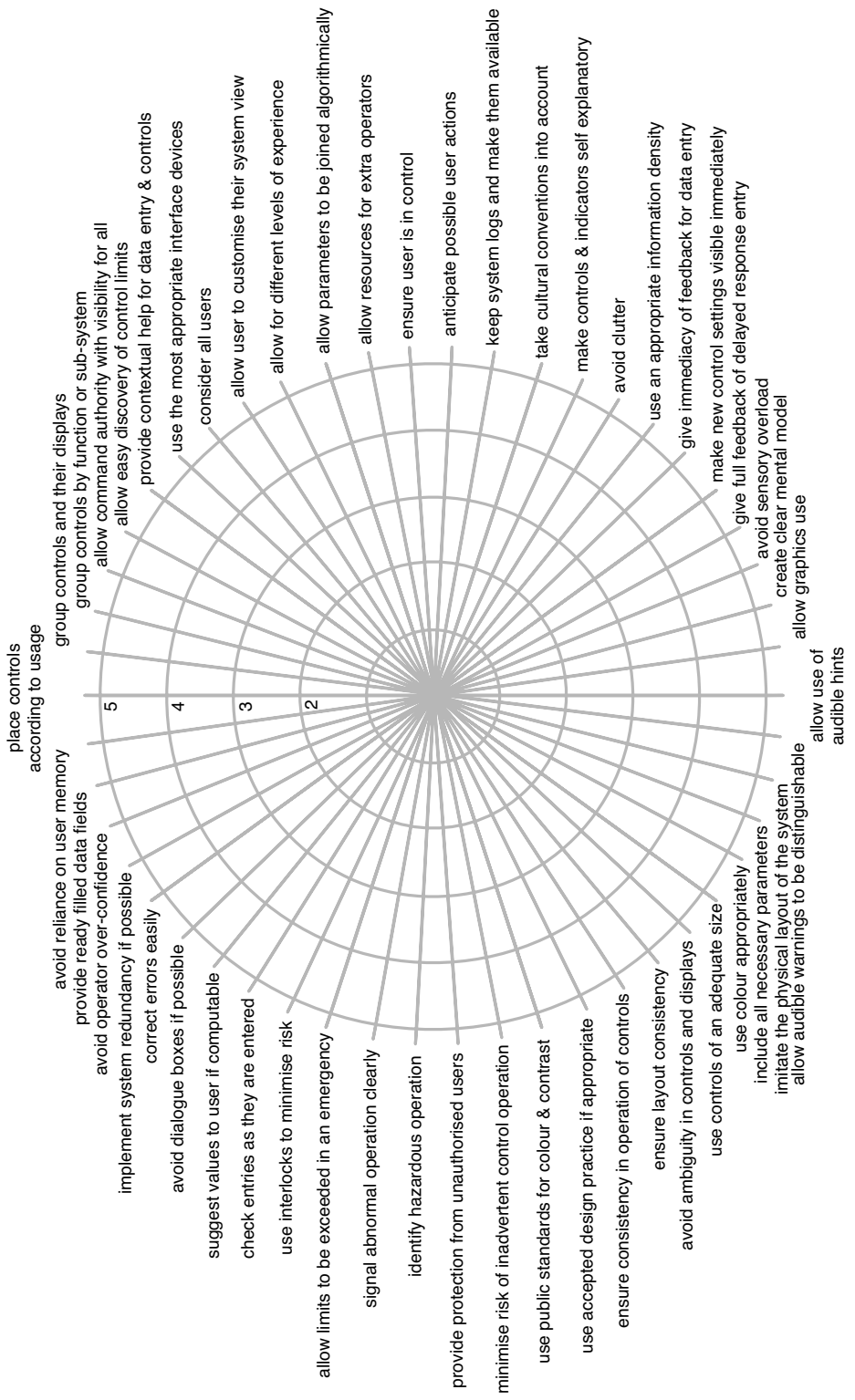


Fig. 2 Engineering Interface