UNIVERSITY
of
GLASGOW

Department of
Computing Science

GIST

Reconnaissance: a widely applicable approach encouraging well-informed choices in computer-based tasks

Aran Edward Lunzer

Submitted for the degree of Doctor of Philosophy

September 1995

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

The aim of this thesis is to propose and demonstrate a novel approach to human–computer cooperation in exploratory tasks, that encourages the pursuit of thorough explorations and thus increases the likelihood of finding the best available results.

The thesis identifies a class of computer-supported tasks—such as artifact design, decision analysis, data presentation, and many kinds of retrieval—that all involve an exploration among alternative result specifications to find one whose outcome best fits the user's current needs. Such explorations cannot be automated, but must be directed by the user. Since there is typically no way of knowing in advance what results are available, the user must continually trade off the effort required for further exploration against the unknown chance of discovering results preferable to those found so far. The more arduous the exploration is perceived to be, the greater the user's temptation to accept early results without due consideration of alternatives.

By using a new *illumination zone model* to analyse existing systems, the root of the problem is characterised as a lack of support for visualising the trade-offs between results in terms of the criteria that the user feels are significant at the time. Reconnaissance—in the familiar sense of using scouts to examine unknown territory—provides a suitable metaphor for the task of instructing a computer to generate a range of results, to summarise them according to the current set of criteria, and to collate these summaries into a single visualisation.

A challenge in supporting computer-based reconnaissance is the provision of a convenient interface that integrates the requesting, viewing and comparison of result summaries. An implemented example of reconnaissance support shows that the *parallel coordinates* plotting technique can be extended into an interactive form that meets this challenge. Tests carried out on this implementation, which provided reconnaissance facilities for an existing document formatting system, confirmed that more thorough explorations were indeed encouraged. The tests also brought to light additional, unforeseen advantages to the use of result ranges and summarisation.

Summary contents

Contents

List of Figures

Acknowledgements

Obviously the major thanks go to Phil Gray and Steve Draper (my supervisors) and Frances Clark (my analyst)—whose roles, predictably, have overlapped quite a bit. The greatest help has been their constant faith and patience; Phil, in particular, has undertaken some epic missions to lead me out of various research quagmires.

My research was funded under EPSRC studentship 91308610, while the pain of giving up a salary in return for a student grant was greatly reduced by the generosity of my father Raph. I am also extremely grateful to my erstwhile IBM manager John Carter for his understanding, interest and assistance both during and since my time in his group, and to John Patterson for helping me find time to finish the writing-up.

The department's computers have been a joy to use, thanks to the care of the technicians and support staff (as well as much personal assistance from Duncan Sinclair). In particular, I can't imagine what I would have done without the Smalltalk-80 VisualWorks environment[1], which made it a pleasure to build my own note-keeping system and the various prototype tools used in this research.

This thesis owes a lot to Alastair Reid and Will Partain for their assistance throughout the implementation of InteracTEX; likewise to all the other test victims. Also to Chris Johnson and the three members of my *viva* panel, for lots of constructive feedback.

The city of Glasgow, its (original) University, and this department have all turned out to be great places and great sources of support and friendship. I couldn't list *all* the people around here who have become important to me, so I'll start by thanking the other members of the night shift (or 24-hour shift)—notably the recently-Doctored Alex Ferguson—for their whacky but reassuring company, and then give special mentions for Trish, Sharon, Mark3, Jackie, Nicola, Niall & Guy, and of course [blush] Susan.

Finally, huge thanks (from a distance) to Raymond M. Smullyan, mathemagician and philosopher, whose Taoist writings help make a lot more sense of Everything.

[1] VisualWorks is a registered trademark of ParcPlace Systems, Inc.

Declaration

The material presented in this thesis is the product of the author's own independent research carried out at Glasgow University, under the supervision of Phil Gray of the Department of Computing Science and Steve Draper of the Department of Psychology.

A description of the idea and implemented proof-of-concept of 'reconnaissance' as defined in this thesis has been published as a conference Technique Note (Lunzer, 1994), and also made available in longer form in a 1995 anthology of research from GIST, the Computing Science department's interactive systems group.

Chapter 1

Introduction

1.1 Motivation

The problem that motivates this thesis might be stated as follows:

> How can a computer support exploration in a large unfamiliar search space, where narrowing the search too quickly based on early impressions might cause many of the best available results to be missed?

1.1.1 Scenario: at the travel agent

Last year I visited a travel agent to book a plane flight from Glasgow to Los Angeles, about a month in advance. Here is an approximate account of what happened:

- Instead of printed timetables, my travel agent (like most others) now uses a computer service that can locate all available seats, on all scheduled routes of all the major airlines, for many months into the future. The system handles a vast store of dynamically updated information, and must serve a high rate of enquiries from booking agencies and airlines around the world. To help keep the responses timely, the interface available to most travel agents will only handle very narrow queries—typically, specifying precisely the departure and destination cities and a single desired date of travel.
- Cost is important to me, and the travel agent and I soon made a significant discovery: my journey fell near a change in pricing season. We discovered that on November 1st

most fares were a full 20% less than their equivalents the day before. This is a discount worth getting, so we decided to concentrate just on November flights.

- The computer system held at least 40 alternative official routes for this journey. But when we looked in detail at the seats that were available on my preferred date we found that the cheapest ones involved lots of intermediate stops, or arrived inconveniently late at night. So we started to explore the alternatives—such as paying slightly more, travelling via different cities in which I have friends or family I could visit, or changing my journey dates slightly. This involved running several extra queries.

  Note that the designers of the query system could not have predicted the set of queries that I decided to run, because the alternative dates and cities that I was prepared to consider were entirely particular to me and to the occasion. Furthermore, some alternatives were only chosen in response to the results I obtained from the earlier queries.

- Eventually I found an available seat, with an acceptable set of compromises on date, price and cities. Many of these factors were far from what I initially requested--so was it really the compromise I would have been happiest to make? In particular, what if I hadn't made the early decision not to look for October flights? Despite the price change, there might happen to have been a seat available on October 31st, on a flight that suited me better than the compromise I ended up accepting.

- But after spending almost half an hour focussing in on the available compromises, the travel agent and I shared a strong distaste for going right back to the beginning and starting on an alternative branch of the exploration—remembering that the date marked a change in season, so there may have been substantially different schedules to be explored.

- So I booked the seat we had found; the transaction was complete. But I was not entirely happy...

1.1.2 The problem: under-informed outcome

The travel agent and I had examined a number of alternative flights, and had arrived at a rationale for evaluating the advantages and disadvantages of each in the context of the desired journey; this took a lot of time and cognitive effort. But despite this effort we knew nothing about the flights just outside the range of our search, and we might have missed a better alternative.

In particular, near the start we narrowed the space of alternatives to be evaluated by making the decision to look only at November flights. The reasoning seemed clear-cut at the time, but with all the compromises we had to make thereafter this original decision became open to question.

The query system's interface strongly encourages a selective form of progress, by which the user can be seen as following a narrow path through the search space. In relation to all the results that would potentially be of interest our eventual choice was an **under-informed outcome**[1]; the decision was based on evaluating a range of candidates that might not have constituted a good representative sample.

Put simply, the risk of an under-informed outcome can be reduced by pursuing a **thorough exploration**. This term should not be taken to imply an *exhaustive* exploration; there may be far too many results to examine and, as is the case in this scenario, most are known to be entirely irrelevant. Instead, the ideal case of a thorough exploration is one in which all *promising* directions of search are pursued, rather than having to rule out some directions for no reason other than a lack of time. Because of the costliness of time itself this ideal might never be reached, but the aim of this thesis is to bring computer users one (significant) step closer.

1.1.3 The activity: opportunistic exploration of a result space

The activity of searching for a flight from a database is just one example of a general class of activities tackled with the help of computers, that is referred to within this thesis as **opportunistic exploration** of a **result space**.

Result space

As used in the thesis, 'result space' is a theoretical concept meaning the enumeration of all possible results that can be produced by a given computer-based system in response to a result specification supplied by a user. This concept is intentionally more general than the bounded form of search space explicitly embodied in a database of flight details, or in a document base accessible through an information-retrieval system. As a radically different example, a computer-assisted architectural design system can be seen as providing access to a potentially infinite result space of building designs that may be articulated by use of the system's facilities.

[1] Terms having specialised meanings within the thesis are introduced in bold type, and are gathered in the glossary on page 232.

An important implication of this concept can be seen as a generalisation of the distinction between indexed and non-indexed properties in a database search: when the user supplies a result specification that expresses certain desired characteristics, the results produced by the computer may have additional properties that are just as important to the user as the ones that were specified. But just as a flight reservation service might not provide any facility for requesting all destinations that can be reached within five hours' flying from Glasgow, so an architectural design system may be unable to produce all apartment designs having less than 50 metres of internal wall. Therefore it is up to the user to direct the computer to reveal parts of the result space; to do this the user must control just those properties that *can* be included in the specification.

Opportunistic exploration

This thesis is concerned with tasks in which a user is approaching a result space with the intention of finding within it one or more results that are particularly suited to that user's current needs. More specifically, the thesis addresses tasks in which, as illustrated in the scenario above: 1) the computer system cannot be designed in advance to predict result-specification details that will be of interest to the user, 2) even the user cannot know in advance what will constitute a good result or a worthwhile compromise between different result properties (whether part of the specification or not), and 3) the user cannot make reliable predictions about the result properties that will be obtained from a given specification. In such tasks the user must therefore engage in an exploration of the result space, which will be opportunistic in that the user decides its course at the time—according to a developing understanding of what the result space holds.

1.2 Thesis structure

The goal of the thesis is to propose and demonstrate a novel approach to human–computer cooperation that alleviates the problem of under-informed outcome in opportunistic exploration of result spaces.

As has already been indicated, opportunistic explorations occur in a wide variety of domains in which computers are applied; a primary concern in pursuing the thesis was therefore to limit the range of domains to which the thesis proposals are aimed to apply. To this end, figure 1.1 emphasises the key role played by chapter 2 in which are defined the framework concepts underlying the thesis: the *human-driven delegation* constraint on human–computer interactions, the *illumination zone model* (developed specifically for the thesis) for analysing

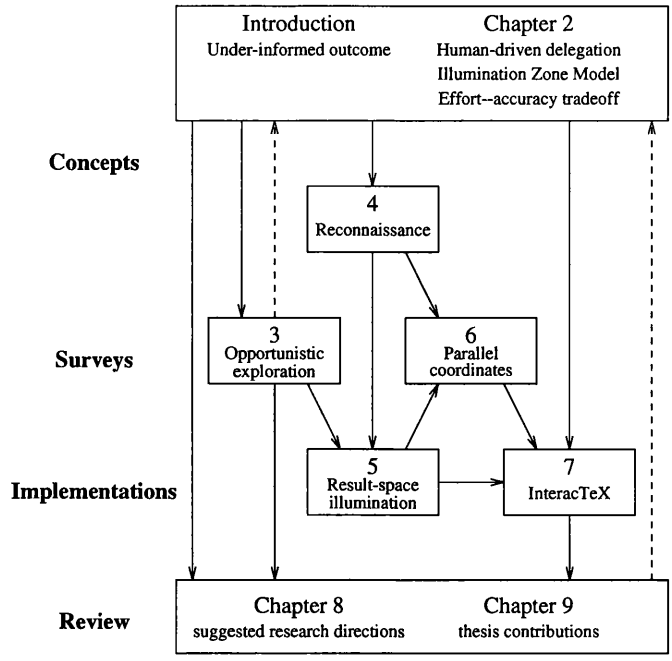


Figure 1.1 Dependency relationships between thesis chapters.

the circumstances leading to under-informed outcome, and the *effort–accuracy tradeoff* that provides guidance on how to influence users to pursue more thorough explorations.

Although the illumination zone model is presented as an *a priori* structure that is then adapted to a range of domain-specific embodiments, its design is largely a reflection of themes that emerged from a survey of a wide range of existing opportunistic-exploration systems. The survey, presented in chapter 3, is thus necessarily more illustrative than critical—one corollary being that it contains more descriptive matter than would a purely critical review.

Chapter 4 injects the main novel concept of the thesis: the *reconnaissance* approach to human–computer cooperation. Reconnaissance is a powerful technique for tackling real-world explorations; chapter 4 illustrates how reconnaissance could improve the effectiveness of one particular form of computer-based search, and then uses the illumination zone model to analyse the general strengths of the approach. The chapter concludes with a set of requirements for successfully applying reconnaissance to opportunistic result-space explorations.

The first major pieces of computer-system design and implementation carried out as part of this research are described in chapter 5. The work is reported as a number of case stud-

ies, examining the applicability of reconnaissance-like search support in various domains. Although none of these individual studies was felt to be a suitable basis for a major evaluation of reconnaissance within the scope of the thesis, many valuable ideas emerged from the design activity. A sub-class of explorations was identified for which reconnaissance could be represented straightforwardly, and could be supported by developing a novel interactive form of the *parallel coordinates* display formalism to address the challenge of letting the user request and compare a large set of summarised results.

Chapter 6 expands on the use of parallel coordinates as a basis for a reconnaissance interface, starting with an extensive survey of the development and properties of the parallel coordinates technique. Since the proposed adaptation was found to be novel, the role of the survey is not so much to critique existing embodiments as to demonstrate the power of parallel coordinates as an underlying representation. It is therefore highly descriptive, and this section may be skipped by any reader who is prepared to take on trust the assertion that the representation offers great potential for data analysis and understanding. The second part of the chapter then addresses the design requirements for the interactive form of parallel coordinates display required to support reconnaissance.

Chapter 7, drawing on the concepts from chapter 2, the findings from the implementation work reported in chapter 5, and the design proposed in chapter 6, describes in detail the analysis, implementation and evaluation work undertaken in adding reconnaissance support to the activity of document-formatting within the LaTeX document preparation system. The chapter describes the iterative development of the reconnaissance framework and the specialised facilities for document formatting, drawing on the experience of existing LaTeX users to progress the ideas from an initial demonstration-level prototype through to the hands-on tutorial that was used in the final round of evaluations. The chapter's conclusions bring together various lessons towards future attempts to implement reconnaissance for this or similar domains, showing the perceived impact on the conduct of the underlying activity, the applicability of the chosen interface design, and a number of areas in which further work appears likely to produce interesting findings.

The final chapters consolidate the thesis contributions. Chapter 8 sketches out some research directions that follow on naturally from this work, while chapter 9 summarises the main findings of the thesis and their significance to the design of computer-based exploration systems.

Chapter 2

Thesis framework concepts

In order to impose some practical bounds on the range of influence claimed for the thesis, this chapter introduces three 'framework concepts' characterising the approach to be taken.

First, the *human-driven delegation* constraint stakes out as the area of interest just those computer-based explorations in which the exploration effort is divided between user and computer, but in which the user is always explicitly in control of the moment-to-moment details of that division.

Second, the *illumination zone model* declares a particular pattern of progress in explorations, and a framework of cooperating components that support such progress. This model was developed specifically for use in the thesis. The model is intended to be general enough to apply to a wide variety of application domains, but to have components with sufficiently specific functions to enable their embodiments to be identified and compared between systems serving diverse domains.

Finally, the *effort–accuracy tradeoff* is what Payne, Bettman and Johnson (1993) propose as a key to human behaviour in decision tasks in which greater 'accuracy' of choice can only be obtained at the cost of greater 'effort' in evaluating the available alternatives. Their framework, described in section 2.3, provides the motivation for the straightforward approach adopted in this thesis: in order to reduce the risk of under-informed outcome, simply provide users with a low-effort means to perform simple but effective evaluations of a large range of alternatives.

2.1 Human-driven delegation

This section declares and explains the restriction of this thesis to considering explorations in which the user is explicitly and actively in control of all delegation of activity to the computer. The argument proceeds in the following stages:

1. Only activities that can be *formalised* to the user's satisfaction should be delegated to a computer. As part of this, it must be clear to the user which of the activities have been delegated.
2. Many tasks necessarily involve a mix of unformalisable and formalisable (and hence delegatable) activities, so the computer system must allow both levels of activity to take place.
3. Opportunistic explorations, in particular, are likely to have the nature of being *semi-formal*—that is, to require the ability to change dynamically the boundary between formal processing by the computer and informal processing by the user.

2.1.1 The user only delegates tasks that can be satisfactorily formalised

The degree to which a computer can be delegated to carry out activities on the user's behalf hinges on the extent to which those activities can be expressed *formally.*

Within alternative sub-disciplines of computer science the term 'formal' currently has a number of interpretations of varying nuance and strength; as explained in appendix A the sense intended within this thesis is 'represented on a computer in a way that allows operations that are meaningful to the user to be performed computationally'.

How much of the task *can* be formalised?

One possibility is that the whole of some task—at least within a boundary that includes enough to be useful—can be formalised, and can therefore be delegated fully (i.e., 'automated'). The user is left with the job of supplying the raw materials and extracting the end results. Winograd and Flores (1986) point out that man-made *systematic domains,* such as the stock-market world of shares, options and futures, provide entirely appropriate domains for such formalisation:

> 'Computers are wonderful devices for the rule-governed manipulation of formal representations, and there are many areas of human endeavour in which such

> manipulations are crucial. In applying computers appropriately to systematic domains, we develop effective tools.'
>
> (p.174)

But there are many examples of computer systems that over-stretch the legitimate application of formal processing. In the past few years, for example, some word processors have been sold with 'grammar checking' as a feature. The manufacturers claim that these systems can analyse English grammar, detect phrases that do not appear to conform to accepted 'rules', and even offer suggestions for improvement. But the analysis of natural language is notoriously computationally hard, so it is no surprise that these systems make a lot of mistakes; they do not live up to the claim of successful formalisation of this activity.

And at the other end of the spectrum entirely are tasks that are deemed to be completely unformalisable, such as creative design. In such domains computer systems are relinquished to the role of uninformed assistants, because the generality of the task is deemed to preclude formalisation either *a priori* or in use.

How much does the user *want* to formalise?

At some level, there may not be a choice: if information is to be represented on the computer at all, it must be expressed in a way that can be handled formally to some extent. For example, even apparently free-style recording of sounds is typically represented as synchronous sampling of signal level.

Additionally, if the user wants to gain from the ways in which a computer may help by performing routine work such as high-speed repetitive processing of large volumes of information, the information has to have a precisely defined formal representation. And the user needs to be satisfied that a given formal representation of some information contains enough of the intended meaning so that mechanised processing of it will produce meaningful results.

One source of reluctance to formalise arises when the user will be required to perform a conversion between his or her normal way of thinking about the information and the system's formalism; the effort required for the conversion might weigh heavily against the potential benefits of formalisation.

Some advantages and problems of hidden formalisation

Many computer systems are designed to relieve the user of concerns regarding what should or should not be formalised, by maintaining and updating their formal representations of user and task in an internal, hidden form. One way of characterising such systems (often identified as 'intelligent' or 'adaptive') is that they hide from the user the full extent of the delegation that the system has taken upon itself. This can cause problems. As Höök, Karlgren and Wærn (1995) state:

> 'When introducing intelligent interface techniques into an application, one must be aware of the risks inherent in such solutions. These evolve from the fact that an intelligent interface will *change:* it will automatically and sometimes actively adapt to the perceived needs of the user. Unless carefully designed, these changes may lead to an unpredictable, obscure and uncontrollable interface... Systems that act too independently, e.g. knowledge based systems or intelligent interfaces, have not been acceptable to users (Berry & Broadbent 1986, Meyer 1994).'
>
> (original emphasis)

There has been work on reducing the obscurity of adaptive systems—Höök *et al.,* for example, present their own adaptive help interface using the 'glass box' approach (du Boulay, O'Shea & Monk 1981; Karlgren, Höök, Lantz, Palme & Pargman 1994), which gives users the feeling of being able to look through to the workings of the system to see what it has deduced. Work in a similar direction in the field of Demonstrational Interfaces (e.g., Cypher *et al.* 1993) includes presentation of *anticipation* information in the interface of Eager (Cypher, 1991) to show the user how the system has generalised the ongoing activity.

To simplify the goals of this thesis, however, any system that makes its own calculations regarding the delegation appropriate at each moment—even if the user has the means to observe what has happened—is excluded on the grounds of being undesirably complex. This thesis addresses issues that arise when the user has explicit control over delegation; incorporation of assistance offered by adaptive systems can be addressed in future work.

2.1.2 The computer explicitly supports both user activity and delegation

Norman (1988, p.184) draws attention to '... two different ways of getting a task done. One way is to issue commands to someone else who does the actual work: call this "command

mode" or "third-person" interaction. The other way is to do the operations yourself: call this "direct manipulation mode" or "first-person" interaction. The difference between these two is like the difference between being driven by a chauffeur and driving an automobile yourself.' He goes on to suggest that for computer systems 'Both forms of interaction are needed. Third-person interaction is well suited for situations in which the job is laborious or repetitive, as well as those in which you can trust the system (or other person) to do the job for you properly. ... But if the job is critical, novel, or ill-specified, *or if you do not yet know exactly what is to be done,* then you need direct, first-person interaction.' (emphasis added).

Winograd and Flores (1986) note corresponding distinctions in the literature on management and decision making—such as the identification of 'structured tasks' that could be described to a computer using rules, contrasted with 'unstructured tasks' for which rules could not be formulated. But they cite Keen and Scott-Morton (1978) as identifying the in-between area of 'semi-structured' tasks: 'with some degree of recurrence but not so much that one can fully specify the relevant rules', and as seeing this as the relevant area for computer aid to human decision making.

To some extent this mix arises in any computer system that is designed to support an activity in which the range of potential user goals is infinite, but analysis of the domain has revealed that a useful subset of goals appears to be supportable by provision of some form of constructive mechanism such as a query language, programming language or a kit of parts. This characterisation probably accounts for the majority of all computer systems.

Relationship between delegated and user activities

Figure 2.1 is a schematic view of the zones of delegation between a client and a provider (for example, but not restricted to, a user and a computer) in some partially delegatable domain of activity. The regions have the following meanings:

- **black box**

  Aspects of the activity that the client would be unable or unwilling to perform, and is therefore content to accept as 'black box' services from the provider. This tends to be activity of general application within a domain, but of a level of specialisation in which the client may not even be qualified to participate. Examples would be the chauffeur's driving skill, including choice of gears and evasion of obstacles; in the travel agent case it would include the details of composing queries to the centralised

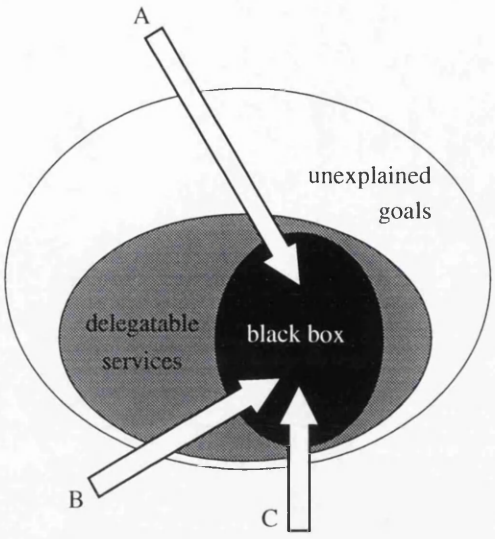


Figure 2.1 A representation of the characteristics of partially delegatable activities.

flight database and interpreting the results. The client just asks for the service, and trusts the provider to get on with it without further prompting.

- **delegatable services**

  These are aspects that, being known in advance to be the sort of services that clients require, the provider is able to perform at the client's request. For example, a chauffeur should be able to undertake the navigation between places chosen by the client; a travel agent should be able to perform the appropriate consultations to inform a client about options for travel between given cities around a particular date. This level is more specific to the goals of the client, so the provider waits to be told which of its services are wanted.

- **unexplained goals**

  This is the level of activity that cannot be delegated, because it is not within the realm of the activity itself but is governed by external influences that mean nothing to the service provider. For example, a client's choice of places and dates for making journeys.

A computer-based example of the above levels can be illustrated with respect to storage of a user's data: the black box incorporates activity such as the allocation of files to sectors on a hard disk; delegatable services include listing, moving and deleting files; unexplained goals would include the user's reason for deleting files X and Y but not Z.

The three arrows marked A, B and C represent different specific tasks that a client may request. The different thicknesses of the regions passed through by tasks coming in from different directions represent the degree of abstraction introduced above the raw 'black box' facilities. So task A represents a task with a high degree of unexplained reasoning but involving a fairly direct call on the black box services: for example, a client who has figured out the one way he could attend an afternoon wedding on the same day as a dinner party 400 miles away may ask 'are there flights from Gatwick to Prestwick on Saturday evenings?'. Task B, by contrast, can be solved almost completely by use of a relatively high-abstraction service: for example, 'please book flights to allow me to spend one day in each of our three overseas offices next week'. Task C is a direct call on the black box services—perhaps 'is there a seat on the next London flight?'.

Indispensability of 'unexplained' user activity

Among the tasks that can be characterised according to the above division of labour is the class of multi-attribute problems dealt with by Bertin (1981). He recommends an approach based on building, evaluating and manipulating a 'graphic' containing the data pertaining to the problem. Although he acknowledges that the display, manipulation and some aspects of the evaluation of the graphic might be delegated to a computer, he firmly distinguishes two classes of information involved:

> 1. *Intrinsic information,* this is, the internal relationships revealed by the image;
> 2. *Extrinsic information,* that is, the nature of the problem and the interplay of the intrinsic information with everything else. And, by definition, everything else is that which *cannot be processed by machine.* Extrinsic relationships cannot, by definition, be automated. They are, however, of fundamental importance in interpretation and decision-making. Thus, the most important stages—choice of questions and data, interpretation and decision-making—can never be automated.
>
> (p.9, original emphasis.)

The importance of making available extrinsic relationships in computer-based presentations is also discussed by Sørgaard (1988). His example is seat reservation in trains; he suggests that it makes a big difference to a potential traveller whether seat position is described simply using terms such as 'window' or 'aisle', 'facing' or 'back', or is shown on a diagrammatic seating plan. The diagram does not even need to be particularly rich to

enable an unpredictably large number of implicit relationships that the passenger may regard as important—such as proximity to the end of the carriage, or to a 'smoking' section, or position relative to a buffet car. In a parallel situation, Casner (1991) admits as an 'important limitation' the inability of his BOZ system's rendering component to make use of domain-specific conventions such as seat layout in an airline reservation system.

The need for delegated and user activity in result-space exploration

However much a user is dependent on the computer for generating the results to be viewed in the course of a result-space exploration, it can be seen that unexplained user activities will usually be crucial as well. The user's concern is to direct the exploration to the level of being able to identify one or several individual results that are comparatively 'good' for the user's current needs. But only the user can judge 'goodness', and only the user can make the necessary opportunistic decisions on how to try to work around undesirable result properties that may be found. Illustrating with the example of the architectural design system, typical issues include the following:

- **results can only be judged in the context of what is available**

  For example, an architect's evaluation of some apartment layout will be affected by features of the other layouts that are discovered to be feasible (in addition to any absolute factors such as budget limitations).

- **the user cannot predict all potential result problems**

  For example, an architect may discover that some suggested layout causes an unexpected infringement of an obscure planning regulation.

- **problems may require indirect, user-chosen modifications**

  A planning regulation may be infringed through a combination of factors; the architect has many parameters to juggle, none of which is directly responsible for the problem. Because of this, it may not even be possible for the architect to know if a proposed modification will resolve the problem, other than by trying it out.

- **problems cannot necessarily be solved in isolation**

  Finding one way to resolve one problem may inadvertently give rise to another.

As will be seen in the survey in chapter 3, some recent developments in the fields of database search and information retrieval can be characterised as belated acknowledgement of this

potentially opportunistic nature of searches—moving away from these fields' traditional implicit assumptions that users only ever engage in searches for which they have a fixed information-seeking goal, known at the start of the search. In the case of databases, the designs of systems such as Rabbit (Williams, 1984; reviewed in section 3.2.3) explicitly recognise that the user might not know even what kind of data are available to be found. Information retrieval systems such as NRT (Sanderson & van Rijsbergen, 1991; reviewed in section 3.2.4) likewise acknowledge that a user may not be able to guess details of documents that would be relevant, but after looking through a few results might then be able to identify those that are more or less interesting.

2.1.3 The user needs to change dynamically the delegation boundary

Consider the changing need for delegation within the flight-booking scenario: At the beginning the travel agent and I submitted specific, narrowly defined queries for particular dates and routes, because we did not know how easy or difficult it was going to be to find a seat. And as the initial results revealed various problems, we had to be there to make judgements on alternative queries to try. But eventually, once we had worked out various alternative compromises that might be acceptable, it became tedious to have to try each combination of compromises in turn; it would have been more satisfactory to be able to delegate a whole batch of compromises and filter all the results using the criteria we had discovered to be relevant. Then again, later still—perhaps when checking on details of a few flights of roughly equal acceptability—we would want to engage in an even more narrow and detailed form of query than at the start.

This is a simple illustration of a need for what may be called *opportunistic delegation*—the ability to define opportunistically some additional level of delegatable services that the user has come to realise would be useful for the task in hand. Lai, Malone and Yu (1988) sympathised with this need for systems that are flexible in this way; they identified a category of *semiformal* systems, as follows:

> 'We define a semiformal system as a computer system that has the following three properties: (1) it represents and automatically processes certain information in formally specified ways; (2) it represents and makes it easy for humans to process the same or other information in ways that are not formally specified; and (3) it allows the boundary between formal processing by computers and informal processing by people to be easily changed.'

In terms of the elements of figure 2.1, the user needs to be able to make opportunistic extensions to the 'delegatable services' region for the system being used, so that a task that is being pursued no longer has such a large 'unexplained goals' segment.

Thus the culmination of the call for human-driven delegation is the identification of a need for facilities by which the user can define, on the fly, additional *opportunistic delegation.* Respecting the view that delegation should only be achieved by means of explicit formalisation, the requirement becomes that of *opportunistic formalisation.* But then, formalisation of a task—according to the definition being used in the thesis—simply involves equipping the computer with rules that it can follow to achieve the task. Stated another way, the user must be given facilities—at some level—by which to 'program' the exploration system.

2.2 The Illumination Zone Model

To assist in analysing and comparing the activities supported in a wide range of computer-based opportunistic explorations I have developed the **illumination zone model.** This was after examining various other models for computer-supported exploration, including (roughly in order of increasing specificity):

- The Information Access Model that underlies the concept of Information Workspaces (e.g., Robertson, Card & Mackinlay, 1993).
- MIE—a Model of Information Exploration (Ahlberg & Truvé, 1994), based on the Seven Stages of Action model (Norman, 1986).
- A model relevant to the *search paradigm* in computer-assisted design (Woodbury, 1991), in which the human information processing model (Card, Moran & Newell, 1983) is supported by a computer in the role of the *external memory* component.
- The functional modules of systems supporting 'Specification by Reformulation' (Yen, Neches, Debellis, Szekely & Aberg, 1991).
- A system architecture for cooperative computer-aided design—specialised to the Flats architectural-design example (Kochhar, 1994).

None of these provides a suitable combination of representing the significant pools and flows of information as well as the stages of progress in the search. In addition it was advantageous to be free to use non-standard names for the model's various components, to avoid pre-conceptions of meaning and thus to let the model describe as broad a range

of domains as possible. As will be seen in chapter 3, the Illumination Zone Model is a way to show many similarities, and some subtle distinctions, between the support offered by systems designed for some highly disparate domains—domains that have not, to my knowledge, been previously identified as a family.

The key to the usefulness of this model is, as the name suggests, the 'illumination zone' itself. As will be explained in the following pages, the illumination zone is an abstract component that can be identified at the heart of a wide range of exploration systems. Basically it is the store for a 'working set' of results that are available for the user to view and compare. The nature of a given system's illumination zone—e.g., the number of results it can hold, and the way its contents are made available to the user—plays a crucial role in determining the system's supportiveness for thorough exploration of a result space.

2.2.1 Activities within exploration

The model is designed to describe explorations that proceed according to cycles of **illumination**, **evaluation** and **consolidation**, as explained below.

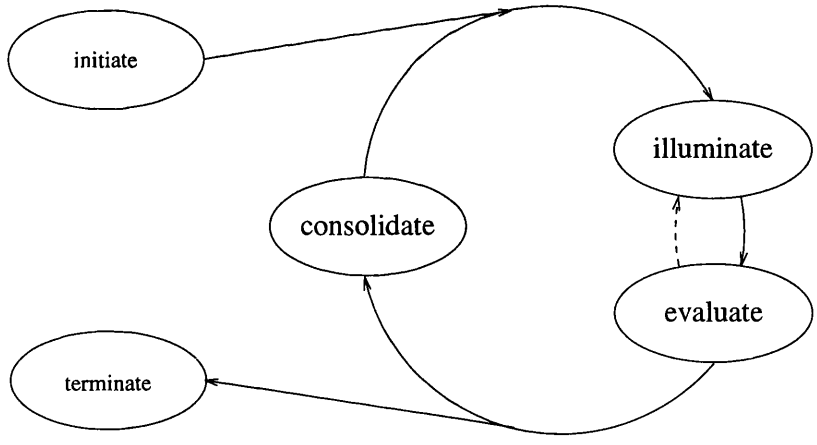


Figure 2.2 The component activities in pursuing an exploration

The explorations of interest in this thesis can be characterised by analogy with a traveller exploring an unknown terrain, who would engage in the following activities:

- **Illuminating** the available ways ahead

  At each step of the way, the traveller performs the actions necessary to obtain some information about what is ahead—i.e., the range of alternative directions available.

- **Evaluating** what has been illuminated

  When information is obtained about the available directions it must be evaluated. During this stage the traveller may choose to revert to requesting more illumination. Eventually the information is used to reach a decision on...

- **Consolidating** progress

  Based on the evaluation of the available directions ahead, and the knowledge of the terrain that has been encountered so far, the traveller consolidates by deciding a) whether the exploration is finished, and if not b) which of the available directions to follow (the choice usually includes the chance to retreat to an earlier location).

2.2.2 The Illumination Zone model

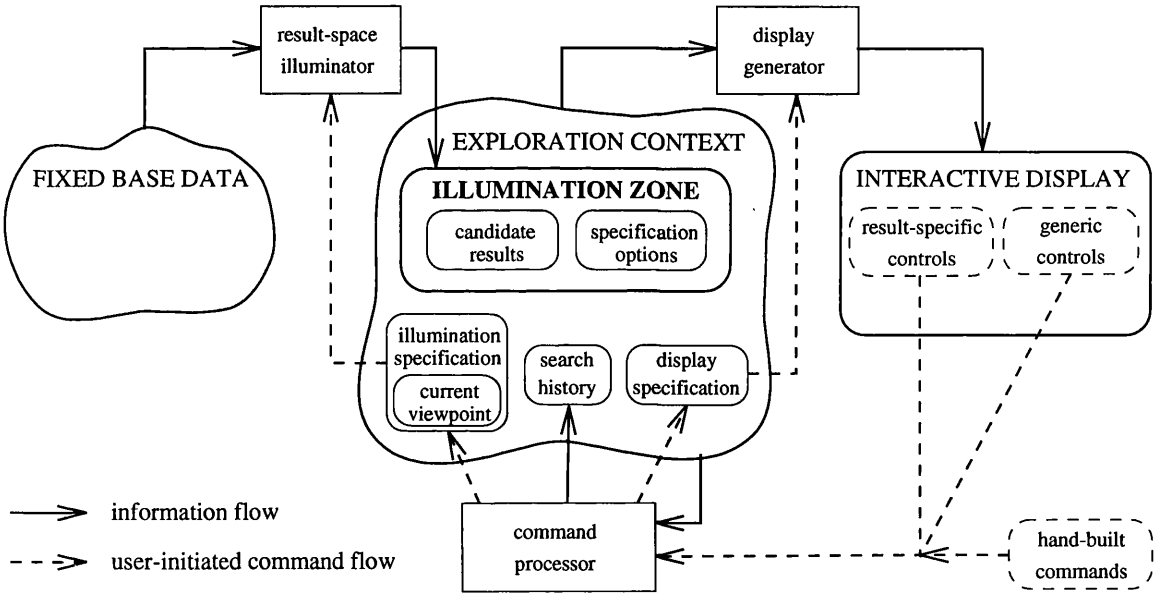


Figure 2.3 Illumination Zone model for user-driven exploration of a result space

Figure 2.3 is a schematic of the components that make up computer support for a user-driven exploration, showing the flow of information under user control. The components' functions can be summarised as follows:

- All the tasks can be characterised as a user requesting the illumination of results that can be obtained from a stable body of **fixed base data**, by supplying an **illumination specification** to some **result-space illuminator**.

The abstract term 'illumination' is intended in a broad sense, encompassing the retrieval or generation of individual **candidate results** and also the illumination of characteristics of the result space as a whole, in terms of **specification options** that could be used for future illuminations. All the illumination information can be regarded as residing in an abstract place that is the **illumination zone** itself.

Note the reference to 'retrieval or generation' in the above description. In some domains the results already exist within the base data—for example, in Information Retrieval or database search—but the model is more general than that: the base data may provide merely some material that assists in generating results. Examples later in this section will illustrate both concepts.

The illumination specification includes a notion of the **current viewpoint**, which comprises whatever information is needed to describe the user's 'location' in the exploration—e.g., the latest version of a design.

- The illumination needs to be converted into a visible form, to which end a **display generator** acts in accordance with a user-controlled **display specification** to produce an **interactive display**.

  Since the illumination doesn't necessarily include the details of how it was requested, the display generator needs to have access not just to the illumination zone but to the entire **exploration context**, which also includes the illumination and display specifications and also whatever form of **search history** the system may record.

- The user determines the retrieval and display specifications, and hence the operation of their respective engines, by interaction with a **command processor**.

  The instructions to the command processor may be **hand-built**, such as requests typed on a 'command line', or launched from **generic controls** or **result-specific controls** embodied in the presentation in the system's interactive display. Like the display generator, the command processor may interpret the user's commands by reference to any information held in the exploration context.

The exploration sub-activities described above are considered to be carried out by the components of the model as follows:

Illumination

An 'illumination' request is handled by the result-space illuminator. It may produce candidate results or specification options (or both), depending on the domain and what the user requests.

Illumination by direct retrieval of candidate results is the most straightforward kind. The user makes some request, and is presented with one or more results corresponding to the request—for example, a set of alternative flights or floor plans. The generation of the results may take quite some time—several seconds, or even minutes—but there are benefits to the user in having many results in the illumination zone at once, rather than only being able to encounter them one at a time.

For some domains, the user has to be able to make the first illumination request without any knowledge of the results that are going to be available. The flight search is an example: the user has to declare at least the broad region in which this exploration is going to occur, by constructing a query that includes, say, departure and arrival airports and a date. The corresponding first set of candidate results will then guide the user in deciding on subsequent requests to make.

The nature of any available specification-option information differs from system to system, and includes the following types:

- a listing of permissible parameters that are available for illumination—e.g., all the different classes of ticket that are represented in a flight database;
- a context-sensitive listing of alternative parameters available in the current exploration state—e.g., production rules that could be applied to some existing state of a floor plan;
- measurement of some aspect(s) of a result implied by the current viewpoint, or of the results that would be retrieved if the user were to choose each of various alternative retrievals—e.g., the floor area of the current working design, or a list showing some file directories and, against each, the total size of the files within it.

Evaluation

The user has to view the illumination information in order to evaluate it, so evaluation is the chief responsibility of the display generator.

As will be shown in the survey of existing exploration systems, a particularly important aspect of evaluation is the *comparison* of alternative results and/or directions of search. In most cases comparison is greatly facilitated by being able to view all the alternatives simultaneously, side by side. Therefore, for making comparisons it is of great importance whether the system allows the information of interest to be accommodated within the illumination zone at the same time.

Consolidation

Consolidation is the act of moving on in the exploration. It is manifested in the illumination-zone model as a change in the current viewpoint.

There are many forms of consolidation. An example that one might consider to constitute 'advancing' in the exploration would be when the user decides to adopt some part of the illuminated result space in the new viewpoint, such as by picking one of the candidate architectural designs to be the seed for the next illumination. A 'retreat' in the exploration may be signified by the user relaxing a constraint that was previously a part of the viewpoint description, presumably having decided that the search space in that direction is not worthy of more detailed investigation.

In systems that support exploration by successive refinements that are all accumulated in a search history, the user may request explicitly to revert to an earlier state. But with respect to a system that has no built-in model of search progress (e.g., the flight database) it is only in the mind of the user that there can be a distinction between advancing and retreating—as far as the system is concerned, all that happens is that the user supplies one illumination specification after another, guided by reasoning that is never expressed.

Sample applications

To illustrate how these components can be identified in existing examples of exploration, and to give an impression of the breadth of exploratory interaction the model is capable of representing, here are three example activities:

1. **Accessing a query service for flight seat availability** as illustrated in the thesis introduction. The user's goal is to find information revealing a suitable available flight.

   Here the base data represents the repository of all flight and reservation information, and the illumination specification is simply the user's current flight query. Many flight reservation systems offer little opportunity for the user to affect the display of retrieved flight details—perhaps no more than scrolling through results one screen-full at a time. The result-specific controls may be rudimentary, such as showing alongside each flight an index number that the user can type to call up detailed information on that flight.

   Multiple pages of results can be thought of as being brought into the illumination zone simultaneously in response to a user's query, and revealed a page at a time in response to commands that affect the display specification but not the illumination

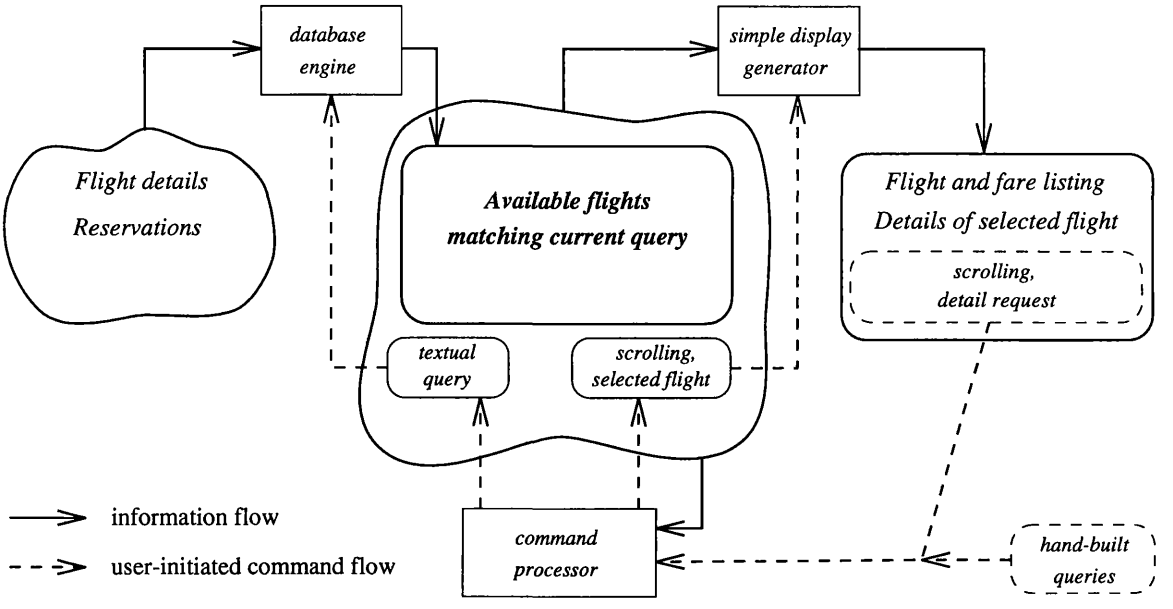


Figure 2.4 Illumination Zone Model for flight booking.

specification—because as far as the user is concerned the *illumination* ('flights from X to Y on date D') remains constant, and the illuminated results remain accessible *until the next query is specified.* But the results from a new query will take over the illumination zone, replacing the previous ones and making it harder for the user to view them again.

Note that this might not be how the system is implemented: perhaps the query is only processed in stages, waiting after each page of results to see whether the user wants more; indeed it may involve multiple stages of result fetching and caching. There may be no single storage location that corresponds to an illumination zone. But that is not my concern. The crucial observation is that there is a qualitative distinction between the level of commitment involved in (a) browsing through the results within the scope of the current illumination, and (b) moving on to illuminate another part of the result space.

2. **Changing parameters that control the formatting of a given document** with the aim of finding a satisfactory layout (e.g., sensible pagination, placement of figures)[1].

[1]This is probably not a *common* scenario in document processing but, as I found in my experiments (see chapter 7), it certainly does happen. More often, an author is given a style guideline and has to massage the document to suit the style; that is also an example of exploration, but a different one.

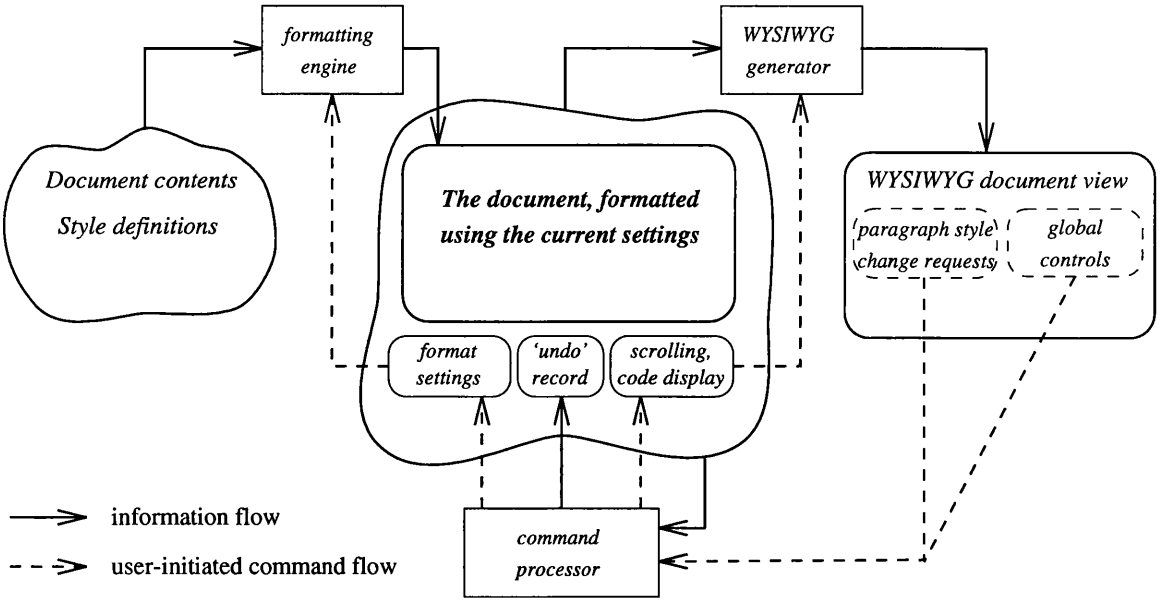


Figure 2.5 Illumination Zone Model for document formatting.

For this exploration the base data contains the definition of the document: its text, figures, headings, paragraph boundaries, etc. The illumination specification consists of the formatting details such as the margin settings, font selections, line separations and so forth. Obtaining illumination corresponds to formatting the document; the formatting engine can be considered to deliver into the illumination zone the full descriptive form needed for a display generator to produce a WYSIWYG online presentation. The display specification simply determines details such as scrolling, if the document it is too big to be seen in a single display, and whether or not the display should include non-WYSIWYG aspects of the document such as control codes.

An interactive WYSIWYG word processor display constitutes a powerful example of result-specific controls; the user can request a large range of commands by acting on the displayed text as well as on the adjustment widgets in a 'ruler'.

But, again, the user is faced with a clear distinction between the impact of commands that affect just the display specification, such as scrolling from one page to another, and those that cause the document to be reformatted, such as an adjustment to the margin settings. The latter causes the document in the illumination zone to be replaced, although typically there is an 'undo' facility that enables at least a single step of reformatting to be undone if the user dislikes its effect.

3. **Exploring architectural design alternatives** such as the sizes, shapes and arrangement of rooms in an apartment of given overall shape. The goal is bound to involve satisfaction of planning regulations, as well as finding a layout that will be convenient and pleasant as a living-space.

   The base data for a computer-supported cooperative design system (e.g., the Flats system, reviewed in section 3.2.4) consists of rules that embody the basic knowledge about what is possible and permissible in designing a building—such as the minimum useful sizes for various kinds of room (including connecting passages), standard widths of doorways, and higher-level rules such as regulations relating to areas without any external wall. Each illumination generates a set of alternative plans that are different refinements of a single partially developed plan. The illumination specification includes the current state of the plan as the viewpoint, alongside the user-suggested directions for progress. In the Flats system there are various display facilities for the user to browse the generated alternatives, and to accept one of the designs as the next stage of development of the plan.

2.3 The effort–accuracy tradeoff in opportunistic search

In their book *The Adaptive Decision Maker,* Payne, Bettman and Johnson (1993) propose a framework for discussing a tradeoff that is invoked, consciously or unconsciously, by decision makers faced with more alternatives than they can afford to examine in detail. This section describes various parts of the framework that are relevant to the aims of this thesis.

2.3.1 Overview of the effort–accuracy framework

Payne *et al.* identify the domain of their study as 'decision making', stating one of the characteristics distinguishing this activity from other forms of problem-solving as being that 'decision problems are generally ill-defined about exactly how the final goal state is to be characterized'. Although this thesis is *not* intended to be restricted to activities whose goal is a 'decision' on a single result, the unpredictable nature of the journey and arrival point are characteristics shared with their decision-making model.

The decision-maker's behaviour is regarded as proceeding according to *decision strategies.* Formally, a decision strategy is defined as 'a sequence of mental and effector (actions on the environment) operations used to transform an initial state of knowledge into a final goal state of knowledge where the decision maker views the particular decision problem

as solved.' Different strategies achieve this goal by calling on different atomic operations, and different ways of combining them. For example, two of the strategies examined in the book are the 'weighted additive rule', in which the decision proceeds by calculation and comparison of detailed quantitative scores of every result, and the 'satisficing heuristic' which involves qualitative evaluation of successive results against those seen previously, until one result seems unlikely to be bettered. The first requires a great deal of calculation and comparison, leading to a thorough appreciation of the relative merits of all results (high effort rewarded with high accuracy), whereas the second requires much less effort but exhibits the flight-booking scenario problem of potentially missing good results (low effort, with potentially low accuracy).

The main thesis of the book is that when faced with a decision problem, a decision maker who is familiar with a range of such strategies will choose a strategy that, for the particular task and result set being considered, offers sufficient accuracy at an affordable level of effort. This choice is opportunistic in that it may need to take into account the ongoing discovery of the available results: for example, a strategy must be rejected if it is found to be unable to produce clear distinctions between those results. Payne *et al.* offer evidence suggesting that strategy choice is a rational, justifiable response to what the decision maker perceives as the likely effort and likely accuracy obtainable from available strategies.

Although Payne *et al.* present many precisely specified strategies such as the two mentioned above—to illustrate some of the different approaches available—they acknowledge that in any real task it would be unusual for any such strategy to be followed to the letter. It is more usual for a decision maker to engage in a sequence of operations that can be seen as an assembly of elements from many known strategies, and for this sequence to have 'a less than coherent overall structure'. Overall, however, the operations chosen reflect the decision maker's position regarding the effort–accuracy tradeoff.

It is noted that there are many contextual influences on a decision maker that can guide the attitude towards appropriate levels of effort and accuracy, and hence choice of decision strategy. To be punctiliously 'accurate' in some kinds of decision may involve emotionally stressful tradeoffs that a decision maker would prefer to avoid; by contrast, in other situations the decision maker must be careful to choose a strategy that can later be justified to the people it affects. But Payne *et al.* note their expectation that 'constraints on accuracy are much less common influences on strategy selection than constraints on effort'—in other words, it's less typical for a decision maker to have to seek the best possible solution than to be obliged to make a decision within constrained time.

In the next section I review some of the characteristics of strategies that are described, showing ways in which effort and accuracy are influenced by choice of strategy. Then I

describe their view on what it takes to influence a decision maker to seek greater accuracy, and finally the component of this persuasion that derives purely from the nature of the provided decision-support artifacts.

2.3.2 Decision-strategy characteristics affecting effort and accuracy

Payne, Bettman and Johnson identify the following general properties of choice strategies:

- **Compensatory vs. noncompensatory**

  Noncompensatory strategies involve discarding a result as soon as it is found to score badly on any attribute, while compensatory strategies allow good scores on other attributes to be weighed in the result's favour. My flight-booking scenario included the noncompensatory move of discarding all October flights early in the search. Although noncompensatory processing will tend to involve a lot less effort than painstaking tradeoff of all attributes, it can lead to the elimination of potentially good results early in the decision process.

  Payne *et al.* note that switching to a noncompensatory strategy is a common response to time pressure in decision making, as is *filtration:* simply cutting out some parts of the potential range of hypotheses or actions considered.

- **Consistent vs. selective processing**

  Closely related to the previous property, this is a measure of whether the amount of information examined for each result or attribute is the same, or whether it varies from one to the next. Noncompensatory strategies, for example, are usually selective in that only the results retained at each stage are subjected to further levels of examination.

- **Alternative-based vs. attribute-based processing**

  Alternative-based strategies involve evaluating each result by examining many of its attributes together before advancing to examine the next result, whereas attribute-based processing involves examining how several results compare on one given attribute before looking at the next attribute. Russo and Dosher (1983) are quoted as suggesting that attribute-based processing is cognitively easier.

- **Formation of evaluations**

  Some strategies involve assembling an overall evaluation for each result, taking all its attributes into consideration, whereas in others the results are judged (e.g., compared) on the basis of just a subset of their attributes. Clearly attribute-based strategies

would tend to fall into the latter category, although in theory the result evaluations could be accumulated on the basis of one attribute at a time.

- **Quantitative vs. qualitative processing**

  The reasoning operations that are used within a given strategy may be predominantly quantitative, such as the multiplications and additions used in the weighted additive rule, or qualitative, such as the judgement relative to 'aspiration levels' that is used in the satisficing heuristic.

2.3.3 Opportunities for improving decision performance

Payne, Bettman and Johnson point out that the choice of decision strategy is not just determined by the theoretical effort and accuracy involved in the various alternative approaches, but is influenced by the more person- and context-sensitive issues of *availability, accessibility, processability,* and *perceived benefits* of each candidate strategy. The strategies that are *available* might be just those of which the person has prior knowledge, obtained through experience or training; *accessibility* (ease of calling a strategy to mind) is also influenced by prior experience; *processability* can be affected by considerations such as the decision information only being available in a format that would be cognitively hard to manipulate in the way required for a particular strategy. The *perceived benefits* of a strategy include the perception—which may be distorted, as outlined below—of the strategy's effort and accuracy characteristics.

Overall, one cannot simply expect that awareness of a strategy suited to the task in hand will be sufficient incentive for it to be employed. Payne *et al.* propose the following as necessary conditions for take-up of a strategy, and suggest typical reasons for these conditions not being satisfied:

- **A belief that the present strategy gives less than the desired accuracy**

  There are many reasons for a decision maker to fail to appreciate that a given strategy is less accurate than is wanted. The first is overconfidence:

  > 'Indeed, one of the most well-established errors in judgment is the overconfidence bias. ... Therefore we hypothesize that individuals may overestimate the accuracy they will achieve with particular decision heuristics. If true, this suggests that people may select heuristics that will save effort but not produce the expected level of accuracy.'
  > (Payne, Bettman and Johnson, p.209)

Another reason is that it is simply very difficult to know how well one is doing—or how well one might do with an alternative strategy—compared with an inaccessible, theoretical 'normative' model of task performance. On the other hand, it is relatively easy to get a feel for the level of effort being expended. As a result, effort may be overweighted relative to accuracy in strategy selection. Paese and Sniezek (1991), for example, report that increased effort can lead to increased confidence in judgement without accompanying increases in accuracy.

- **Being able to see that there is an alternative better strategy**

  Tversky and Kahneman (1981), in their studies of how people 'frame' particular decisions, suggest that subjects 'are normally unaware of alternative frames and of their potential effects on the relative attractiveness of options.' In later work (Tversky and Kahneman, 1990) they also note that people attempt to save effort by processing a problem as originally presented, rather than looking for alternative views or approaches that might be more helpful.

- **A belief that one is capable of executing the new strategy**

  An individual may feel unable to execute a strategy for reasons such as the difficulty of performing the required calculations, or of keeping track of goal plans in working memory, or for external reasons such as time pressure or heavy levels of distraction.

In summary, a decision maker may either fail to realise that a change in strategy is needed, or may see the problem but feel unable to rectify it. For a strategy to stand a good chance of being picked up it must be clearly available and executable. But the way a task is presented can have a forcing role in guiding strategy use: Kleinmuntz and Schkade (1993) observe that even static displays such as price lists and menus can influence strongly the selection of decision strategies, by virtue of characteristics such as the form, organisation, and sequence of presented information. A computer-based system that only offers particular facilities may leave its users little or no choice among strategies, so the provision of facilities—and the users' reaction to them—must be evaluated with care.

This thesis addresses the development of a new kind of high-level strategy for tackling opportunistic searches. The work of Payne *et al.* suggests that when a strategy is available and executable, and evidently offers an advantageous effort–accuracy tradeoff compared to the others available, users are likely to choose the advantageous strategy. Thus the development of a novel strategy that clearly provides a relatively low-effort means to perform a wide range of evaluations—even if the total effort over the course of the exploration will not obviously be lower—does stand a good chance of being adopted by users, and therefore improving the accuracy of the exploration.

Chapter 3

Computer support for opportunistic exploration

3.1 Introduction

This chapter reviews a selection of computer systems that are designed for domains that in some way involve result-space exploration. The aim is to reveal and analyse the features that are liable to engender an *under-informed outcome* in an opportunistic exploration of available results.

Section 3.2 contains a detailed analysis of fourteen different kinds of computer-supported exploration, grouped into five categories. The analysis addresses the division of exploration effort between user and computer: what the computer can and cannot do, and what the user would therefore have to do to pursue a *thorough exploration* in the domain being served. The point being illustrated is that even systems with ingenious domain-specific features for harnessing the computer's processing speed and flexibility in pursuit of good results can still provide poor support for the legitimate user goal of being thorough. To explore the hypothesis that the key issue in thorough exploration is that of *illumination*, the analysis is couched in terms of the Illumination Zone Model and its accompanying three-stage exploration cycle as introduced in section 2.2.

Section 3.3 then considers exploration systems from an alternative direction, addressing the relationship between the specificity of domain a system serves and the facilities it provides for the user to formalise (and hence delegate) exploration sub-tasks. In section 2.1 it is suggested that exploration requires opportunistic delegation, and that this implies that the system must be semiformal. This review examines whether there are any exploration

systems that combine semiformal delegation behaviour with the kind of task-generality necessary to support exploration.

Finally, section 3.4 contains a survey fleshing out an issue briefly touched on in section 2.1—that opportunistic delegation tends to imply a form of programming. The aim of the survey is to show that programming can take a number of alternative forms, and certainly need not be as arduous and abstract as the learning and use of a traditional programming language.

3.2 Exploration strategies

As described above, the tools reviewed in this section are organised into categories. These categories each correspond to a different *exploration strategy*. Like the decision strategies considered by Payne, Bettman and Johnson (1993) (as described in section 2.3), the exploration strategies constitute alternative approaches to a result space—differing in terms of which results are examined, how many are examined (what proportion of the 'space'), what sort of evaluations and trade-off decisions are made and when, and so on. However, the analysis presented here does not parallel Payne *et al.*'s investigation into how a decision maker selects one of many strategies that could be pursued on a particular decision. Explorations made using a computer-based tool are overwhelmingly guided by the facilities that the tool provides: a user is only minimally free to choose between alternative approaches to the result space. Therefore the 'strategies' are simply the principal modes of use supported by various kinds of computer system.

Because of the generality of result-space exploration, this survey cannot hope to cover all examples that may be relevant to the thesis. However, the survey does address systems drawn from many diverse topics within the field of human–computer interaction; even to rationalise this scope of existing research is an unusual and worthwhile goal. In the future it would be valuable to incorporate other specialised fields of computer application—such as dynamic system simulation, theorem proving, and rule-based analysis—to understand whether they offer radically more powerful approaches or are essentially limited by similar constraints to the systems examined below.

Each strategy is described under the following headings:

- **Examples**

  Systems that embody the strategy, including a characterisation of one example in the form of the illumination zone model.

- **Exploration progress**

  Pursuit of the strategy in terms of the illumination, evaluation and consolidation activities. For each activity I characterise the typical support provided by the computer system, and what a user therefore has to do to achieve a thorough exploration.

- **Opportunity to delegate**

  A summary of the overall balance of exploration effort and initiative between user and computer, and the appropriateness of this balance with respect to the kinds of domain in which the strategy appears.

3.2.1 Trial and error

A minimal form of exploration 'strategy' is the use of trial and error—i.e., the user keeps trying different specifications until a satisfactory result has been found.

Examples

- **A simple database interface**

  An example of the trial-and-error strategy is the flight-search scenario that was used for illustration in section 1.1.1. That scenario assumed the existence of a simple form of database query tool, used by the travel agent.

- **A document formatter**

  As was noted in section 2.2.2, the refinement of a document's formatting in an interactive word processor can be seen as an example of result-space exploration. But a typical general-purpose word processor has no formalised facilities for pursuing an exploration as such, so a user wishing to experiment with formatting a document in different ways is obliged to follow a trial-and-error approach.

The analyses of these systems into illumination-zone components are shown in figures 2.4 (p.22) and 2.5 (p.23) respectively.

Exploration progress

- **Illumination**

  The region to be illuminated must be specified explicitly by the user. For a simple database interface the specification is typically in the form of a textual query, with a

specialised syntax for which the user is given no on-line assistance. A word processor's formatting commands, by contrast, may be made available for interactive perusal and selection through menus, dialog boxes (e.g., 'style sheets'), direct-manipulation ruler lines and tool bars; a new illumination request is implied by each alteration of a format parameter, or of a set of related parameters as in a style.

As shown in figure 2.4, the illumination zone only holds the results of the *current* query. The number of results from a database depends on the correspondence between the query and the data, and is therefore largely unpredictable—although a user with experience in the exploration domain can often make an educated guess at the likely level of response for a given query. The user needs to find a balance between having so few results that each iteration provides a negligible amount of insight into the exploration, and having too many to be able to evaluate them (as discussed below). Thinking up and expressing a query in a complex database language can represent quite a large effort, for each query.

Because some document format changes can be requested simply by the push of a button, each illumination request can require considerably less effort than for the database. However, since each illumination request replaces the previous one there is only ever a single version of the document on view. There is no theoretical barrier to allowing multiple results, but there are at least the following strong pragmatic reasons: firstly, because the document is displayed in its entirety it may be prohibitively expensive in system resources to show multiple versions; secondly, since the document display is an interactive direct-manipulation environment, having multiple alternatives would create the tough challenge of providing a robust and understandable means of handling consistency between them.

- **Evaluation**

  Result display is typically restricted to a single format, 'controllable' only in terms of scrolling the display if it is too large to be presented as a whole.

  Because the illumination zone contains just one set of results, and the systems tend to support little if any search history (see below), there is no way to obtain a display that combines results corresponding to different specifications—unless one is a subset of the other, of course. Apart from the opportunity, discussed below, of using 'undo' facilities to revisit earlier illumination results, comparison between results must be carried out entirely by the user, by use of memory or even external notes if appropriate.

- **Consolidation**

  A trial-and-error database system does not provide the means for a user to feed aspects of the results from one request into the specification of another. The two ends

of the process are deemed separate, with the user performing the opportunistic task of deciding what alternative requests to try if the results are not satisfactory.

Systems that lack an explicit model of progressive exploration are unlikely to maintain a search history. But many word processors maintain the limited form of history needed to support an 'undo/redo' facility, and therefore allow relatively easy comparison of the document state before and after a given change. However, even systems with multiple-level 'undo' support typically maintain only a single thread of changes; if a user undoes a succession of changes and then starts on a different direction from before, the 'redo' information is discarded. Accumulating information on the state of exploration progress, including the best result properties found so far, is entirely the user's responsibility.

Slow progress through the result space, and poor support for keeping track of the results that have been seen, make it especially challenging for the user to decide when to stop searching—i.e., to make an informed decision on the likelihood of finding better results in a continued search.

Opportunity to delegate

One reason for a designer to provide a system that supports only trial-and-error explorations is that the supported activity is too general to be able to predict the kinds of exploration a user may want to undertake. The systems therefore provide facilities at a low level of abstraction, making them flexible but not labour-saving.

The user has to specify each request by hand and to evaluate every result in detail, and is given no help in reformulating requests using previous results. There is no 'exploration context' to support a developing state of progress, or even to support comparison of successive results.

Both the database and the word processor make thorough exploration costly in terms of user actions and mental operations. The designers' expectation appears to be that somehow users will be satisfied with less than a thorough exploration. User experience helps to maximise the effectiveness of a cursory search—for example, the travel agent who has a good 'feel' for the airline schedules and the way they tend to get booked up. But the over-confidence tendency discussed in section 2.3.3 suggests that the actual accuracy obtained will often be lower than users are aiming for, or believe they have achieved.

Both systems examined also require users to know what requests are available, and to have some idea of the effects that such requests should be expected to achieve. For the word

processor the presence of menus and buttons that 'afford' particular actions is a help; for the database an insufficiently trained user will be in trouble, as pointed out by Fischer and Nieper-Lemke (1989):

> 'People who attempt to use a complex information store on a computer encounter a number of problems: They do not know what information exists or how to find information, they get no support in articulating a question, and they are unable to phrase their question in terms that the system understands.'

This particular source of difficulty in pursuing an exploration is addressed by the 'direction illumination' strategy (section 3.2.3), which includes Fischer and Nieper-Lemke's own approach to the problem.

3.2.2 Computer critiquing or improving

Some systems can assist a search by evaluating and perhaps 'polishing' results proposed by the user. Design-assistance systems with these capabilities are referred to by Kochhar, Marks and Friedell (1991) as *critic-based* and *improver-based* respectively.

Fischer, Lemke, Mastaglio and Morch (1991) provide a detailed discussion of critiquing systems for what they call *cooperative problem solving.* Their opening claim is that 'cooperative problem-solving systems help users design solutions themselves as opposed to having solutions designed for them.' Any design articulated on such a system is immediately analysed, in a process that Fischer *et al.* refer to as the artifact 'talking back' to the designer. This usually consists of generating a list of design shortcomings that the designer may not have the experience or skill to notice.

Critiquing systems reviewed by Fischer *et al.* include the following:

- Janus, for assisting in the design of residential kitchens;
- Framer, that can help with the development of tools with window-based user interfaces by advising on conflicts with pre-coded style guidelines;
- LISP-Critic, which suggests improvements to LISP code to make its execution more efficient, or to make the code easier to read and maintain.

Improvers mentioned by Kochhar *et al.* include:

- a 'beautifier' for network diagrams (Pavlidis and Van Wyck, 1985), in which boxes and their connections are aligned and squared up, and have minor overlaps and gaps removed;
- the Designer system (Weitzman, 1986) for creating graphical interfaces to instructional systems, which includes facilities for recognising and tidying subobject relations based on similarity, proximity and repetition.

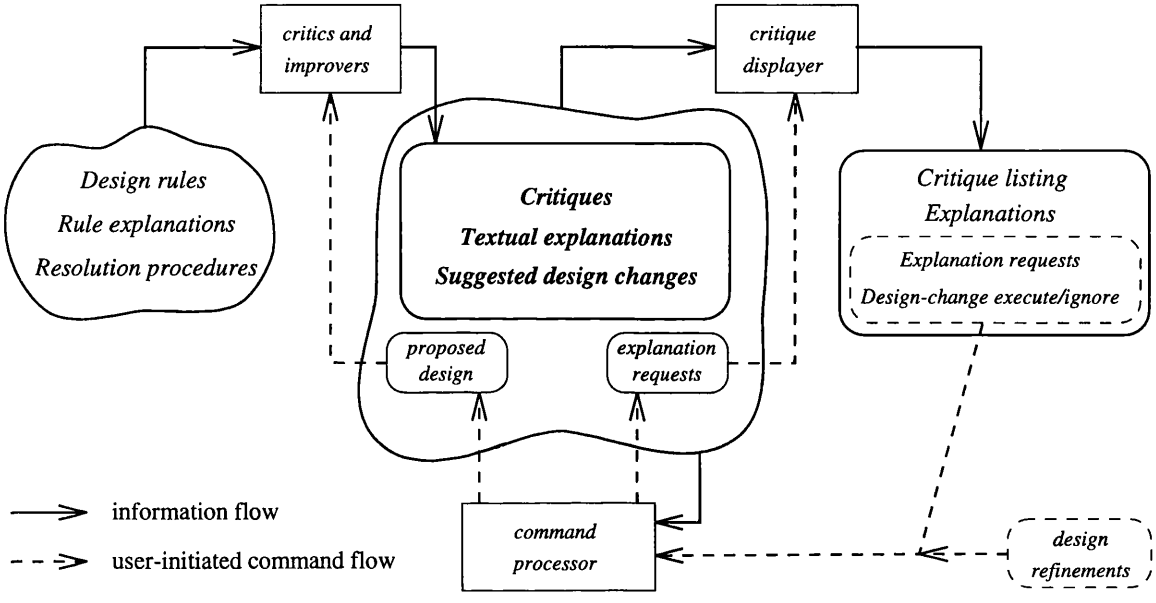


Figure 3.1 Illumination Zone Model for computer-based critics.

Figure 3.1 shows the components of a critiquing system, which supports the following pattern of activity:

1. The user articulates a proposed design. This design can be thought of as the current viewpoint of the exploration.
2. The illumination process consists of evaluating the user's design against a knowledge base of design rules. This is performed by one or more computer-based *critics.*

   The critics are simply an abstraction of processing elements that each analyse the proposed design according to a particular pre-built design rule. When one of its rules is contravened, a critic generates a *critique* by retrieving the appropriate explanation from an *argumentative hypertext* component that is linked to the rule base.

3. All the critiques are gathered and presented to the user; since some of the rules may be classified as mandatory while others are merely optional recommendations, the system may ensure that the mandatory ones appear at the top of the display.

   Some systems' critics can be equipped with resolution procedures for solving simple problems—such as the lack of a menu bar in a user interface design. The display for such a critique can provide an easy way for the user to apply the suggested change. Fischer *et al.* describe the presentation of critic messages in Framer as follows:

   > 'Each message produced by Framer is accompanied by up to three buttons: *Explain*, *Reject*, and *Execute*. The *Explain* button displays an explanation of the reasons the designer should consider this critic suggestion; it also describes ways to achieve the desired effect. Optional suggestions have a *Reject* or *Unreject* button depending on the state of the suggestion. The *Execute* button accesses the advisory capability of Framer, which is available for issues that have a reasonable default solution.'

4. Having evaluated and understood the system's suggestions, and possibly incorporated them into the ongoing design, the user iterates the process by specifying a further level of provisional design for critiquing by the system.

Instead of providing a detailed critique, an *improver* is typically empowered to go ahead with changes to the design suggested by the rules in its rule base. The result of the illumination is therefore an updated version of the supplied design, which the user can then update and iterate if wanted.

Exploration progress

- **Illumination**

  A critiquing system deals with only one result—a critique of the proposed design—at a time. This 'illumination' is generated entirely by the computer system. Some critiquing systems incorporate *passive critics,* that have to be invoked explicitly by the user when the design has reached a stage on which some feedback is desired. Other systems have *active critics* that 'watch over the user's shoulder' and generate critique messages as soon as rule violations are detected. Typically the user does not have a choice over which rules are checked.

  Each critic presents an explanation of the rule that is deemed to have been contravened. As well as helping the user understand what is wrong with the current

proposed design, this explanation can clearly guide the user's future design attempts and can thus be seen as illuminating aspects of the result space that were presumably unknown to the user.

- **Evaluation**

  A critiquing system supports evaluation of the design against the rules in the rule base. It does not make it easy for a user to try alternative designs and evaluate them against each other.

- **Consolidation**

  Where a rule contravention has an obvious solution, the critiquing system can offer the user a push-button way to resolve the problem. If invoked, this updates the proposed design and may change the set of rules it now contravenes. The system can update the critique list accordingly so the user does not try to resolve problems that no longer exist.

  Contravention of rules for which there is no unique obvious resolution requires further design activity by the user, and a further round of critiquing in case new problems have been introduced.

Result-space coverage

Critiquing is essentially just a way of helping a trial-and-error search by providing good reasons for or (more usually) against the results that the user proposes. It helps the user to become aware of some pitfalls in the result space, but only those that are stumbled into; if the design contains no problems, according to the system's definitions, it offers no advice such as alternative designs. Thus if a user already has a number of alternative designs in mind a critiquing system will help point out which of them have particular failings, but there is no explicit support for generating or comparing such alternatives.

Opportunity to delegate

A critiquing system only takes on the parts of design analysis that its own designers felt could be unambiguously codified in rules. As such it can incorporate a large body of design rules and guidelines, that would otherwise require a great deal of user effort to check and enforce. All creative design ideas, however, are considered the inviolable domain of the user. An exploration of a large number of alternatives would require a large amount of user effort.

3.2.3 Refinement from illumination of available progress directions

Introduction to refinement-based strategies

Many different kinds of system can be seen as embodying a model of search as *result specification refinement,* in which the user starts with a vague retrieval specification and progressively adds constraints until the specification precisely describes what is wanted. Both this section and section 3.2.4 illustrate different forms of search that support this kind of strategy.

But note that refinement implies more than simply letting the user build up a progressively clearer idea of what is wanted. When a system provides explicit support for refinement it does so by placing great emphasis on constant movement *forward* by expressing additional constraints. So whereas someone in the course of searching for a flight may haphazardly wish to submit a request involving a completely different date or route, just to confirm an expectation or venture a wild guess, the user of a refinement-based system is discouraged from such behaviour. This effect, and its influence on searching, will be pointed out in each of the systems reviewed below.

Examples

The first refinement-based strategies to be examined are those whose support simply involves helping the user to see the alternative ways in which the current specification *could* be refined.

- **Retrieval by reformulation**

  Retrieval by reformulation was first explicitly supported in Rabbit (Williams, 1984), and developed further in Helgon (Fischer & Nieper-Lemke, 1989). The concept was inspired by the *descriptions* model of human remembering (Norman & Bobrow, 1979), which accounts for the way people appear to recall items from very long-term memory (such as the name of an erstwhile classmate) by considering the first related items that come to mind, and using specific properties of those items in successive cycles of refinement to point the way to the target.

  In arguing for the applicability of the reformulation approach to computer-based retrieval, Fischer and Nieper-Lemke pointed out that 'The interaction paradigms for dealing with complex information stores (as well as databases) have often been based on the unfounded assumption that people using these systems approach them with a

precisely described task. But in most problem-solving and information retrieval tasks, the articulation of a precise task is the most difficult problem.' In other words, users might not even know what they can ask for until they have seen some examples of what is available. The job of a retrieval-by-reformulation system, therefore, is to help the user refine the notion of what is wanted by reference to properties observed in example results.

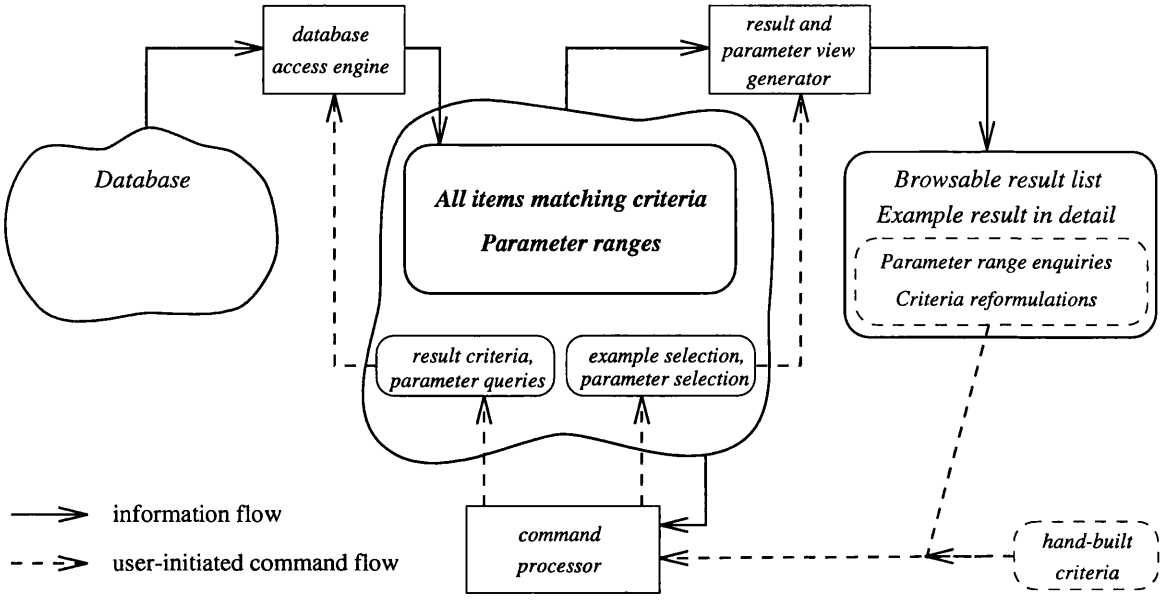


Figure 3.2 Illumination Zone Model for retrieval by reformulation.

Figure 3.2 is an illumination-zone representation of a retrieval-by-reformulation system. Using Helgon as an example, a search proceeds as follows:

1. The user supplies initial result criteria that at least specify the kind of item that is wanted. For example, 'a flight'.
2. The database access engine retrieves all the items that match the current criteria.

   If the database is large and the query only minimally constrained the number of matching items may be far too large for them all to be displayed; the system may impose a practical limit on the number of items actually enumerated in the illumination zone.

   But until the final stages of the search the user is not expected to examine all the matching items in detail, but just to examine some for examples of further ways to constrain the search...

3. A display is constructed to show a list of matching items, with a single example item (e.g., the first in the list) expanded in detail. This detailed presentation will reveal to the user some of the attributes that are used to describe items in the database, and the values for the attributes possessed by the example item.

   The user can interact with the display to select from the list alternative example items for detailed presentation.

4. While there remain too many items in the list for the user to find the 'ideal' item for the current need, the user needs to continue the refinement by expressing additional constraints. The system provides various commands for updating the result criteria by interacting with items and attributes shown on the display.

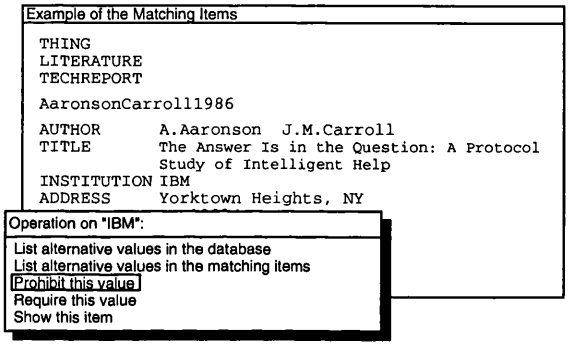


Figure 3.3 Helgon, showing reformulation options. (From Fischer & Nieper-Lemke, 1989)

For example, in figure 3.3 a Helgon user searching for technical reports in a publication database has been shown an example (the first report in the database). Among the attributes displayed for this example is the INSTITUTION, which in this case has the value 'IBM'. Having requested the pop-up menu on this attribute, the user has been given a range of available operations. In this case the user is not interested in reports from IBM, and by refining the query to prohibit this attribute–value combination can ensure that the next iteration will be narrowed by excluding such reports. Being able to select from a menu in this way saves the user the trouble of learning the syntax for the underlying query language, or the available attribute names or values.

To assist in deciding what the constraint should be, the user can request a listing of all the values of that attribute that appear in the data. Figure 3.2 shows this activity as the submission of 'parameter queries' to the database engine, resulting in additional 'parameter ranges' information in the illumination zone.

In Rabbit (the first retrieval-by-reformulation system) additional criteria could *only* be introduced by menu-based interaction with displayed item attributes. An advantage of this restriction was that a user could never specify an invalid constraint. But in Helgon the restriction is relaxed, allowing users to add hand-built criteria. For inexperienced users the menu-based reformulation mechanism remains easier and safer, but experienced users who know how to express some constraint directly are saved the trouble of having to find a display of the attribute they wish to constrain.

Experienced users may also wish to take advantage of other commands that can be invoked by use of a command line. For example, a user who is interested in finding technical reports from the University of Colorado, but who is unsure of how the institution name may have been abbreviated (e.g., 'U. Colorado'), can find out by typing the word 'Colorado' and requesting a search for all stored Institution objects that include this substring.

5. The user decides when to request a new retrieval corresponding to the updated result criteria, having made as many additions or alterations to the criteria as are wanted. Progress continues from step 2.

   The eventual goal is to arrive at criteria that will deliver into the illumination zone a small set of items that includes the best item for the current need.

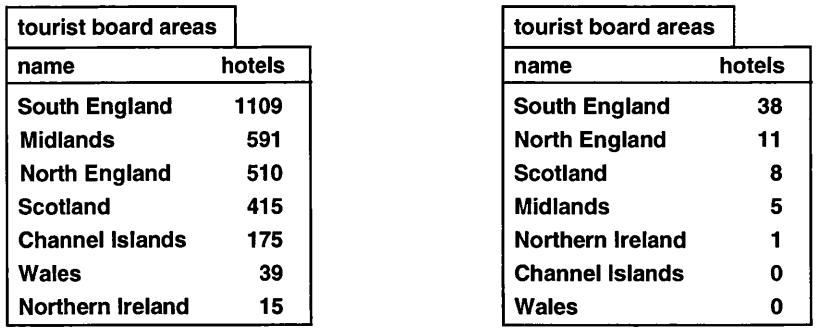
One limitation of Helgon's facilities is that there is no way to tell how large a difference to the search a given reformulation will make. How many of the reports have IBM as their institution? If it were only a tiny percentage of the retrieved sample there would be little point in going to the effort of 'prohibiting' that value; one might get more worthwhile selectivity by constraining, say, the year of publication instead.

The systems described below provide examples of such information.

- **Helping the user distinguish alternative refinements**

  An example of a system that goes some way to overcoming blindness to the impact of a reformulation is HIBROWSE (Pollitt, Ellis, Smith and Li, 1994), an enhancement of the earlier Query-by-Menu DBMS query system (Pollitt, 1986). As well as listing the values available for a selected attribute, HIBROWSE can maintain context-specific summary lists for a number of attributes, showing against each of the attribute values the number of current candidate results having that value. Figure 3.4 shows one of these summary windows at two stages in a query for hotels from a large commercially-maintained database[1]. This window shows the distribution of

[1]The UK Hotel Groups Directory, a relational database maintained at the Hotel and Catering Research Centre at the University of Huddersfield.



tourist board areas	
name	hotels
South England	1109
Midlands	591
North England	510
Scotland	415
Channel Islands	175
Wales	39
Northern Ireland	15

(a)

tourist board areas	
name	hotels
South England	38
North England	11
Scotland	8
Midlands	5
Northern Ireland	1
Channel Islands	0
Wales	0

(b)

Figure 3.4 Contents of a HIBROWSE summary window (a) at the start of a search, and (b) after the search has been narrowed. (From Pollitt *et al.*, 1994)

candidate results against the 'tourist board area' attribute: before any constraints are applied there are clearly candidates available in all areas, while at a later stage (when the user has narrowed the query by requiring large, 4- or 5-star hotels) the summary shows that some areas are no longer in the running at all while others have changed relative number of candidates. Without this information, a user might have wasted effort by trying to require or exclude hotels in Wales or the Channel Islands, or have failed to appreciate that prohibiting the South England items removes over half of the remaining candidates.

A further apparently useful level of illumination is demonstrated by Godin, Gecsei and Pichet (1989) in their prototype interface to an information retrieval domain structured using a Galois lattice. The lattice relates couples containing a set of documents and a set of terms, where the term set is the *largest* set of terms that will result in the retrieval (by boolean query) of exactly the document set. Restricting to the largest term set keeps the overall size of the lattice manageable, and makes the transitions between elements correspond to minimal refinements of the query.

Godin *et al.* illustrate the system with a very small database of art images containing six items (numbered 1 to 6), each with up to five terms describing the topic, artist and location of the original painting. In the resulting lattice two connected elements are the pairs ({1,2,3,4}, {Venus}) and ({3,4}, {Venus, sleep}), from which we can tell that: a) there is no search term that, in addition to 'Venus', will be satisfied by the four images {1,2,3,4}, and b) there is no set of terms that will result in retrieval of sets {1,3,4} or {2,3,4}.

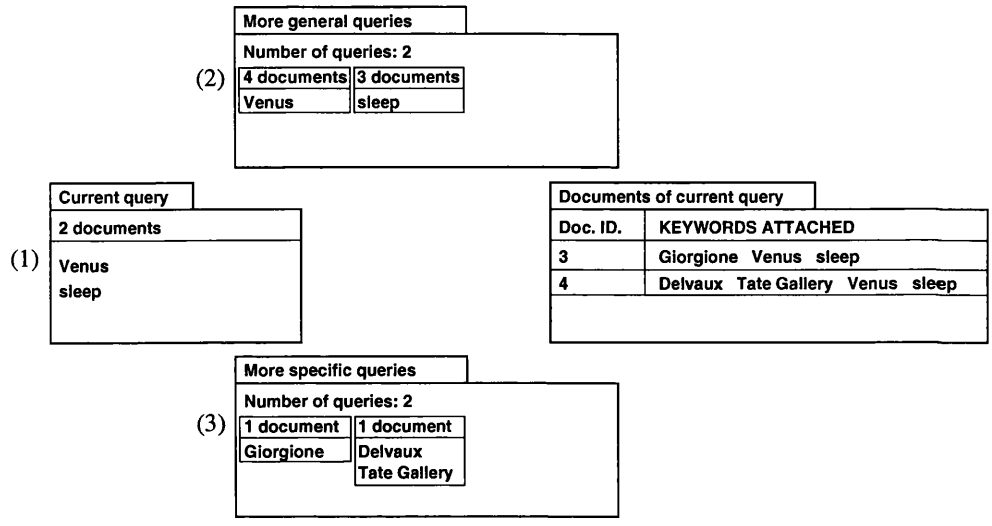


Figure 3.5 Contents of the main interface at one stage in navigating a Galois lattice (From Godin, Gecsei & Pichet, 1989)

Figure 3.5 shows a snapshot of the contents of the main windows in the prototype interface, during a search in which the user has reached the element containing documents 3 and 4. This shows that, as well as information on possible ways to narrow the search (window 3), this interface provides information on how to generalise (window 2). In this case reducing to just the 'Venus' term gives the set {1,2,3,4} in accordance with deduction (a) above; reducing to just the 'sleep' term gives a set containing three documents although this cannot be either of the cases mentioned in deduction (b).

If an analogous interface were available for the hotel search, one example of a use for it would be in taking the results at stage (b) in figure 3.4, constraining the search to Wales (with its currently empty result set) and obtaining an illumination of the constraints whose relaxation would restore the query to a non-zero size.

There is a separate issue as to whether this particular lattice structure provides a model that users find intuitive. Godin, Missaoui and April (1993) report on a promising but small-scale experiment comparing user performance with the Galois lattice against navigation in a manually built hierarchy or pure Boolean querying.

Another prototype interface, that illuminates the possible directions of progress in both the specialisation and generalisation directions but does not reveal the impact on result set size, is the Aggregation/Generalisation Hierarchy implementation discussed by Weiland and Shneiderman (1993).

Exploration progress

- **Illumination**

  The 'direction illumination' strategy is distinguished by its form of illumination, in that the displayed results contain or can be supplemented with information about available ways of refining the result specification. Direction illumination can be regarded as making available within the illumination zone some summary information about features of the result space just beyond the current viewpoint.

  In some systems the direction illumination has a fixed form that is generated automatically—for example, the Galois lattice specialisation and generalisation summaries shown in figure 3.5.

  In retrieval-by-reformulation systems some direction information is provided implicitly in the display of result characteristics beyond those that were used as item-specification criteria. Williams (1984) explains that Rabbit has a knowledge base allowing it to display the attributes that are 'associated with the generic categories the user is committed to in his query'. For example, when a user has only specified that the item sought is a *product* the illumination shows just *cost* and *manufacturer* attributes, but if the user then narrows the specification to *computer* the results are shown with the additional computer-related attributes *display, memory* and so on.

  Helgon additionally allows the user to make explicit requests for information about parameters and their values. This is a lower level of support that is more flexible than the examples above, but cannot necessarily provide as much help with evaluation.

  All the reviewed systems only support an illumination zone containing the results and direction illumination relating to a single viewpoint—i.e., a single result specification. When the user changes the viewpoint the new illumination takes the place of the old.

- **Evaluation**

  As was noted for the trial and error strategy, the fact that the illumination zone can only contain information relating to a single viewpoint makes this strategy poor in its support for comparative evaluation between results. What some of the systems do provide is information that can help the user make comparative evaluations of nearby regions of the result space as a whole. For example, the HIBROWSE display of result counts against property values helps the user to compare alternative directions, and hence to avoid straying into a region that has no results or wasting time by specifying criteria that have insignificant impact.

In terms of expended effort against expected accuracy, such information helps to increase the efficiency of effort deployment. But because the available information cannot be tailored to reveal particular properties of the results themselves, there is an unavoidable risk that the regions a user decides *not* to explore might happen to contain the best results.

- **Consolidation**

  The provision of direction illumination lends support for two aspects of consolidation: 1) the system has an explicit manifestation of the available directions for refinement, so the user can specify the desired change just by pointing to the appropriate displayed element, and 2) when the display includes information by which to discriminate the different directions, the user can make an informed choice from among them.

Result-space coverage

Because the premiss of result-specification refinement is that the search can be focussed progressively to reach a narrow specification that includes just a few supposedly optimum results, such strategies inevitably require the user to rule out large areas of the result space. Furthermore, the motivation to minimise effort in the search encourages the adoption of dramatic filtering steps, chosen with a minimum of evaluation effort between each step. This brings a high risk of under-informed outcome. To some extent, direction-illumination systems may even exacerbate this effect.

Opportunity to delegate

Direction illumination is suitable for domains for which the designer cannot predict the nature of the user's overall goals, but can predict the nature of information the user needs in order to decide on a direction of progress. Either this information is made available automatically without request, or some query facilities are provided. This provides assistance in performing comparative evaluation of some aspects of a local region of the result space, but can only be provided for one location at a time—so as with the trial-and-error and computer-critiqued forms of investigation, for evaluating diverse results the user just has to remember the properties of the results that have been seen. Finally, consolidation is easily expressed to the system, but only in some domains is its effect predictable.

3.2.4 Refinement by human critiquing of computer-generated results

The systems reviewed in the previous section can illuminate the available directions of progress because exploration is taking place within an existing, pre-structured space of results. By contrast, in tasks such as computer-aided design the subtlety and variety of available options lead to a result space whose overall layout cannot be pre-defined or enumerated.

Examples

- **Cooperative Computer-Aided Design**

  Kochhar (1994) describes the Cooperative Computer-Aided Design paradigm (CCAD) in which—like the 'direction illumination' approach—the computer presents alternatives to the user, who then makes critical design decisions. The key to coming up with a valid, finite set of alternatives from an infinitely rich search space is for the computer to *generate* alternatives on the fly, based on a short-range foray from a user-selected 'viewpoint'.

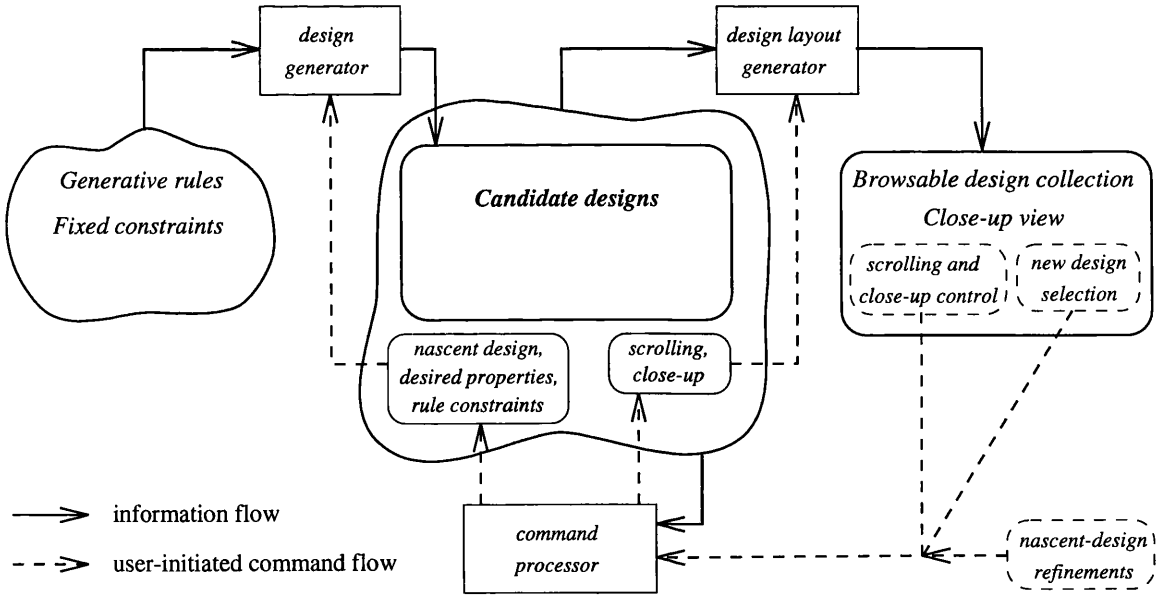


Figure 3.6 Illumination Zone Model for Cooperative Computer-Aided Design.

Figure 3.6 represents a CCAD system using the illumination zone model. Kochhar's description of exploration progress using CCAD is as follows:

1. The user provides an initial *nascent design* to serve as a starting-point. The construction of this design is outside the scope of the exploration, and may even be performed using a different system.

   In the generic illumination-zone terminology, the nascent design constitutes the current viewpoint.

2. The user specifies to the computer some desired properties of the developed design, in terms of constraints on the design and/or on the generative rules used in its derivation.

   These instructions and the nascent design make up the illumination specification.

3. Using some production formalism suited to the domain being supported (e.g., a generative grammar), the design generator develops alternative evolutions of the nascent design that are consistent with the constraints specified by the user, and any domain-specific fixed constraints—for example, in designing architectural plans the fixed constraints would include building regulations.

   The designs are collected in the illumination zone.

4. The design layout generator presents the candidate designs arranged according to properties selected by the user, and provides facilities for obtaining a close-up view of a few of the designs side by side.

5. The user selects one candidate design as being the most promising. This design becomes the basis for the next iteration.

   As was the case for the start of the process, at this stage the user may call on modelling facilities that are separate from the search support, to make modifications that are not suitable for, or don't need, exploration. When the next piece of exploration is to be undertaken, the latest design takes the place of the previous nascent design.

   Steps 2 to 5 are repeated until the user is satisfied with the completed design.

Kochhar gives details of a tool called Flats, a prototype CCAD system for the design of small architectural floor plans. The user is able to control the scope of the system's design generation by means of facilities that Kochhar refers to as *constraint-based design pruning, design-rule restriction, grammar programming* and *resource bounding.*

The description of Flats includes a system architecture schematic which is analogous to the illumination zone model, the main difference being the explicit inclusion of the independent modelling system as the source of the nascent design that is supplied to the design generator. Kochhar also places CCAD within a taxonomy of models for division of labour in computer-supported design tasks, as developed in his ongoing

work with colleagues (see Kochhar & Friedell, 1990 and Kochhar, Marks & Friedell, 1991). Woodbury (1991), reporting from the perspective of the architectural design profession, gives formal bases for many aspects of the *search paradigm,* an approach to computer support in design that would encompass CCAD. As an exemplar of the paradigm he describes LOOS (e.g., see Flemming, 1989 and Flemming, Coyne, Glavin, Hsi & Rychener, 1989), a system for setting up exhaustive searches for designs that can be meaningfully approximated by a set of non-overlapping rectangles.

- **Evolutionary development**

  Another system that generates alternatives for the user to evaluate is Mutator (Todd & Latham, 1992), a tool for creating artistic 3-dimensional graphical sculptures based on constructive solid geometry.

b+d + r1	b	b+d + r2
b+d + r3	b+d	b+d - r3
b+d - r2	b + 2*d	b+d - r1

Figure 3.7 Layout of a Mutator display of a starting-point and eight alternative mutations, for comparison by the user. Code: b=previous best; d=direction vector; r1–3 are random mutation vectors.

A Mutator sculpture is composed of geometric primitives such as coils, cylinders and spheres, that can be merged and distorted in various ways. At each iteration the tool *mutates* the current state of the design by applying randomised changes to the parameters (*genes*) that determine the size, orientation and distortion of each of its parts. Mutator can support various forms of generation of mutations, including facilities for *steering* and *marriage* under user control, and various displays to allow the user to evaluate, compare and critique the alternative mutations. Figure 3.7 shows a layout for presenting eight mutations based on a specified *best* version from the previous generation, a *direction* determined by aggregation of all positive and negative critiques supplied by the user, and three alternative random mutations whose effects in opposite directions is presented at the diametrically opposed points.

What is especially interesting about Mutator is that its simple conceptual model and user interface support the development of highly complex sculptures—containing dozens or even hundreds of primitives, each with many properties. Although it is possible for a user to access and alter the alphanumeric file containing the specification of a model, working on a complex sculpture in this way would be prohibitively time-consuming and frustrating. By working with Mutator a sculptor does not even need to learn how to edit a computer file, let alone understand the mathematical model-definition language.

Allowing the user to steer the direction of exploration simply by selecting the candidate result or results most closely embodying the desired direction of progress is a powerful technique, known as *relevance feedback.* As long as the user perceives that progress is induced in the desired direction, the complexity of the underlying mathematics needed to achieve this can be concealed.

- **Relevance Feedback**

  Another domain in which relevance feedback has been implemented successfully is free-text information retrieval, as exemplified by NRT (the News Retrieval Tool—Sanderson & van Rijsbergen, 1991). NRT allows users to search a news item database for articles dealing with topics of interest, but rather than requiring articles to be categorised using keywords, and the user to build a machine-centred query in terms of such keywords, NRT supports retrieval of free-text items using free-text queries.

  Each NRT retrieval results in a ranked list of documents that the system has evaluated as matching the query. The interface allows the user to retrieve and examine the full text of the matching documents, and to indicate one or more as being most relevant—i.e., the best match with what the user is hoping to find. This feedback is translated by the system into additional guidance to the retrieval engine, so in most cases the next retrieval will result in a list of documents in which the ones most similar to the indicated best matches are ranked most highly. Repeated selection and retrieval will strengthen the relevance weightings, bringing to the top of the ranking more and more documents that are like the ones the user has selected, until eventually the top entries in the list might all be documents the user has already rated highly.

  The simplest level of NRT interface hides from the user how all of the following tasks are actually performed:

  - analysing the free-text query used to start the exploration;
  - matching available documents against the provided query;

- producing a list of the best matching documents, ranked in order of decreasing correlation with the query;
- reformulating the query to take account of the user's selection of the documents that are closest to what was wanted.

There is a further level of interaction that NRT can provide for the use of more proficient users: an adjustable display of the intermediate form of the query, which consists of key terms that each have a weight on a continuous range from strongly negative to strongly positive. This can help the user to understand and correct some errors in NRT's interpretation of the intention of the supplied query and feedback. For example, although NRT is able to handle free-text queries it does not analyse the grammatical structure of either the query or the stored articles. Negation in a query (e.g., 'computer news not relating to American companies') goes undetected, biassing the results in the opposite sense from what is wanted. In this case use of the key term display would allow the user to detect the problem, and change from positive to negative the weighting attached to the term 'American'.

- **Interactive Knowledge Discovery**

Knowledge Discovery (KD) systems are used to search very large data repositories for unexpected patterns that may constitute worthwhile knowledge for the user. They are used, for example, in analysing financial and insurance records on a nationwide scale to find correlations between supposedly unconnected transactions that might indicate a trail of fraud. In their overview of the topic, Frawley, Piatetsky-Shapiro and Matheus (1992) identify the main challenge as reducing the noise constituted by discovered knowledge that is uninteresting to the user:

> '... a system might discover the following:
>
> `If At-fault-accident=yes Then Age>16.`
>
> To the system, this piece of knowledge might be previously unknown and potentially useful; to a user trying to analyze insurance claims records, this pattern would be tautological and uninteresting and would not represent discovered knowledge.'

One approach to this problem is for a user or designer to guide the search, by suggesting what kinds of trend to look for and where to try looking. But many KD developers take the view that such guidance removes the system's ability to find information that is completely unexpected—which, by the same token, is potentially the most valuable. So the system should find as much 'knowledge' as it can, and then let the user separate the interesting from the uninteresting items. The role of a user

in an interactive KD system is therefore to evaluate the system's suggestions and use feedback to steer the search away from areas that were simply gaps in the pre-coded domain knowledge.

Exploration progress

- **Illumination**

  The human critiquing strategy is useful in domains whose result spaces are large and of unfathomable structure, so the user will never be able to make an explicit request for illumination of a particular region. Instead, the user specifies guidelines that the system uses to arrive at a set of results that have some desired properties.

  The number of results shown in each illumination step depends on the resources that are available. Mutator is restricted by the need to lay out graphical designs on screen. Flats uses organised layouts of reduced-detail plan displays, so the limiting factor is not screen space but the number of designs the user is prepared to examine and compare. NRT, likewise, *could* produce a ranking that includes every document in its corpus—but the user is only expected to be interested in a few tens of entries.

- **Evaluation**

  The main point of a human critiquing system is that the system can generate alternatives but only the user can judge their suitability. Evaluation is entirely up to the user, typically requiring detailed examination of the presented alternatives.

  Finding the best illuminated result is crucial in a CCAD system such as Flats, since it becomes the next version of the working design. Even though the results are all nominally 'on view' together, making trade-offs between results that are arbitrarily distributed over the view can still demand a large amount of cognitive effort from the user. This effort can be partly alleviated if the user can request that results be ordered (or mapped onto a spatial plane) according to some formalised characteristics, so that at least some of the result attributes are directly reflected in the organisation. In addition the user can select a particular small set of the results for detailed viewing to help decide between them.

  In Mutator the selection of a 'best design' from the alternatives does not commit the user to that particular design, but to the direction of evolution that it embodies with respect to the previously selected best design. It is therefore especially important for the user to be able to compare the designs against the starting-point as well as against each other. This is facilitated by a layout such as that shown in figure 3.7.

Detailed contrast between results is not a crucial issue in NRT. The retrieved news articles are listed showing their headlines; if the user is not sure whether or not some article is relevant, its full text can be requested.

- **Consolidation**

  Even more than the direction illumination strategy, the human critiquing systems support consolidation by reference to the results that have been displayed. Indeed, some systems provide no other way to guide the exploration.

  Because the domains are large and unpredictable, the effect of a consolidation request is no easier for the user to follow than is the process involved in generating a set of candidate results. In a large corpus NRT is certainly effective at finding articles that are *textually* similar to those selected by the user, but there is no way of telling how many other articles that are just as relevant to the desired topic have been rated poorly because they use different terminology.

  The discarding of large portions of the result space is more plainly noticeable—though not necessarily more drastic—in the use of CCAD, in which each consolidation step involves rejection of all but one of the suggested designs. But at least in this case the user can make hand-built modifications to the design and could therefore add desired features that happen to be missing from the design that is otherwise the winner.

Result-space coverage

Being another refinement-based strategy, human critiquing relies on discarding all but a single direction of exploration. Because it is used in domains that are too large to be enumerated exhaustively, there is no way to provide the user with a feel for the regions of potential results that have been missed.

In some systems the user is given facilities for guiding the illumination to exclude results that are known in advance not to be useful, but those results that are illuminated must be viewed in the system's default presentation style. Where the judgement of a result depends on detailed examination, the effort involved in performing these examinations will discourage evaluation of a large number of results.

Opportunity to delegate

The user delegates illumination (result generation), which can be constrained according to those factors that the user knows in advance and that the designer built into the result

generator. The outcome, which may include many results that are undesirable but could not be described as such to the result generator, must be evaluated entirely by the user. Each illumination generates a number of results that can be from diverse points in the result space, so there is some scope for interesting side-by-side comparisons. Feedback for consolidating exploration progress is expressed in terms that make it easy for the user to specify, but difficult to predict how the exploration will be affected.

3.2.5 Dynamic queries and other browsable result mappings

This section reviews some systems that support a strategy based on dynamic, user-controlled browsing of a mapping of an entire result space.

Examples

- **Map-based dynamic queries**

  Some of the leading work on the use of result mappings is that embodied in the Dynamic Query (DQ) tools pioneered at the University of Maryland[2].

  Figure 3.8 is a view of the Dynamic Homefinder, a system for potential home-buyers to explore properties on the market—in this case in Washington, DC. The homes shown as dots on the map are just those that satisfy the 'query' defined by the user using the sliders on the right: here the homes must have from two to four bedrooms, and cost between $160,000 and $380,000; the query has also been constrained to include only the 'house' properties (rather than townhouse or condominium), and to require a garage (other features are Fireplace, Central Air Conditioning, and New construction). Finally, the homes are constrained to be within specified distances of the two reference points 'A' and 'B', which the user might have placed at office and school locations.

  The power of the system comes from the apparently instantaneous update of the view (e.g., within 100 milliseconds) in response to changes in any constraint. For example, as the user drags the 'Dist to B' slider to the right, additional dots appear at the expanding perimeter of an imaginary circle centred at B. Since the display depends only on the current positions of the sliders, and not on any query 'history',

[2] A review of the principles, early work and suggested research directions can be found in (Shneiderman, 1993); for details of the *tight coupling* extension discussed below see (Ahlberg & Shneiderman, 1994) and (Ahlberg & Truvé, 1994). Ahlberg and Wistrand (1995) introduce IVEE, a tool for automating the creation of new DQ interfaces for new database domains. The original development of the HomeFinder is covered in (Williamson & Shneiderman, 1992).

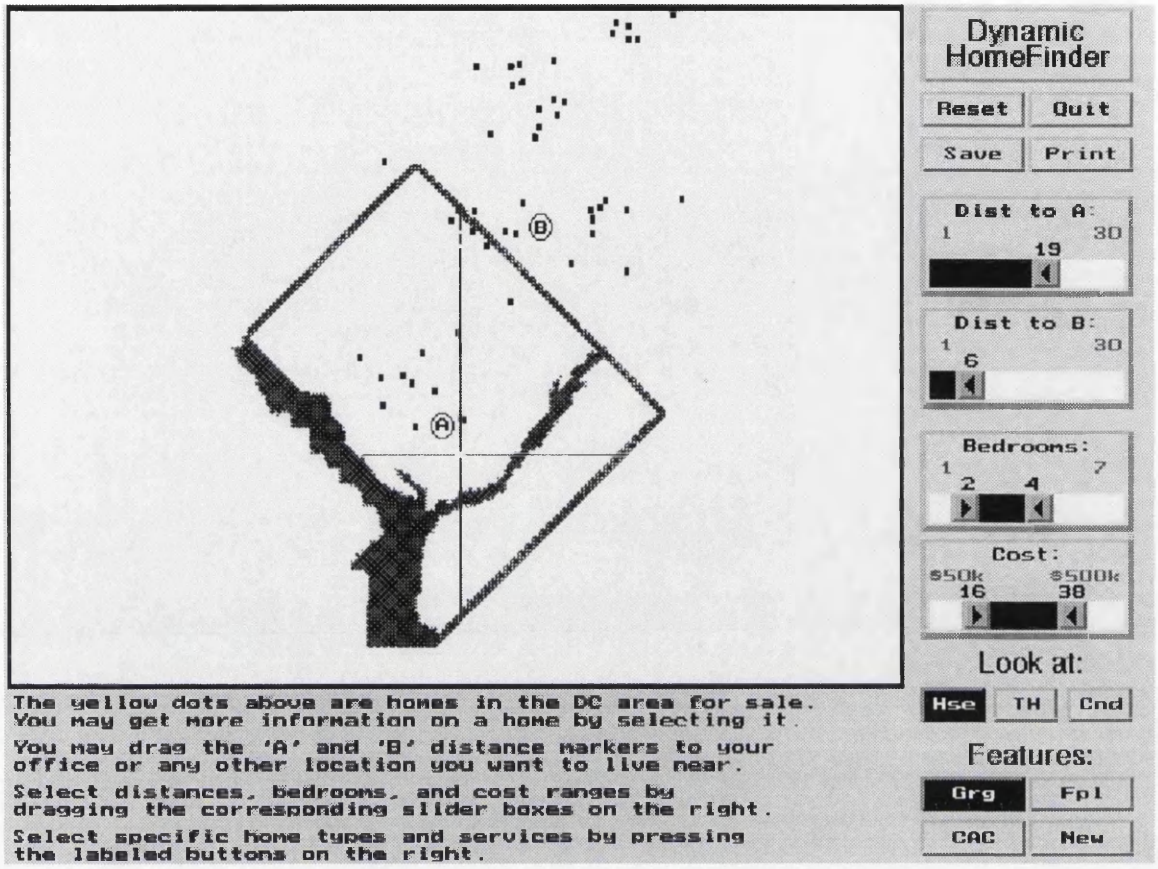


Figure 3.8 The Dynamic Homefinder for homes in and around Washington, DC. For orientation the map shows the city boundary, whose south-west edge is defined by the Potomac river. The 'yellow dots' mentioned appear as dark grey in this image. From (Shneiderman, 1993)

all constraint changes are reversible. This means there is no risk in experimenting: the user can change a constraint quickly or slowly, skimming or scanning; if the result is pushed through and beyond an interesting value, the user just needs to back off the change until that value is found again.

The essence of this interface is the manifestation of search parameters in a *direct manipulation* form, which Ahlberg and Shneiderman (1994) summarise as involving:

- visual representation of the world of action, including both objects and actions;
- rapid, incremental and reversible actions;
- selection by pointing (not typing);
- immediate and continuous display of results.

The issues that need to be addressed for dynamic queries to be implemented in any candidate domain are summed up by Ahlberg, Williamson & Shneiderman (1992) as follows:

> 'A good visualization must be found, such as a map, an organization chart, or a table, with good color combinations for highlighting. The control panel manipulating the query must be placed in a logical way to reduce eye and mouse movement. Sliders must be implemented so they are easy for novice users to use. ... The search time must be immediate so that users feel in control and have a sense of causation.'

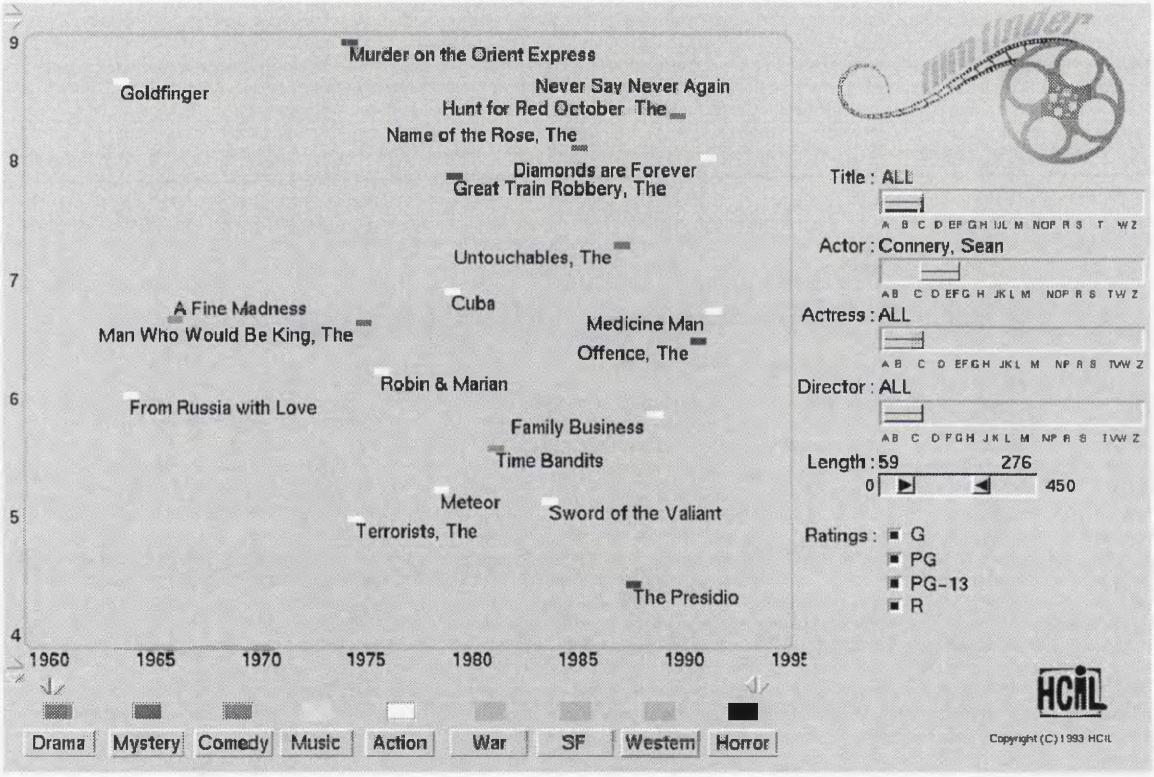


Figure 3.9 The FilmFinder (from Ahlberg & Shneiderman, 1994)

The FilmFinder, shown in figure 3.9, extends basic DQ with *tight coupling*. As usual in DQ, when the user specifies a constraint on some scale—e.g., that only films made before 1935 are of interest—all the other films disappear from view. But with tight coupling, all the scales are also updated so they contain just the values that occur within that reduced set of films—so in this case the 'Director' slider would no longer

include, say, Francis Ford Coppola. The idea is that the user is interested in incremental narrowing of the search space, so the system should help by revealing only the constraints that will still leave some results on view. Some implications of this assumption are addressed later.

Unlike homes in the HomeFinder, films do not have an obvious spatial mapping that most users will want to use. So FilmFinder includes a 'starfield display', a 2-dimensional spatial layout that can be based on any mappings that will be of interest to the user and that will give reasonable result separation—in this case, the x-axis shows the date and the y-axis a measure of film popularity.

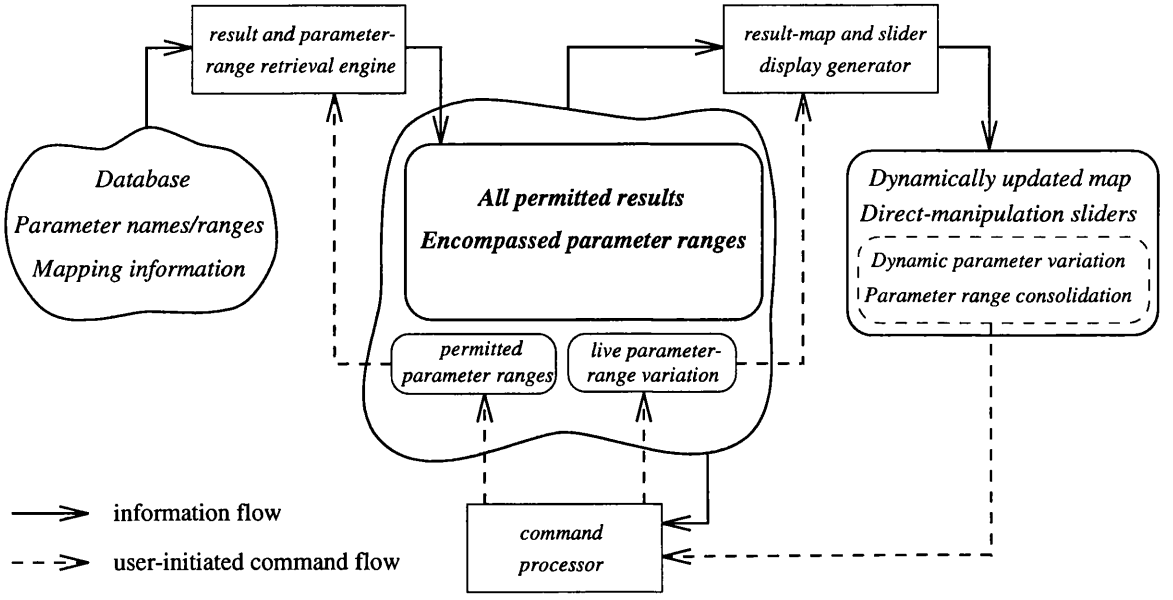


Figure 3.10 Illumination Zone Model for dynamic query.

Figure 3.10 shows one way of describing (tightly coupled) DQ using the illumination zone model. Because basic DQ explicitly seeks to remove the user's perception of a distinction between result retrieval and display, the concept of an 'illumination zone' as an intermediate stage is unnatural. However, when using tightly coupled DQ the illumination zone serves a natural purpose of holding the subset of results that is currently permitted by the narrowed settings of the sliders.

Explorations using a tightly coupled DQ system necessarily involve notions of iteration and search progress that are more constrained than the behaviour of basic DQ, and care must be taken to ensure that the user understands the implications of each step. An exploration proceeds as follows:

1. Initially the illumination zone contains all results, and the display generator presents sliders that all show the full available ranges of their parameters.

2. The user can pick up a slider to experiment with dynamic, reversible variation in the constraints on the parameter it affects. While the slider remains under the user's control the set of results contained in the illumination zone remains fixed, but the display generator excludes from the result map display those that fall outside the instantaneous parameter bounds set by the slider.

   The provision of tight coupling suggests that during this time not only the result map should be changing, but the other sliders too. But Ahlberg and Truvé (1994) found that users have enough trouble concentrating on the impact of their selections on the starfield display; having all the other elements of the display continuously updated constituted a distraction whose cost outweighed the benefits. They have proposed that it is therefore not appropriate for the other sliders to be adjusted dynamically during use of a slider.

3. When the user *releases the slider being moved,* this constitutes a greater level of commitment to the parameter range controlled by that slider. This is therefore deemed to be an appropriate stage at which to update all the other sliders to show ranges that are consistent with the newly set range of the adjusted parameter.

   But the further question arises as to whether to update the slider that has just been moved. For example, if the user has just made a choice from the 'Director' slider, the only way to make that slider fully consistent with the current set of displayed results is to reduce its own range to just the selected value. But Ahlberg and Truvé note that this would have a highly undesirable consequence: if the user's selection of the director had been merely tentative—as all dynamic query selections are allowed to be—it is unacceptable to remove the possibility of trying an alternative. So while the user is experimenting with just a single slider, its range remains unchanged from the state that was in force when the user picked it up.

4. When the user, having moved one slider, *starts moving a different slider,* this is deemed to finalise the setting on the earlier one. At this stage, therefore, the range on that previous slider is updated to be consistent with what has been selected—for example, showing just a single selected director.

   Compared with the dynamic movement of sliders this is a relatively irreversible action—a consolidation in the exploration progress. Its impact is therefore appropriately represented in the illumination zone model as a change to the contents of the illumination zone itself.

- **Variable-space mappings**

  Another example of a dynamic query interface is the prototype Attribute Explorer (Tweedie, Spence, Williams & Bhogal, 1994). Following the example of the Dynamic Query research team, Tweedie *et al.* chose to demonstrate the system on a house-search scenario. But the Attribute Explorer, being designed for multi-dimensional domains in which it would be pointless to have to nominate a single pair of parameters from which to construct a 2-D mapping, dispenses with the DQ approach of having a separate map and sliders.

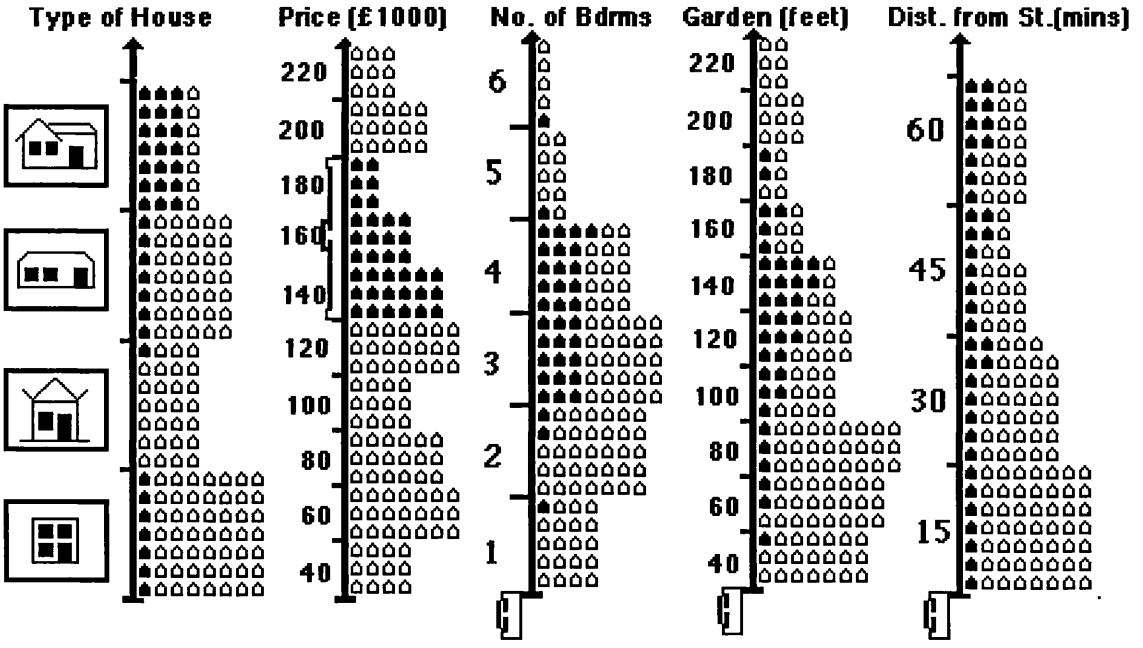


Figure 3.11 The Attribute Explorer, working on a house-search data set. The user has asked to highlight all homes in roughly the £140,000 to £180,000 price range.

The Attribute Explorer display is based solely on a representation of the parameter ranges, all the results being displayed in stem-and-leaf plots along the parameter axes. This allows it to show trend and correlation information among the variables, as is seen in figure 3.11. Each of the small house symbols that make up the histograms corresponds to an individual house instance, so as well as seeing at a glance that very few (precisely five) of the properties have six bedrooms, it is clear that the selected price range encompasses only one of those properties—most properties in that range having three or four bedrooms instead.

Note that this histogram-based result display does satisfy Ahlberg, Williamson and Shneiderman's (1992) requirement, expressed above, that 'the control panel ... must be placed in a logical way to reduce eye and mouse movement'. It may be regarded as a significant contribution that this is achieved without the system designer having to become involved in selecting a specialised map-based visualisation that would suit the properties of the particular results.

- **Information workspaces**

  An alternative form of result mapping is exemplified by the Information Visualizer project being pursued at Xerox PARC. An initial definition of the aims of this work is given in (Card, Robertson & Mackinlay, 1991), and a progress report in (Robertson, Card & Mackinlay 1993). Some recent additions to the tool portfolio are described in (Rao & Card, 1994) and (Mackinlay, Robertson & DeLine, 1994). Themes being followed in this work include:

  - maximising the capacity of the 'information workspace'—the cache of data that is within the user's immediate grasp—for various generic structures and information types including hierarchies, time-based data, document collections;
  - construction of novel 3-D mappings that can be accessed effectively on a 2-D screen, including various forms of 'fish-eye view' that appear to map distant surrounding context into distant portions of the display;
  - facilities for smooth change of viewpoint around and within 3-D models, respecting cognitive characteristics of users such as susceptibility to disorientation as a result of sudden changes.

Exploration progress

- **Illumination**

  The speciality of DQ tools can be seen as letting a user explore the properties of a large number of results, quickly and without great effort, by interactive sweeping of an illumination region back and forth across the result space.

  The results must either exist already, or must be generatable instantaneously. Because of the ever-increasing processing power of desktop computers, the scope of the kinds of result that can be generated within the necessary time constraints is growing. In addition, specialised techniques can be applied in particular domains in which processing can be localised to the level at which it can be performed quickly but will still be informative to the user: for example, in some painting and page-layout tools

the interface controls for adjusting some parameters no longer employ the dialog-box style of picking a setting and pressing an 'Apply' button, but instead provide sliders that dynamically update the presentation of at least a small sample area sufficient for the user to judge the overall effect.

But there remains a research issue in figuring out how to support interactive variation along more than one dimension at a time—to provide effective illumination of an area or volume, rather than just along a line. Interaction widgets that can map continuous variation in two or three dimensions offer some possibilities, although their additional modes of freedom compromise two important aspects of interacting with single-dimensional sliders: (1) the *exhaustiveness* of the coverage, in that it is not feasible to move a selector through every point in a multi-dimensional space; (2) the *reversibility,* in that if the selector is moved through some point that gives rise to a particularly interesting result display, it is not necessarily easy for the user to back up and find that point again.

- **Evaluation**

  The approach embodied in dynamic browsing tools is to utilise the human perceptual ability to make comparisons based on rapid incremental changes to the presentation of a supposedly constant underlying object or scene. Thus, although the Dynamic Homefinder cannot illuminate different parts of the result space simultaneously, the user can gain an understanding of the influence of any single result parameter by varying it and observing the corresponding changes in the display.

  The Attribute Explorer feature of having static plots of result distribution along each dynamically variable parameter scale provides a simple but useful additional level of trade-off visualisation, since the user can see at a glance the global distributions that in normal DQ would require experimentation with each of the property sliders in turn.

- **Consolidation**

  In plain Dynamic Query (e.g., the Dynamic Homefinder) the user is free at all times to narrow or broaden any aspect of the specification; there is no explicit notion of commitment attached to a narrowing of the exploration focus. But the addition of tight coupling brings the issue of progressive commitment as the range of each parameter is reduced.

  The fact that dynamic query can (currently) only be pursued along one dimension at a time militates against keeping open a variety of options. Rather than attempting a painstaking sweep across interacting ranges of multiple parameters, the user is encouraged to narrow the search by constraining dimensions independently.

Result-space coverage

The effort in finding out about a large number of results is obviously far reduced for systems that can support DQ—but only along one dimension at a time. The evident 'mechanical advantage' of the technique inevitably leads it to be applied to tasks with the largest result spaces, that are also likely to be multi-dimensional; in such domains, illumination according to one parameter at a time, however rapid, still requires enormous user effort in order to achieve a good proportion of coverage.

Tightly coupled DQ is subject to a task- and domain-specific issue concerning the implication of receiving an empty query result. In some tasks an empty result is entirely unhelpful, while in others it serves the purpose of highlighting that there are no results having the specified criteria. The motivation for tightly coupled DQ is to save the user from having to trawl the entire range of some parameter once an existing constraint has narrowed the values it could take and still produce some query results. For example, having selected that a movie should be from before 1935 the user can be saved an enormous amount of fruitless wandering through the 'directors' scale if it immediately excludes those people who were not being directors—perhaps weren't even alive—in the specified time range. But the effects are more wide-reaching and might not always be what the user wants. In this example, the user might indeed be interested to distinguish between those directors who were not yet alive and those who had just not yet started their careers—perhaps they had a period of acting before they started directing. By analogy, a lending-library database that only displayed the books currently on the shelves, deliberately hiding loaned-out books on the grounds that showing them would suggest they were now available for borrowing, would sometimes be extremely irritating.

Opportunity to delegate

Dynamic Query is currently only suited for domains in which:

- the parameters the user will want to vary are known in advance (for example, being the columns in a database);
- there is a suitable result-display mapping (that may also need to be designed in advance) on which the impact of changing any parameter is readily discernable;
- the results can be produced and mapped quickly enough to allow interactive variation;

- the result space includes trends (rather than being entirely stochastic) on at least some dimensions, so the user stands a chance of locating preferable regions within those dimensions of variation;
- the result space does not involve extremes of trade-off reversal, such that narrowing on the basis of a good region for one parameter might inadvertently rule out the very best results when considered on the basis of other parameters.

3.2.6 Summary: support for thorough opportunistic exploration

Summarising the above, how would a user wishing to undertake a thorough opportunistic exploration find the supportiveness of systems supporting each strategy? There are two main issues to this: (1) Thorough exploration involves looking at a large number of results, so it is important to note which of the aspects of exploration would have to be carried out entirely by the user for each result. (2) Because the exploration is opportunistic, any system facilities that cannot be extended or redefined on the fly are likely to be unsuitable at least some of the time.

1. Trial-and-error:

   Exploration is entirely up to the user; no aspects can be delegated to the system.

2. Computer critiquing:

   The user is entirely responsible for *illumination. Evaluation* is handled by the computer, but normally assumes pre-determined evaluation criteria; it is theoretically possible (but probably laborious) for the user to build or extend the evaluation rule-base opportunistically to deal with new criteria that are found to be of interest. *Consolidation* support, if any, is likewise restricted to pre-coded rules for resolving detected problems; real moving-on in the exploration is up to the user.

3. Direction illumination:

   In this case the user is entirely responsible for *evaluation.* The computer provides basic facilities for *illumination* and *consolidation*, but they are usually fixed and therefore too restricted for an opportunistic exploration.

4. Human critiquing:

   Again the user is responsible for *evaluation.* The computer is specialised to supporting wide-range *illumination,* and interpretation of the user's critique to assist in *consolidation,* but again these facilities may be too restrictive if the user wishes to pursue unexpected exploration directions.

5. Dynamic queries:

   Although some mechanical advantage is offered for both *illumination* and *consolidation*, *evaluation* is usually entirely up to the user.

Every strategy makes it laborious to examine many results because of lack of delegation support for either illumination or evaluation of results (or both). A lack of support for opportunistic illumination prevents the user from obtaining mechanised help in illuminating whatever region of the result space appears to be promising. A second impediment is the lack of support for evaluation in the form of detailed, trade-off-based comparison of an arbitrary collection of 'best so far' results: this kind of comparison, which is best supported by a side-by-side presentation of the candidates, requires mechanised support in summarising results according to properties that are found to be of discriminatory value, and arranging arbitrary collections of such summaries into a single display (unachievable with most consolidation facilities) so they can be compared side by side.

3.3 A formalisation/delegation taxonomy

The preceding section has suggested that the principal cause of problems in carrying out a thorough search is a lack of opportunity for the user to delegate to the computer—either to carry out repetitive variations on requests, or to collate results for viewing side by side. As noted in section 2.1.1, the capability of a computer system to allow delegation depends on the extent to which the activity can be *formalised,* i.e., expressed using a notation, and operations for manipulating the notation, that can be represented on the computer.

Some tasks are more difficult to formalise than others. This section expands on the issues surrounding the chance to formalise exploration in a given domain.

Figure 3.12 is a mapping of the systems reviewed in this chapter, correlating the specificity of the tasks they are designed to support against the extent to which the system dictates the direction of task progress. The following subsections explain these scales.

3.3.1 Specificity of supported tasks

Every man-made tool can be seen as occupying some position on a scale from general-purpose to fixed-function usage. The more specific the intended usage, the more the tool can be designed to suit the task—consider, for example, a vegetable peeler compared with a cook's knife—but the more constrained the possible uses of the tool.

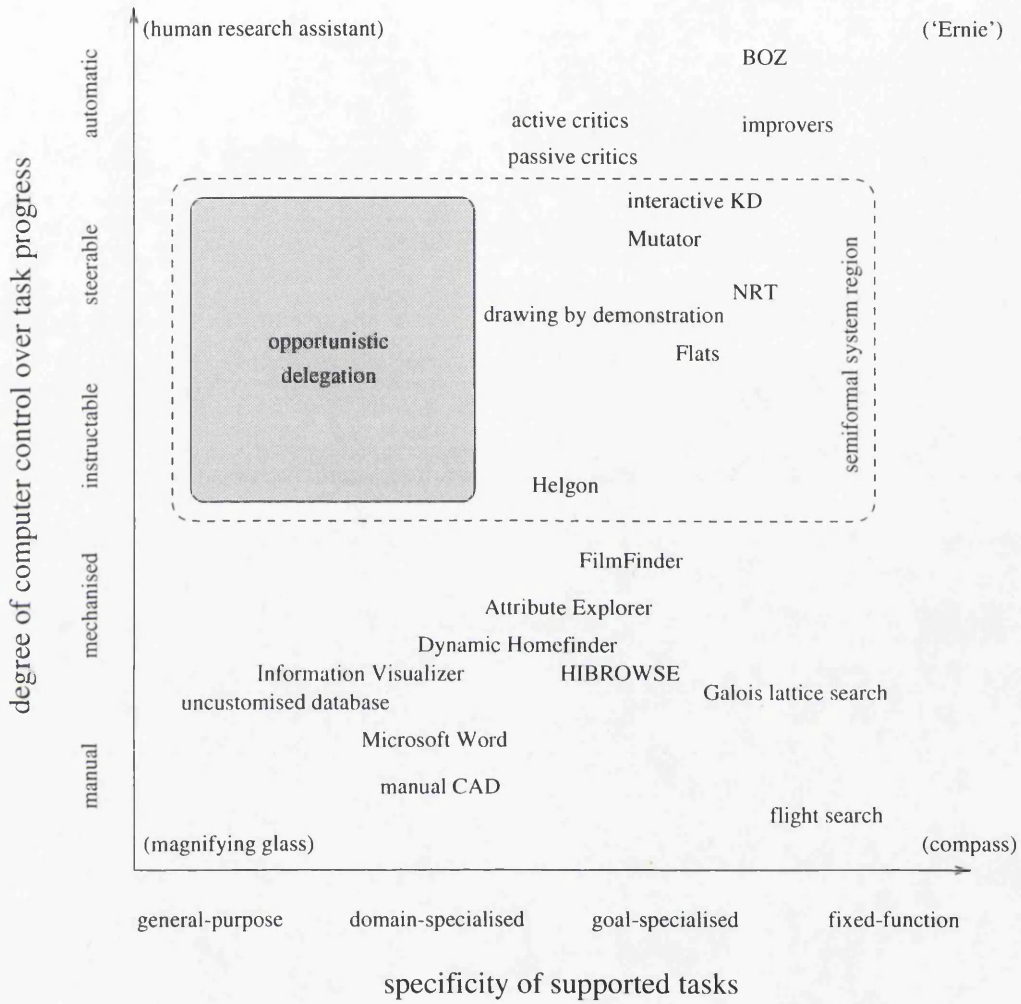


Figure 3.12 A provisional taxonomy of exploration tools, showing task-specificity against automation. The items at the corners, in parentheses, are intended as archetypes for the extremes of the two dimensions; 'Ernie' (Electronic Random Number Indicator Equipment) performs the single fully automated task of drawing winning numbers in the UK's weekly Premium Bond draw. The dotted box represents the region of this space within which all 'semiformal' systems would reside (see text, p. 68).

In our taxonomy of computer support for exploration, the closest to a *general-purpose* tool would be one that supported any domain whose data could be expressed in a simple, standard format—for example, a completely uncustomised interface to a relational database. A *domain-specialised* tool is one suited to a range of exploration activities within a given domain, such as text processing; a *goal-specialised* tool incorporates specific models of what a user is likely to be trying to achieve, rather than just providing operations to be invoked (by analogy, declarative rather than procedural support). Finally, a fully *fixed-function* tool would support just one task, guiding the user to a result using a method that is hard-coded into its design.

Nardi and Johnson (1994) have made a start on illuminating this scale, conducting a study of users' preferences for generic vs. task-specific tools for preparing and maintaining presentation slides. One of their expectations was that, like a dedicated chef, people for whom slide-making was an important and frequent task would spend the money and make the effort to learn how to use specialised slide-making tools rather than making do with generic tools such as text formatters or 'painting' packages. But their findings included, firstly, that the generic/specialised scale is not clear-cut and, secondly, that the professional slide-makers in their study group were often *not* using specialised tools, because although nominally supporting every part of the slide-making task these tools often had only mediocre support for particular aspects such as production of high-quality illustrations. Nardi and Johnson summarised their findings as follows:

> 'What we call "task-specific" software programs today really only support some subtasks relevant to a given goal. ... We found that professional slidemakers creating fancy presentations prefer using collections of interoperable tools. Their usage pattern suggests a modular set of interoperable software services as an alternative to individual highly task-specific programs, which often fall short on some crucial subtask necessary for getting a job done.'

Moving up the scale involves making progressively firmer predictions about what the user is trying to achieve. But Suchman (1987) provides some alarming examples of how difficult it is for a tool to be built to deduce a user's intentions even once a particular operation has been invoked. In particular she reveals some findings from the testing of a newly developed photocopier that incorporated a sophisticated advisory message system. The copier's software included a model of all its operations, including all predictable problems, e.g., interruption due to a paper-jam, and user-initiated exceptions such as aborting an unfinished copying job. But during the testing, situations frequently arose in which the copier software's model of its status, and the advisory messages it was therefore displaying, were at odds with the user's perception of the state of the task.

So although a photocopier designer can be certain that the system will only ever be called on for a well-defined range of *domain-specific* tasks—i.e., related to copying rather than, say, providing timed lighting for photographic purposes—it is still difficult to predict the detailed *goal-specific* intentions of a user who is performing a particular task. Above all, it may be impossible for the designer to foresee all the ways in which the user might misunderstand what needs to be done, or simply decide to change intentions part-way through a task. And, as is shown in the transcripts from Suchman's experiments, just this level of unpredictability can be enough to cause the user serious confusion in fairly common situations.

3.3.2 Degree of computer control over task progress

The grades on the second scale are described moving down the scale, i.e., in increasing order of ability for the user to apply task-specific control:

- **Automatic**—the user expresses some properties of the exploration, and the system constructs a solution that achieves, without further human intervention, some goals that the user did not express explicitly. This can be broken down according to the level of these goals:

  Automatic deduction of *end goals,* in which the user-provided specification is analysed to deduce the properties desired of the eventual outcome. An example is BOZ (Casner, 1991), a tool for building tailored presentations of a given data set according to the user's description of the task to be performed with the presentation.

  An 'improver' is an example of automatic deduction of *short-range goals,* in which the user's intended goals are deduced only at a local level based on a fully specified starting-point in the 'exploration'. An example is the page layout rules encoded in the algorithms of text-formatting tools[3]. Rather than specifying in detail the position of every word on a page, an author merely defines the higher-level organisation of the document (such as paragraphs and headings) and lets the tool calculate precisely how much white space, justification, hyphenation, etc., are needed to produce presentable pages of output.

- **Steerable**—the system's behaviour does not presume user goals, but if steered by the user the system can perform more than need be specified explicitly.

  The systems reviewed in section 3.2.4 (p. 46) mostly fall into this level. Like the text-formatting algorithms mentioned above, the processing models of these systems

[3]The TEX system, which embodies a particularly detailed model of typographic design—including iterative algorithms for exploring alternative line-break and page-break locations—is described in chapter 7.

can be highly complex, and thus beyond what the user would be able or willing to understand and specify explicitly.

The great advantage of this kind of support is that highly complex tasks can be controlled through relatively simple interfaces.

The main drawbacks are:

- A user who *does* have explicit result requirements may have no facilities for expressing them directly.

  For example, an artist using Mutator may find that a particular section of one of the presented sculptures has evolved in a very appealing way, while the other parts of that mutation are undesirable. But the system provides no way to retain or reject subsections of the workpiece.

- There are often hidden limitations of the underlying model.

  For example, as reported in section 3.2.4, NRT's apparent ability to cope with natural-language queries can be let down by its lack of grammatical analysis.

- **Instructable**—the user specifies, in terms of abstractions, precisely what the system is to do.

  Whereas *steerable* systems allow only indirect specification that must pass through hidden levels of interpretation, *instructable* tools are those that provide an explicit formalism for the user to constrain the search space. The Flats system (also described in section 3.2.4) is arguably on the border of this and the previous category.

  The power of instructable systems depends on the specification language; it must be sufficiently expressive, and the user must be prepared to use it. There have been some notable success stories in the provision of languages for end-users (e.g., spreadsheet macro languages).

- **Mechanised**—the user is saved effort in specifying the desired direction of progress, but without the use of explicit abstractions.

  This covers tools such as the Dynamic Query systems, based on 'direct engagement' facilities, and those that at least provide result-specific controls making it easy for the user to use one set of results to specify the next step in the search.

- **Fully manual**—the exploration is deemed so user- and task-specific that the tool provides no assistance in guiding any exploration.

3.3.3 Trends in the taxonomy

In reality a complex tool has various components that inhabit different parts of both the specificity and automation scales, and may also support some activities that are hard to characterise as 'exploration'—for example, a word processor with format adjustment, outlining facilities, and a spelling-checker. But since the goal here is to shed light on how the scales as a whole are related, the taxonomy has been drawn by thinking up tools that *in some capacity* occupy a given location in this Cartesian space.

One trend in the systems that are reviewed is that most of the examples at the top end of the automation scale are those with the largest search spaces, such as the creative design systems. The user could not hope to explore such domains in detail, and the automation can be seen as reducing the effective complexity of the search space to a level at which, like the database searches, it is *deemed* to be tractable. But this involves two debatable assumptions: firstly that the reduction of the search space is safe—i.e., not throwing away all or most of the good results—and, secondly, that presenting the user with a search space that is still of comparable complexity to a traditional database will result in a suitably accurate search.

The region bounded by the dotted box in figure 3.12 represents the scope of the definition of *semiformal* provided by Lai, Malone and Yu (1988) and reviewed in appendix A. Restating their definition, the criteria for a system to be considered semiformal are as follows:

1. it represents and automatically processes certain information in formally specified ways;
2. it represents and makes it easy for humans to process the same or other information in ways that are not formally specified;
3. it allows the boundary between formal processing by computers and informal processing by people to be easily changed.

The implication of the automation scale is that the tools at the high end are so specialised for automation that they do not fulfil requirement 2; these tools were therefore identified in section 3.2 as encouraging under-informed outcomes by cutting down the search space without giving the user the opportunity to make necessary choices. Similarly, the tools placed at the low end are those that give insufficient support for property 1, and thus risk under-informed outcomes because they leave the user too much of a burden. In order to provide flexibility in the boundary between formal and informal processing there must be some form of instruction, or at least steering—but note that the converse implication is not

necessarily true: provision of these facilities does not imply flexibility, as we see in systems like Mutator that have rigidly defined roles.

The region marked 'opportunistic delegation', into which none of the surveyed tools seems to fit, is the region of interest for this thesis: tools that can support semiformal interaction, but without being specialised to the extent that would allow the designer to predict the users' goals. The following section reviews some forms of computer system that can be characterised as providing this level of support.

3.4 Opportunistic delegation

Opportunistic delegation requires that the computer-based 'assistant' doesn't merely have a fixed repertoire of capabilities, but can be instructed to undertake new forms of labour-saving activity that the user realises would be useful for some task in hand.

3.4.1 Programmable environments and construction kits

In many domains of computer use it has already been recognised that it is not just the people who develop systems for other users who can benefit from being able to 'program' the computer—that is, define novel forms of processing that the computer is to provide for them. In the words of Nardi (1993):

> 'We have only scratched the surface of what would be possible if end users could freely program their own applications... As has been shown time and again, no matter how much designers and programmers try to anticipate and provide for what users will need, the effort always falls short because it is impossible to know in advance what may be needed... End users should have the ability to create customizations, extensions, and applications...'
>
> (p.3)

Spreadsheets are a domain-specific but highly popular example of end user programming, their power and flexibility being derived from their grid-like framework and automatically maintained one-way constraint mechanism. Less specialised are the various 'scripting' toolkits that allow personal-computer users to mechanise aspects of their day-to-day work.

Eisenberg and Fischer (1994) argue that the market in *domain-oriented design environments* (such as tools for creating charts and other graphics) has evolved into unmanageably feature-bloated systems that should be replaced by *programmable design environments* (PDEs), in

which the hundreds of individual specialised features can be replaced by an integrated programming language and supportive development tools.

Nardi and Zarmer (1993) suggest that a large proportion of information-intensive applications can be realised with the help of computational building blocks they call *visual formalisms:* diagrammatic displays with well-defined semantics for expressing relations. As an example they demonstrate the power of a highly general 'table' visual formalism as an application framework: providing the 'cell' as a unit for displaying, manipulating and providing programmed relations within application data (like a more general form of spreadsheet); providing rows, columns and more general regions for organisation, and so on. Johnson, Nardi, Zarmer and Miller (1993) present the prototype Application Construction Environment (ACE) that is based on development with visual formalisms, and they express clear views on who they believe should be doing this development:

> 'Traditional UI tools suggest a methodology in which applications are implemented *for* users by professional programmers and designers... Spreadsheets, on the other hand, suggest a very different methodology in which users are the *primary* implementers of applications... ACE is intended to foster a methodology more like that fostered by spreadsheets. It puts users at the center of the development process.'
>
> (p.53, original emphasis)

Nardi and Zarmer (1993) reported on the accumulation of the Visual Formalisms Library, the goal of which is 'to give developers high-level functionality for developing applications for creating, structuring, modifying, editing and browsing information—not just providing static displays.' At that stage they had implemented four visual formalisms: a panel (a container for other visual objects, e.g., suitable for creating dialog boxes); a table (mentioned above); a graph (for handling objects with explicit relations, e.g. useful for displaying hierarchies or circuit layouts); an outline (providing the structure and operations of a traditional outline processor). For assembling applications from these formalisms, Johnson *et al.* noted that although much of the construction could be handled with a framework-integrated *extension language* there would sometimes be a need for some use of a low-level language (in their case, `C++`) for creation of novel object types. When low-level programming is necessary, it inevitably compromises the accessibility of the toolkit to general end-users.

Smith and Susser (1992) discuss their own prototype, the Component Construction Kit (CCK), that demonstrates the feasibility of developing standardised 'pluggable' components such as a text editor, a spreadsheet cell, and non-visual elements such as a database retrieval

component or a spelling checker. As far as possible, their goal is for the selected components to be connected automatically by detecting conformance of their typed and keyword-coded inputs, outputs and operations. (In an appendix to their chapter, Smith and Susser list a number of examples of component-based construction kits, including Bill Budge's well-known *Pinball Construction Set* from 1983.)

Clearly, the major challenges in providing a form of toolkit are (1) the design of a set of component parts (or language concepts) that provides sufficient expressiveness and flexibility, and (2) provision of a suitable mechanism by which the user instantiates and draws together the parts needed to describe new processing capabilities.

Easing the transition from user to programmer

There is lively HCI research addressing the design of systems that can support some form of programming, but without imposing an offputting distinction between programming facilities and the straightforward interactive use of pre-defined system commands. The End-User Programming working group of a recent workshop (Myers, Smith & Horn, 1992) expressed their area of concern as being 'the transition from the pure user level, which is typically a non-programming task, to some sort of programming. Situations where this would be desirable include automation of a repetitive task; extension of the current application to perform a function not anticipated by the original program designer; or simply to customize the application and its user interface to different requirements or personal taste.'

Smith and Susser (1992), from the same workshop (also quoted above), argue that the transition can often be approached as a need for 'tailoring' the provided facilities:

> 'Every user's job is different. Users know what they want to do; developers can only guess. It will not be sufficient in the future just to give people fixed sets of functions packaged as inflexible applications. We must provide people with *tailorable* sets of functions, which they can mix and match on a task by task basis. In fact, the term "application" probably needs to be replaced by some other term such as "task environment" '.
>
> (p.35, original emphasis)

Smith, Ungar and Chang (1992) acknowledge that a user may approach a system with a different set of thoughts and goals when attempting to change its behaviour rather than simply to use it, but they note that the current tendency for programming to be described using a different language from the use (e.g., a textual language rather than direct-manipulation

operations) necessarily imposes an intrusive need to learn explicitly how to 'mention' interface elements in the programming language. They give some examples of systems that avoid this 'use–mention' distinction, and they argue that 'the lack of distinction between use and mention means that designers can easily incorporate all functionality, even "programming" as part of their interfaces. It also means that users are able to smoothly grow into programmers, and are able to tailor their interfaces to their individual tastes and needs.'

3.4.2 Systems that attempt to deduce users' goals

Current research includes a broad area that can be seen as tackling the need for opportunistic delegation not by empowering the user to *express* what is wanted, but rather by equipping the computer system to *deduce* the same. This is a characteristic of, for example, *programming by demonstration* systems (e.g., see Cypher *et al.*, 1993). Such systems, though proving increasingly powerful and useful in their particular domains, are not of interest within the scope of this thesis. The first reason, as expressed in section 2.1, is my interest in keeping the user firmly in control of task progress. The second reason is that their capabilities are necessarily limited to those for which the user's *intention* can be analysed as fitting some previously formalised pattern: this goes against my conception of opportunistic delegation, since the user may have in mind some pattern of operation that the designer had not predicted, or may even be following some direction of investigation without having any expressible 'intention' at all.

However, one of the qualities that is seen as an advantage of this form of system is that the user does not need to learn any form of language for explaining intentions to the computer. It is as if the computer is 'watching over the user's shoulder'; when the computer's analysis of the user's actions reveals some operation that could be formalised, it offers to do so. If this is the only form of assistance available then any time when the computer is not offering to help, the user should assume that no assistance is available for the task being performed. Thus there is neither opportunity nor need to switch between 'use' and 'programming' modes. Although this unexpressed form of assistance gives many opportunities for subtle 'misunderstandings' between the user and the computer's algorithms, research continues on defining domain-specific systems in which the programming can realistically be achieved without such mode-switching: Myers, Smith and Horn (1992) report that 'programming-by-example systems show that there is potential to allow a class of programming tasks to be handled without having to describe low-level details in a separate "programming" mode.'

The following section reveals some other approaches that have this desirable property of avoiding any intrusive distinction between 'programming' and 'use'.

3.4.3 Opportunistic formalisation

The forms of opportunistic delegation described in section 3.4.1 provide a means for the user to describe what is wanted using some form of parts kit that, because it can be assembled in many ways, allows flexible control over a subtle and rich variety of facilities. But the kit itself remains constant; all the user can vary is the way the pieces are assembled, and the data they are used to process.

A complementary approach is to provide facilities that effectively allow the user to augment the kit of parts, providing new formal components to assist in the processing. A hard (unfriendly to end users) way to do this is by the use of low-level programming techniques, such as the C++ additions sometimes needed for ACE. An easier means, that is available in some domains, could be termed *opportunistic formalisation.*

The key to opportunistic formalisation is the distinction that is normally drawn between the processing facilities and the data on which they act. Most systems' processing facilities have fixed limits as to the details of the data they can recognise and process—for example, a word processor tool for automatically numbering document sections would not normally pay any regard to the wording of the section titles. But in some cases the user may wish aspects of the data to be used to assist the processing. For example, in the Lotus Agenda PIM most of the text of the individual items is stored 'as is' without being analysed, but if the user decides to create a new named note category, e.g., 'Tennis', any note in which the category name appears can be coerced automatically to become a member of the category. A more refined and controllable generalisation of this procedure is *incremental formalisation,* as reported by Shipman and McCall (1994). Their Hyper-Object Substrate (HOS) system is designed to help users handle large bodies of textual information that is entered in raw (informal) form—typically because the users don't yet know what useful formalisable properties it contains—and can later be transformed by user-driven identification of properties that are felt to be useful to formalise:

> 'As formalization proceeds, progressively more of the following are identified: 1) 'chunks', i.e., meaningful units, 2) relevant attributes, values and relationships, 3) associations between chunks, attributes, values and relationships, 4) constraints, 5) generalizations (both rules and inheritance relationships), and 6) aggregate constructs.'

Users are not compelled to pursue formalisation to any particular degree, or within any time-frame—but the more they do, the more they can use the information-structuring facilities provided within HOS. An example cited by Shipman and McCall is the Interactive

Neuroscience Notebook (INN) developed as a class project, which allows student neuroscientists to collect and organise their evolving information on the subject of neuroscience.

Another interesting example of opportunistic formalisation is Mander, Salomon and Wong's (1992) system for supporting casual organisation of information using a 'pile' metaphor. The authors had noted the way people tend to create piles of papers as a way of temporarily organising them prior to filing them away permanently, and decided to implement a prototype system to explore creation and manipulation of 'piles' of multimedia documents. Piles could be created, shuffled and merged without needing to have a formal structure—indeed a dishevelled appearance and *ad hoc* ordering of piles often helped users remember their contents. But the system also provided facilities for formally assisting with certain kinds of operations—for example, a pile could have an attached 'script' determining the criteria that would make a given subset of the documents eligible for membership of the pile. The user could therefore induce novel system capabilities by deciding to attach additional formal characteristics and criteria to particular documents and/or piles.

3.5 Conclusions

This chapter has reviewed various exploration strategies, demonstrating how they are each suited to domains in which assumptions are made that would not be valid, in general, for an opportunistic exploration. A user must be able to delegate all three major sub-activities of exploration in order to obtain a thorough view of a result space without excessive burden, and in the case of an opportunistic exploration this implies that all three activities will have to be delegated while the exploration is in progress. The goal of this thesis has thus been set at finding a means for all this opportunistic delegation to be expressed.

Chapter 4

The Reconnaissance approach

4.1 Introduction

Reconnaissance is a powerful technique for applying resources to an exploration in which the explorer doesn't know at the outset what might be found, but always has some idea of where might be worth examining next and what to note in this examination. By delegating small steps of progress and observation to nimble scouts who return with summary reports, the explorer is able to make crucial comparisons between the various available directions. The decision on where to direct the main effort can be made on the basis of these comparisons.

Definition (from the Concise Oxford Dictionary, Eighth edition):

> **reconnaissance**
>
> 1. a survey of a region, especially a military examination to locate an enemy or ascertain strategic features.
> 2. a preliminary survey or inspection

One aspect of reconnaissance that helps make it resource-efficient is that the observations can be made without the scouts having to know how the information contributes to the overall task; indeed, they don't need any strategic understanding of the exploration. Therefore the concept of computer-based reconnaissance, introduced by this thesis, puts the computer in the role of the scouts—being instructed by the user to visit locations in a result space and to report back some measurements of the results that are found.

Definition (for the purposes of this thesis):

(result-space) reconnaissance

- a survey, consisting of measurements taken from the results that arise in a region of a result space, used to ascertain features of the result space

The basic concept of result-space reconnaissance is first illustrated by revisiting the flight-booking scenario used in chapter 1. Then section 4.3 fleshes out a non-computing scenario to illustrate how reconnaissance in general contributes to the pursuit of thorough exploration, and section 4.4 specifies some characteristics that make an exploration domain amenable to the use of reconnaissance. The final sections return to the matter of using reconnaissance in a computer-based exploration, outlining the challenges that must be addressed in implementing that support.

4.2 The travel agent revisited

This section illustrates the potential impact of a form of reconnaissance on the scenario described in the first chapter—first by showing a typical sequence of illumination–evaluation–consolidation cycles that might occur with the existing level of support, and then by showing how a reconnaissance-based illumination might help.

4.2.1 Without reconnaissance

Without reconnaissance the search consists of a trial-and-error sequence of queries to the database, evaluation of what has been found, and consolidation decisions based on how the search is proceeding. The travel agent and I pursued steps much like the following:

1. Query: **non-stop** flights, **Glasgow→LAX**, out on **Oct 31st**, staying **7 days**.

   Result: The only available seats on non-stop flights are full-fare (lowest is approx. £600 return). By selecting a seven-day visit we were hoping to find deals for about half that price.

2. Query: as previous, except travel on **Nov 1st**

   Result: Again the only available seats are full-fare, but the lowest is much cheaper (approx. £480). We see that the reduction is due to the scheduled decrease in fares on transition to the quieter November period.

   Decide: DISCARD PREFERENCE FOR OCTOBER FLIGHTS

3. Query: as previous, except travel on **Nov 2nd**

   Result: No improvement over November 1st.

   Decide: RESPECT WEAK PREFERENCE FOR NOVEMBER 1ST; DO NOT INSIST ON NON-STOP FLIGHTS

4. Query: **allow stopovers**, revert to **November 1st**

   Result: Many more flights available–

   - cheapest is £340 but the outbound journey has two stopovers, takes a long time overall (21 hours) and arrives too late (11 p.m.)
   - another at £380 departs 6 a.m., one stopover of 4 hours

   Some routes have no seats only because the segment from Glasgow to the first stop is full. Many of these are via London, where I have family so I wouldn't mind finding my own way there. So...

   Decide: TRY SEPARATE NON-STOP FLIGHT FROM LONDON; REMEMBER TO CHECK ARRIVAL TIME AND ELAPSED TIME

5. Query: **non-stop** flights, **London (Heathrow/Gatwick)→LAX**, **Nov 1st**

   Result: Only business class available, because return flights are full.
   But some airlines will allow return from a different nearby city, so...

   Decide: TRY RETURN FROM SAN FRANCISCO

6. Query: ...

At this point in the search we are looking for flights out of London, but returning from San Francisco. Since step 4 we have only been requesting a single outbound travel date, even though it is only weakly preferred to the day after—and is actually *less* desirable than my original hope of 31st October.

Now that we are finding that November flights are rather full, we can see that the compromises we might have to accept may make up for the differences in pricing. Indeed, we may have more hope of finding remaining seats on October flights exactly because the price changes are putting off most people. So we ought to run London queries for all three acceptable departure dates.

But the idea of finding additional possibilities by coming back from San Francisco might apply to Glasgow flights, too: another set of queries to run.

The cost of running lots of different queries

As well as the time taken to specify the query and for it to be processed, we need to consider the effort involved in evaluating the results. Imagine a system that produced the following style of information for each query[1]:

```
Depart  From  To   Return  From  To   Stops
31 Oct  GLA   LAX  07 Nov  LAX   GLA  <= 1     The query

1:  min. fare  UKL 595                         Lowest round-trip fare available

    Flight   Leave      Arrive    Equip
    UY4567  GLA  650A  LHR  810A  757          1st leg of outbound journey
    BB 987       1010A LAX  150P  747          2nd (final) leg

    BB 988  LAX  530P  LHR 1230P+ 747          1st leg of return journey
    UY4568       130P  GLA  250P  757

2:  min. fare  UKL 631                         Next available fare

    Flight   Leave      Arrive    Equip
    FU  53  GLA  100P  LAX  340P  767          Direct outbound flight

    FU 812  LAX  230P  YYZ  650P  L10          2-leg return
    FU  28       810P  GLA  925A+ 767

3:  min. fare  UKL 635

etc...
```

This display shows compatible combinations of outward and return journeys. For each pair, the system has found the lowest fare that can still be booked on those flights[2]. The flights are listed in ascending order of price, which is assumed to be an important factor.

[1]The information available, and the way it is requested and displayed, differ depending on the service and front-end being used. In this example the flight details and the format of their presentation are make-believe but plausible, although I have not yet seen a system that performs pairing of outbound and return flights. For busy routes the same seat price may be available on many alternative pairs of flights run by a given airline on the same dates, so the system might allow the user to customise whether all such combinations are shown separately, or bundled and summarised.

[2]There may be fares that are lower, but for which the traveller does not qualify. This scenario assumes that the user can separately specify the relevant criteria—e.g., is a student; 'stand-by' not acceptable.

In this case the cheapest fare is £595, for a journey that includes a single stopover in each direction. The next fare in this list is about £35 more expensive, but has the attraction of a direct outbound flight.

Some of the factors that the travel agent and I decided were important can be read directly from this presentation—such as the departure and arrival times. But it does take time and effort to extract this information, and even more so for derived information such as total duration of the journey. If the trade-offs are important to us, we may feel we need to look through and compare many results from each query.

Trial-and-error selection of queries

As shown by this example, there can be a great cost in time and mental effort—part of this effort being the decisions on which queries to run and which not to run. These decisions are made on the basis of a developing feel for where the worthwhile results are likely to be—but with a discrete-valued quantity such as seat availability such 'feel' is (with all due respect for travel-agents' experience) never more than a guess. For example, in the above scenario we have requested a trip of exactly seven days, even though the cheap fares usually apply over a week *or more*. Since the flights are sometimes only full in one direction, as we found in step 5, *perhaps* the return flight on the following day would be available instead. Without running the query we will never know, but just how many of these queries are we prepared to submit and examine?

Because we have to make each query and analyse each result separately, the economics of time and effort strongly discourage thorough searching. In picking a path through the result space we therefore become highly selective, basing our decisions on the results that we happen to find. But this evidence is just a form of sampling, and in this case cannot be extrapolated with confidence: flights being full on two consecutive days does not remove the chance of a free seat on the third day.

As illustrated in the following section, the provision of reconnaissance gives a clear and dramatic reduction in the effort required to carry out extensive sampling; in addition, its assistance in comparing the results should give the user increased confidence in the accuracy of judgements made. Informally this suggests that when reconnaissance is available a decision-maker will select a more thorough search strategy than without, and will make search decisions that are correspondingly better informed.

4.2.2 With reconnaissance

The essence of computer-based reconnaissance, as introduced at the start of this chapter, is to delegate to the computer the evaluation of all results in a specified region of the result space, requesting a summary of the results in terms of measurements that will assist in their comparison.

The above scenario has reached a stage in which the travel agent and I are willing to consider flights that leave on any of three dates, from either Glasgow or London, and that return after seven or eight days from either Los Angeles or San Francisco. We have also noted that the 'measurements' of interest for comparing flights include the time and duration of the journey, as well as the cost.

Requesting a reconnaissance foray

This is an ideal situation in which to request reconnaissance: simply instruct the computer to perform queries using all combinations of these alternatives, and to report the various measurements of all corresponding results. So we specify a reconnaissance foray covering the following query parameters:

parameter	date	from city	destination	stay	return from	max. stops
alternatives	31 Oct 1 Nov 2 Nov	Glasgow London	Los Angeles	7 days 8 days	Los Angeles San Francisco	1 (each way)

The total number of queries we would have to request if we played out all these combinations by hand is $3 * 2 * 2 * 2 = 24$.

Then we need to specify the journey properties that we want the system to extract from the textual query results and present in the summarised display. Some of them, such as the dates of outbound and return travel, will have been forced by the query specification; others are the details that we previously had to read by eye from each result. The combined set of properties is as follows[3]:

property	out	from	return	from	cost/£	leave at	arrive at	length	stops
example	1 Nov	GLA	9 Nov	SFO	565	11:15	17:30	14:15	1

[3]To keep the example manageable, the times and journey length will only be displayed for the *outbound* portion of the journey.

Having specified the queries and the result properties of interest, we can 'sit back and relax' while the system submits the queries and aggregates all their results. If we do wait for all the results to appear, nominally we may expect 24 times the normal length of delay; in practice, caching of results and the ability to run many of the queries in parallel may reduce the time needed. Additionally, the system might let us start to look at results as soon as the earliest ones arrive.

Analysing the results

The next stage of progress is for us to compare the results and decide which represents the best combination of properties—or, equivalently, the least undesirable compromises. The search has taken on the character of a multi-variate exploratory data analysis, and can be supported by some of the many techniques that have been developed for computer-assisted graphical display of multi-variate data.

	date	from	return	rfrom	cost	leave	arrive	length	stops
1	2 Nov	LON	10 Nov	LAX	280	15:55	18:55	11:00	0
2	2 Nov	LON	10 Nov	SFO	312	12:30	15:30	11:00	0
3	1 Nov	GLA	9 Nov	LAX	320	13:20	19:45	14:25	1
4	1 Nov	GLA	8 Nov	SFO	320	13:20	19:45	14:25	1
5	2 Nov	GLA	9 Nov	SFO	320	13:20	19:45	14:25	1
6	2 Nov	GLA	9 Nov	LAX	320	13:20	19:45	14:25	1
7	2 Nov	LON	10 Nov	LAX	325	13:30	19:35	14:05	1
8	1 Nov	LON	8 Nov	LAX	325	13:30	19:35	14:05	1
9	1 Nov	LON	8 Nov	SFO	325	13:30	19:35	14:05	1
10	2 Nov	LON	9 Nov	SFO	325	13:30	19:35	14:05	1
11	1 Nov	GLA	9 Nov	LAX	330	12:15	18:55	14:40	1
12	2 Nov	LON	9 Nov	LAX	335	16:55	00:40	15:45	0
13	1 Nov	LON	8 Nov	SFO	338	13:45	17:30	11:45	0
14	1 Nov	LON	8 Nov	SFO	341	13:45	17:30	11:45	0
15	2 Nov	LON	9 Nov	LAX	341	13:45	17:30	11:45	0
16	1 Nov	LON	9 Nov	LAX	341	13:45	17:30	11:45	0
17	1 Nov	GLA	9 Nov	SFO	350	13:20	15:40	10:20	0
18	1 Nov	GLA	9 Nov	LAX	362	13:20	22:58	17:38	1
19	2 Nov	GLA	10 Nov	LAX	362	13:20	22:58	17:38	1
20	1 Nov	GLA	8 Nov	LAX	362	08:15	15:30	15:15	1
21	2 Nov	GLA	10 Nov	LAX	362	08:15	15:30	15:15	1
22	2 Nov	GLA	10 Nov	SFO	388	11:00	17:30	14:30	1
23	1 Nov	GLA	9 Nov	LAX	388	11:00	17:30	14:30	1
24	31 Oct	GLA	8 Nov	LAX	395	13:20	19:45	14:25	1

Figure 4.1 (Part of) a table of flight results, sorted by cost

Probably the most straightforward kind of display is a table. A portion of the flight search results presented in a table may appear as in figure 4.1. Even a static tabular display affords some forms of analysis, such as rapid scanning to find the cheapest or quickest journey (provided that the table is of reasonable size). But when the table is produced in an interactive system, its effectiveness can be increased dramatically by capabilities such as re-sorting based on the values in various columns.

But for representing multi-dimensional data the field of Exploratory Data Analysis has developed many alternatives to the table, each providing advantages for particular forms of analysis. As will be discussed in chapter 6, one representation that has become very popular in recent years is *parallel coordinates*. The parallel coordinates technique presents data using axes that, instead of being arranged orthogonally in a notional hyperspace of as many dimensions as there are axes, are laid out *in parallel* on the 2-D plane. A data point is represented by a polyline that runs from one end of the array of axes to the other, intersecting each axis at the place determined by the data point's value for that variable.

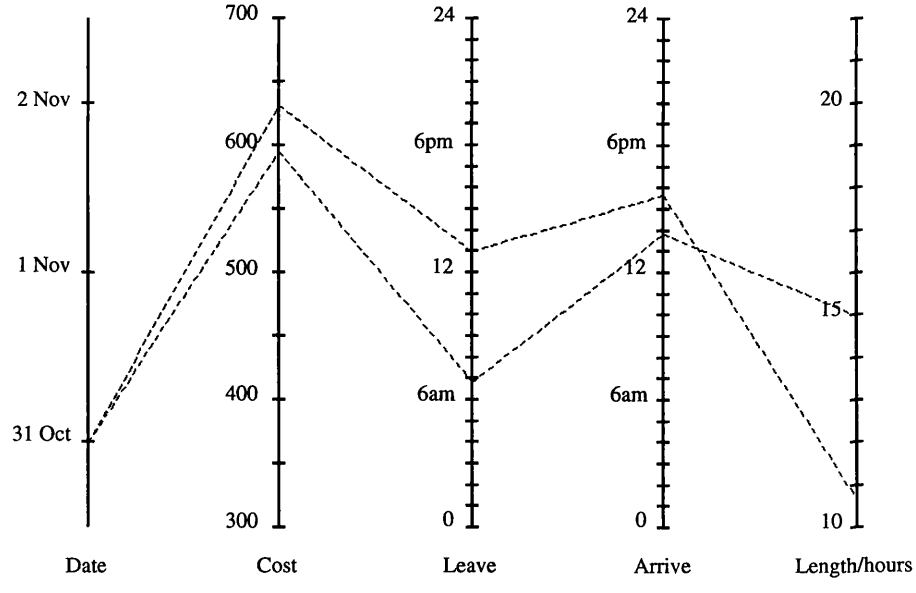


Figure 4.2 Parallel-coordinates representation of two flights

Figure 4.2 shows a representation of some properties of the two outbound flights shown in the result list on page 78. The display makes it easy to see some of the trade-offs involved: that the less expensive flight leaves much earlier than the other, but because of the longer journey it arrives only a little ahead.

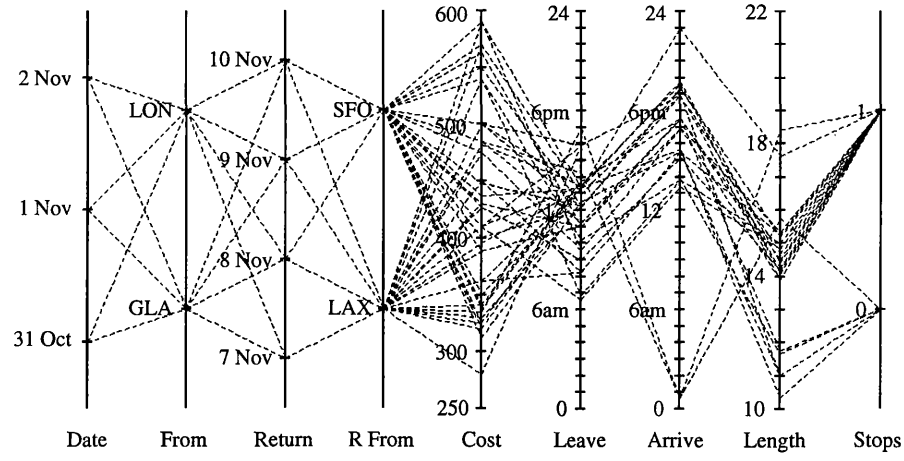


Figure 4.3 Reconnaissance results showing 58 flights with *cost* $< £600$

When the full results of our reconnaissance are displayed on parallel coordinates, it becomes much harder—in many cases, impossible—to follow any single item's plot through all the axes. Figure 4.3 shows just the flights on which seats are available for below £600, of which there are still 58. But even this busy display makes salient some useful information[4]:

- the results seem to fall roughly into four bands of cost, with a promisingly well-populated band below the £400 mark;
- flights in the lowest cost band all leave in the afternoon, and appear to be quite evenly spread between returning from San Francisco or Los Angeles;
- overall journey times vary little, other than the expected delay introduced by additional stops, although...
- ... there are a few conspicuous outliers, such as the allegedly non-stop flight that takes nearly 16 hours.

As with tabular presentations, an interactive support environment can enhance the power of the display. For example, one may be able to re-order the axes to highlight useful correlations, and to filter the results by cutting out those that pass through particular parts of the display. Figure 4.4 shows the state of the display at a moment when its user has re-ordered the axes, and has also blocked from view all journeys that return via San Francisco,

[4]Please note that although the sample data are intended to be plausible they are guaranteed *not* to represent any existing pricing structure.

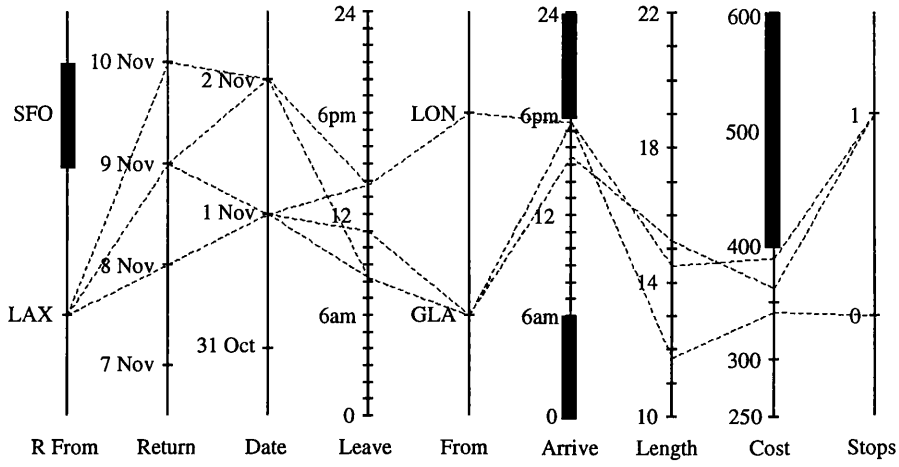


Figure 4.4 Axes re-ordered and filtering applied (5 flights)

or cost more than £400, or arrive outside the hours of 6 a.m.–6 p.m. Of the five results that escape these restrictions, it is immediately apparent that:

- no flight is available on 31st October, but on both the other dates there is the choice of flying from Glasgow or London
- the flights leaving Glasgow are all before noon, whereas those from London are all (both) at about 2 p.m.
- the cheapest flight is also the quickest

Since the results only show a summary, the user is likely to want to examine in detail the ones that appear to be interesting. This inspection might reveal further flight properties that are relevant to the exploration, but which may or may not be expressible on measurement scales. In fact there are various categories of property:

- Measurable and worth constraining: e.g., the cost, as used in figure 4.4 to filter the results.
- Measurable but not worth constraining: e.g., the number of remaining seats (as long as it is non-zero).
- Not expressible as a measurement: e.g., the movies to be shown.

4.2.3 What has reconnaissance provided?

- **Illumination**

  The first benefit of reconnaissance in this scenario is in letting the user be certain that, *within the specified range of criteria,* all possible results have been found. For example, unless the user relaxes some more constraints there is definitely no way to obtain a flight for less than £280, nor to arrive in Los Angeles in mid-morning.

- **Evaluation**

  When it comes to evaluating the results that fall within the range of illumination, the collection of summary information about all results helps the user to focus in detail on just the ones that appear most promising. For example, the user can see that setting a price constraint of £400 will retain a reasonable sample of candidates worthy of more detailed examination—but that if this batch contains no good results the £400–450 range contains a further cluster that can be considered as a fall-back option.

- **Consolidation**

  The wide evaluation perspective gives the user some strong guidelines for deciding on how to proceed with the search. For example, if a flight is found that is satisfactory in all major criteria, and costs £330, the user may take note of the price distribution and decide not to pursue the search further since any potential price advantage can only be marginal.

4.2.4 An important note to users of query services

If the reader is a subscriber to an online flight enquiry service, or the equivalent in some other domain, please **do not** *implement your own naive form of reconnaissance without consulting the service provider! No service or network can sustain an order-of-magnitude leap in query submissions, nor is this necessary. Large commercial databases support highly efficient resolution of the required form of query—essentially* $(a\ OR\ b)\ AND\ (c\ OR\ d)$*—so the 24 alternatives suggested in the example above could usually be expressed in a single request. The results would be received in about the same overall turnaround time as for a fully-constrained query. You just need to agree a suitable query syntax with the provider.*

4.3 Reconnaissance encourages well-informed exploration

Having shown the impact of one possible form of reconnaissance applied to a particular scenario, I now wish to explain in more general terms how the reconnaissance concept constitutes a promising approach to supporting thorough exploration.

For illustration I will consider the case of some people shipwrecked on an uninhabited shore—who have landed with a large cargo of home-making possessions but need to decide where would be a good place to set them up.

4.3.1 Illumination

The castaways need to find out about their surroundings. Rather than proceeding to explore in a single group, taking all their possessions with them, it is more efficient if they can divide into small scouting parties that set off in different directions, eventually returning to a central point to report their findings. The scouts can travel light, simply needing to observe the terrain for features that will help decide where to strike a permanent camp. Separate scouting parties can explore independently to cover many directions at the same time.

At first it may not be clear which directions merit detailed exploration, so there may be preliminary forays in which the scouts are merely asked to report the overall lay of the land—e.g., existence of hills, rivers, forests—so that on subsequent missions they have a better idea of what is available and what factors need to be resolved (e.g., how close to a river they can find the shelter of a forest).

With a limited number of people and a limited amount of time it is not practical to attempt a detailed survey of a large area, but by accumulating centrally the information from successive explorations the group can attain an impression of which areas hold the most promise. It is still possible that some good locations will fall between or beyond the explored areas, and therefore fail to be noticed, but in an unpredictable territory reconnaissance at least makes it practical to search over a wide area and thus reduce the danger of being attracted by local highlights that are not impressive in the overall scheme.

4.3.2 Evaluation

Although each scouting mission may reveal information that is of value in its own right, it is the centralised collation of all the reports that enables the powerful feedback-driven

expansion of knowledge about the area as a whole. As well as an emergent impression of the overall layout, reconnaissance supports the opportunistic discovery of criteria by which to compare findings, and the available ranges of performance on these criteria. This is crucial for the above-mentioned need to gauge each superficially attractive finding against the overall context, and to instruct scouting parties on what to record from their future missions. For example, the settlers may discover that there are different kinds of nearby forest, some affording better shelter or easier availability of building materials than others.

One of the unpredictable features of reconnaissance-based exploration is that when a criterion like this is discovered to be important it may or may not be possible to deduce from the earlier reports how they score on this measure: the earlier scouting parties were not aware of the existence of the comparative criterion, but they might have recorded (or remembered) enough information to fill in the relevant detail. In some cases scouts will have to re-trace their previous missions to pick up the additional information, although this can obviously be limited to the missions that were at least broadly valuable and in which—in this example—there was a sighting of any forested land at all.

Consistency in reporting is crucial to obtaining maximum value from the centralised collection of scouts' reports, since it enables detailed comparison of all the results. The decision makers can notice and weigh up the trade-offs between the sites that have been found, repeatedly re-evaluating them with alternative priorities in mind but without having to re-visit them.

But the reconnaissance reports are, after all, only summary analyses of the sites. For a full evaluation of any site the whole group of decision makers may need to visit it, to see its special features that nobody had thought to request from the scouting parties, or that could not be summarised in a report (e.g., aesthetic qualities of the surroundings). The point of having the reconnaissance is that this full-scale kind of evaluation will only be carried out on the sites for which the comparatively cheap scouting missions reported favourable combinations of criteria and trade-offs.

4.3.3 Consolidation

The use of reconnaissance makes a dramatic difference to the method by which the decision-making group as a whole makes progress in its decision. Many search techniques commit the searcher in a way that would be analogous to moving the whole landing-party, possessions and all, each time a favourable or unfavourable location is discovered. By contrast, when using reconnaissance the group can make 'progress' simply in terms of adjusting the working

set of directions in which scouts are dispatched, and the level of detail they are asked to gather before returning.

The Dynamic Query systems reviewed in chapter 3 also offer the user the chance to evaluate a range of directions before making a heavy commitment to just one. But DQ can only be used for domains in which the directions can be illuminated instantaneously, allowing the user to move so rapidly between them that it is possible to keep in mind all the relevant differences. Reconnaissance gets around this need for instantaneous illumination: after the one-time effort and delay of instructing and waiting for the scouts, measurements from all the explored regions are brought together and can be viewed simultaneously no matter how far apart they originated. Reconnaissance can thus be seen as enabling instantaneous trade-off analysis for domains in which each result may take a significant length of time to generate. It is an issue in the management of the reconnaissance to ensure that the report-assembly delays are not so long that they outweigh this benefit.

4.3.4 Summary: benefits of reconnaissance

The use of reconnaissance encourages thorough search because it allows efficient use of the resources at the decision maker's disposal. Rather than examining in full detail some haphazard subset of the search space, the resources can be applied incrementally in acquiring a context-specific understanding and evaluation of a wide range of available results. The incremental nature allows the decision maker to start with minimal prior knowledge of the layout or comparison criteria of the result domain.

Efficiency is also encouraged by the independence of the scouts once given their mission instructions. This allows multiple forays to happen in parallel, while the decision maker attends to other tasks or starts to evaluate the reports from the scouting missions that return earliest.

The key to the benefits of reconnaissance is that information about disparate parts of the search space is brought together before the decision maker, by scouts who do not need an understanding of the decision as a whole but only the ability to follow simple instructions.

In terms of the effort–accuracy trade-off, reconnaissance clearly ought to encourage greater thoroughness: the user is spared some of the main sources of wasted effort (repetitive result requests, and the full-scale evaluation of results that turn out to be completely unsuitable), and has a cheap way to obtain a well informed view of the decision space and hence to make sensible judgements about decision accuracy.

4.4 What kind of exploration space can be reconnoitred?

Not all explorations are amenable to the reconnaissance approach. In this section are described a set of constraints on the exploration task that will enable the explorer to deploy reconnaissance. Examples are given showing how these constraints may arise in computer-supported explorations.

Note that it is not claimed that these are *minimal* conditions required for implementing any kind of reconnaissance—merely that a domain must satisfy these constraints to support the straightforward form of reconnaissance demonstrated in this thesis.

4.4.1 Constraints on the result space

- **Results must be consistent over time.**

  When a scout provides a report of the result found in some location, that report must be reliable for as long as the decision maker may want to act upon it—i.e., until the result is permanently discarded from consideration. Up to that point scouts may be sent back to the location to gather further information about it; for my model of reconnaissance-supported decision-making it is important that the latter information always builds on what was known before, and never contradicts it.

  Certainly, reconnaissance *could* be provided in dynamic domains, such as live plotting of various alternative futures in Air Traffic Control, but the need for timely refreshing of the reconnaissance result presentation introduces issues beyond the scope of this study.

- **Reconnaissance must not cause undesirable side effects.**

  The decision maker should not feel restricted in the use of reconnaissance, so it is important that the reconnaissance itself does not have undesirable side effects. In particular, and related to the previous point, the running of scouting missions should not in itself alter the result space in unpredictable ways.

  This constraint has obvious practical illustrations: for example, you can't run ATC reconnaissance by trying one set of flight plans then telling all the pilots to back up and start again.

- **Result space must be amenable to sampling.**

  If the results obtainable by reconnaissance apply only to pin-point locations within the search space, and give no clue as to the properties of nearby alternatives, even

reconnaissance cannot perform an effective job of 'illuminating' the search. This would be the case, for example, in an ill-conceived flight search database that required the specification of a precise flight time (e.g., 9.14 a.m.) and only responded with flights departing at exactly that time: even knowing the flight information for 1,000 different minutes of a particular day would not tell you which flights you had failed to find.

When sampling a domain using an effectively continuous parameter, such as a coefficient in a financial model, the sampling constraint becomes one of requiring the model to have regions that are well behaved, rather than being entirely stochastic. Then, for example, if some measurement is found to have changed sign between two points the decision maker may reasonably expect that a more detailed search of the region will locate an informative crossover or discontinuity.

- **Results must be amenable to non-expert summarising.**

  There must be aspects of the results that can be measured by scouts, and that will help the decision maker to evaluate them—rather than a domain such as that of Mutator, in which the results can only be judged by full examination by the decision maker.

- **Summaries must be amenable to collation.**

  Since there may be many scout reports to evaluate, the decision maker needs to be able to organise them for efficient reference and comparison. This strongly favours some form of coded symbolic report summary; consider, by contrast, the effort that would be required in judging trade-offs between reports that were only available as recorded speech.

  In some cases analysis will only be practicable if the reports can be clustered into categories based on their measurements. But the decision maker does not need to define in advance the borderlines that will delimit useful categories; like the other aspects of understanding the search space, categories can be deduced as the information is accumulated.

- **Unseen areas must be predictable enough to 'bootstrap' the exploration.**

  The decision maker must be able to start with at least enough knowledge of the nature of the search space and its results to instruct the first scouting missions; thereafter, the scouts' reports and occasional full result examination will furnish the bulk of knowledge about the domain.

  A simple example of provision for this initial level of knowledge in a computer domain is the use of command menus.

4.4.2 Constraints on scout deployment

- **Scouts must be capable of independent operation.**

  The scouts must be able to follow instructions from the decision maker to visit a specified location in the domain and to bring back a report containing specified measurements of the result found there.

  It is important to recognise the level of instruction that is involved here. In particular, it is a crucial property of the reconnaissance approach that the decision maker does not need to explain to the scouts *why* the information has been requested. In this sense reconnaissance takes the 'cooperative' approach to the *delegation problem,* as described by Fischer, Lemke, Mastaglio and Morch (1991):

  > 'Automating a task or delegating it to another person requires that the task be precisely described. Most tasks involve many background assumptions that delegators are incapable of describing. The cooperative approach eliminates the need to perfectly specify tasks. Instead, the cooperating agents incrementally evolve an understanding of the task.'

  Although I would not claim that a computer-based reconnaissance system constitutes a party capable of 'understanding' a task, it is the case that the information assembled from the scouting missions is what forms the basis of the decision maker's evolving understanding of the decision space. The fact that the scouts can *only* follow simple instructions fits perfectly with the decision maker not yet having anything more complex to request.

  So the user of the computer system specifies the range of locations to be visited and the measurements to be taken, without needing to specify which of the locations, or what values of the measurements, are likely to be preferable.

- **Scouts must return within an acceptable delay period.**

  The decision maker must be willing to accept periods of being unable to make any progress, while waiting for scouts to return with information for analysis. Part of this issue is the decision maker's ability and willingness to attend to other activities, so that delays are not experienced as lost time.

  In specifying the range of locations to be explored by scouts the decision maker must decide how large and how finely divided a search is appropriate for the task being addressed, given the time that it will take to gather the results.

  The impact of these delays can be alleviated if scouts' reports can be viewed incrementally, so the earliest ones can be analysed while others are still being generated.

- **Instructed scouts can be called off if necessary.**

  In some cases the most efficient way to instruct the scouting teams is to define blanket coverage of a large region. But if some scouts are able to return earlier than others their reports may reveal that some of the other missions are bound to be uninteresting. Being able to recall the scouts from those missions, and perhaps instruct them immediately for other missions, makes a big difference to efficiency of resource use.

4.5 Can reconnaissance serve a useful role in the domain?

As Norman (1991, pp.19–22) points out, providing additional computer support for some task rarely leaves the nature of the task unchanged. The adaptation of a computer-based exploration to include the increased level of delegation embodied in reconnaissance is not merely an issue of surface-level interaction details, but defines new roles for computer and user. The difference is analogous to the change in approach needed in any pursuit when moving from independent action to the use of an assistant.

This section addresses some requirements and implications of using reconnaissance, that affect its applicability to opportunistic exploration in various domains[5]. The issue at stake is whether the user is able and motivated to specify a reconnaissance foray that would be workable and useful.

4.5.1 Ability to specify reconnaissance

- **Ability to choose a worthwhile and practicable reconnaissance foray**

  In tasks within the given domain, can the user tell what will be a worthwhile range of the result space to illuminate, and what measurements of the results will be worth reporting?

  The specification of reconnaissance involves both the 1) *reconnaissance range*—i.e., a region of the result space in which all the results are worth exploring, and 2) *reconnaissance content*—i.e., a way of summarising the results that are found. It would be understandable if a user were interested in one but not the other—i.e., wanting to view in full detail all the results from a particular range, or to define summarising techniques to be applied to individually chosen results—but reconnaissance explicitly involves both.

[5]Note that there is nothing implicit in the definition of reconnaissance that *restricts* its use to explorations that are opportunistic—it's just that this thesis is only addressing opportunistic explorations, because they are more challenging than those that can be pursued according to a pre-determined strategy.

In an opportunistic exploration, useful range and content will not be known in advance but must be deduced on the basis of early findings. But in some domains there may be standard 'bootstrapping' forays that will usually provide good starting information.

A further issue is whether the kind of reconnaissance the user wants is practicable—for example, not overwhelmed by problems of scale—but that is typically a domain-specific issue, and therefore beyond the scope of this thesis.

- **Ability to express what is wanted**

  Can the designers of support for the given domain provide a convenient formalism for expressing the kinds of range and content that are likely to be of interest?

  Because the reconnaissance approach is by definition a way to derive value from scouts who may not be able to understand the goals of a search, the facilities for guiding computer-based reconnaissance are necessarily of a low level of abstraction. The user just needs a way to express the set of result-space locations to be examined, and the measurements to be taken from the results.

4.5.2 Motivation to specify reconnaissance

- **Motivation to view result summaries**

  In this domain, can the results provide useful summaries for evaluation?

  Although the user may need to view the full detail of a result to make a final decision, reconnaissance might still be useful at the earlier stages when there are large numbers of candidate results under consideration. Obtaining a collated summary of results—for example, as illustrated with the flights shown in figure 4.3 on p.83—can make a dramatic reduction in the effort needed to evaluate and compare them.

- **Motivation to make an investment of effort**

  In the given domain, is the investment of effort in requesting reconnaissance repaid by the benefits of the result presentation that is eventually obtained?

  The use of reconnaissance is an explicit activity that requires the user to think about the exploration at an abstracted level, rather than pursuing it at a moment-by-moment detailed level. Making a reconnaissance request involves cognitive effort in figuring out what to request and how to express it in the system's terminology, as well as the manipulative effort of communicating the request to the computer. While the reconnaissance is being processed the user has to wait (although this delay is alleviated if results can be displayed incrementally as they are generated). Overall, the benefit of the reconnaissance has to outweigh these costs.

4.6 Can a reconnaissance interface be built for the domain?

In this section I describe the characteristics of the components required to provide computer-supported reconnaissance for a given domain.

4.6.1 System model

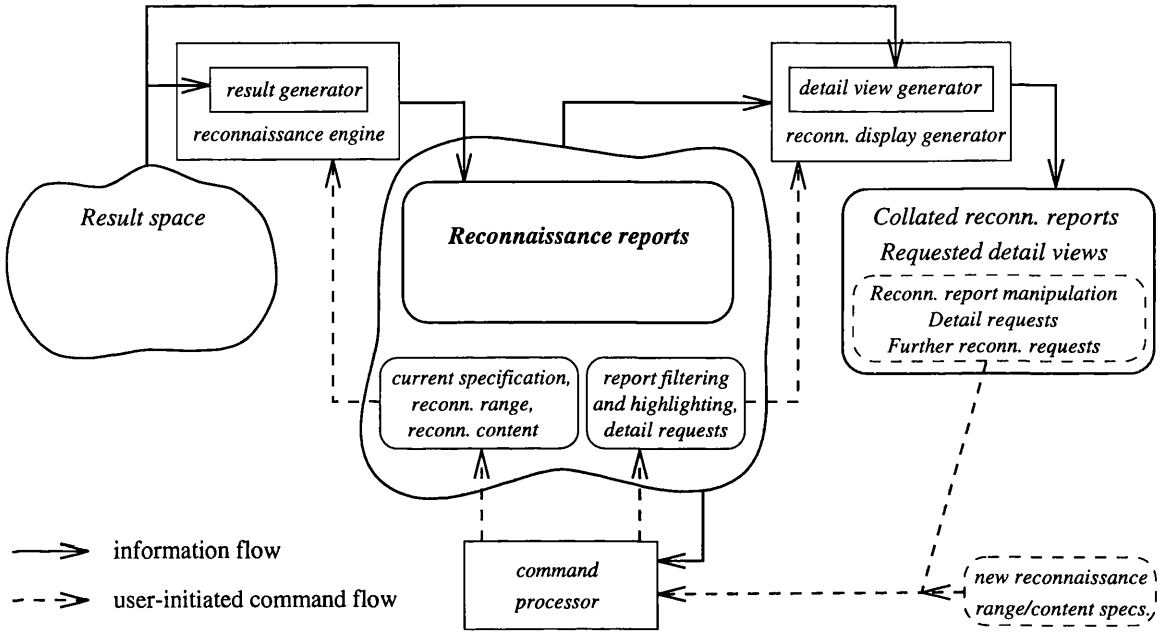


Figure 4.5 Illumination Zone Model for reconnaissance.

Figure 4.5 shows the underlying form of reconnaissance support in terms of the illumination zone model. The main features are:

- **Reconnaissance engine and result generator**

  The reconnaissance engine accepts a *current specification* as a starting point, and calls upon the domain-specific result generator to obtain all the results in a specified *reconnaissance range,* measuring them each according to the *reconnaissance content* specification to produce *reconnaissance reports.*

- **Reconnaissance display generator**

  The display generator controls the construction of the interactive display that allows the user to evaluate and compare all the reconnaissance reports in a single, collated form. It calls upon the domain-specific detail view generator to produce detailed displays of individual results that the user requests to view in full.

- **Command processor**

  Some form of command processor is required to coordinate the user's requests for illumination, reconnaissance-display manipulation and creation of detailed result displays. Because of its central role, the command processor will generally be responsible for maintaining all information within the exploration context, which includes the illumination zone. The illumination zone must be able to hold however many reconnaissance reports the user wishes to evaluate alongside each other.

4.6.2 Illumination

As implied by the travel agent scenario fleshed out in section 4.2, reconnaissance can play the role of adding a higher level of illumination onto an existing computer-based exploration activity. Then, rather than being constrained to the limited illumination-zone characteristics of a system that supports a traditional exploration strategy, a user can collate results sampled from the illumination zone at a wide range of times using the reconnaissance interface as a front end.

This potential separation of functions is signified in figure 4.5 by the separation of the result generator and detail view generator from their higher-level reconnaissance controllers. But depending on the degree of integration of reconnaissance into the domain, they may or may not be separate software components. There are two principal alternative models for support, that I call *reconnaissance shell* and *integrated reconnaissance*:

Reconnaissance shell support

The travel agent example assumes the use of a reconnaissance shell that exists separately from the existing flight-retrieval database. When the user requests flight information that would normally correspond to a large number of separate queries the reconnaissance shell submits each query to the database in turn, collecting textual results that are then parsed and processed to obtain the information the user requested about each flight. Reconnaissance summaries are collated using a parallel coordinates display that has no specialisation to its role of presenting flight information (beyond the designation of axes with appropriate ranges and markings).

The main advantages of implementing reconnaissance as a shell are: 1) the ability to reuse complex components such as the parallel coordinates display, and 2) the ability to adapt existing software systems to provide reconnaissance support, in the ideal case without having to make any changes to the system itself.

A disadvantage that may be overwhelmingly important, as mentioned in section 4.2.4, is that explicit submission of multiple result-generation requests may be extremely inefficient compared with a way in which the full range of results in a reconnaissance foray could be generated by the existing system with just a little adaptation. The existing system may be able to cope with the full complexity of the reconnaissance range specification, or the content specification, or both.

Integrated reconnaissance support

In some domains it may make more sense to integrate reconnaissance fully—indeed, in some it may be impractical to provide reconnaissance support any other way. In essence it was easy to propose a front-end to a flight database because the queries are simple text strings; a range of queries can be created simply by specifying sets of alternative textual terms to be plugged into various parts of the query. But if the reconnaissance were being applied to the design of architectural plans or engineering designs, that are typically built on direct-manipulation drafting systems, text-based specification of reconnaissance ranges would have limited power. Taking measurements is even more likely to need domain-specific support: imagine the limitations of taking measurements from a CAD system by analysing the pixel values of the image displayed on-screen after a design change.

4.6.3 Evaluation

The main evaluation requirement for reconnaissance is a display generator that can show all the reports in the illumination zone in a collated form to support comparison between them. There must also be a means for the user to obtain detailed result views on demand, to check result features that could not be mapped into the summarised display or to find additional features that can be mapped in subsequent forays to provide more informative illumination.

Reconnaissance display format

The need to examine and compare many results at once suggests that a symbolic summary representation is required. In general, this representation may be required to show measurements of multiple independent parameters for each result, and also the location of the result in the search space—typically expressed as a further collection of independent settings. So the representation should be able to handle multiple results that each occupy

a place on multiple dimensions of location and measurement. This may sound like a recipe for information overload, but not if the presentation is designed appropriately; we may take encouragement from Tufte (1990, p.50):

> 'High-information displays are not only an appropriate and proper complement to human capabilities, but also such designs are frequently optimal. If the visual task is contrast, comparison, and choice—as so often it is—then the more relevant information within eyespan, the better.'

The display should be interactive, offering features such as filtering and highlighting of results to help the user analyse the trade-offs they embody. In addition the display should provide interactive facilities for requesting the detailed result views, and for specifying additional reconnaissance forays in result-space regions that are represented on the display but have not yet been illuminated.

4.6.4 Consolidation support

A vital ingredient for opportunistic exploration is the freedom to pursue the search in whatever order turns out to be convenient for the task in hand. As part of this the user must be able to maintain a portfolio of alternative directions of exploration, which is why the illumination zone must be able to support disjoint regions of illumination gathered from multiple reconnaissance forays.

4.6.5 Overall delegation control

The chief challenge in providing a reconnaissance system that users will welcome is to come up with a way of formalising the task such that the requesting of delegation arises naturally alongside facilities by which results are presented and manipulated.

4.7 Conclusions

In chapter 3 I pointed out the problems that arise from a lack of opportunistic delegation in the three key computer-based exploration activities: illumination, evaluation and consolidation. In this chapter I have demonstrated that the concept of *reconnaissance* provides a model for such delegation: illumination assistance is available as opportunistically requested 'scouting missions' to regions of the result space; evaluation is assisted by being

able to request particular measurements of each result, and collating all these 'scout reports' to support comparison and trade-off analysis; consolidation is assisted by the ability to continue gathering and collating reports until the explorer is confident of being able to make a well-informed consolidation commitment.

I have outlined how reconnaissance could be applied to computer-based exploration, and pointed out the main technical challenges involved in implementing it. The next chapter presents my examination of various forms of computer-based task that could make use of reconnaissance-like illumination based on range and content specification.

Chapter 5

Computer-assisted illumination of result spaces

5.1 Introduction

Chapter 4 outlines various characteristics that determine whether a given computer-based activity is amenable to reconnaissance support. The fundamental requirement is that the user be able and motivated to delegate to the computer the illumination of a region of some exploration domain. This chapter presents my early investigations into delegating parts of opportunistic information-handling tasks to a computer; it was this work that led to the realisation of the importance of wide-ranging illumination of a result space, and hence to the idea of reconnaissance.

The unifying concept in the various studies described below is that of harnessing the computer's capability for high-speed generation of alternative results, while giving the user sufficient control over search direction and scope to prevent the presentation of excessive amounts of information to be viewed and analysed. In line with the wish to maintain the user's understanding and feeling of control at all times (as stated in section 2.1), I only considered approaches in which the computer's contributions, and the reasoning by which it was proposing them, would always be explicitly clear to the user. Each approach therefore needed two complementary components: (a) an explicit mechanism for automated generation of candidate results, controlled by (b) an explicit mechanism for the user to guide and filter the result-generation process.

The following three sections of this chapter describe the investigations in terms of separate case studies, addressing the following tasks:

1. Searching for overlap between topic areas in a keyword-based note system.
2. Reformulation-based search for rule combinations that define useful organised views of semi-structured data items.
3. Processing many alternative specifications (as a 'batch') to illuminate the results available in a visualisation based on multiple inter-related options.

For each case I describe the proposed context and scenarios of use, showing how a large-scale user-directed search could be applied. I also explain my judgements relating to the eventual choice of domain on which to test out the thesis claims.

The first study deals with a domain that inherently involves the management of a large result space. The latter two are domains that involve trial-and-error activities for which the running of a range of alternative attempts could help the user to experiment; in terms of the reconnaissance integration models described in section 4.6, these would be cases in which use of a reconnaissance shell might be appropriate.

Section 5.5 then generalises the last of the case studies by identifying a category of exploration domains in which reconnaissance forays can be requested in terms of an *option space* and a *measurement space.* Such domains are attractive as being particularly amenable to the provision of a simple form of reconnaissance support.

5.2 Study 1: Topic-matching guided by concept-map regions

5.2.1 Concept

Figures 5.1 to 5.3 illustrate a scenario in the use of a computer-based assistant for maintaining and using a set of personal notes. This scenario was part of a proposal developed in the early stages of my investigations into how a computer might support semiformal activities.

The system that I proposed to develop, called Custard, was to make use of a concept related to the *mind map* as developed by Buzan (1982). Mind maps were Buzan's proposal for a way of taking notes on paper, that he claimed as both more economical and more memorable than a long-hand, linear style of recording. The key feature of mind maps is the use of compact but highly descriptive terms—typically single keywords—that the note-taker writes and connects with lines to build up a structure representing a set of related

concepts[1]. This structure is supposedly quicker to put on paper than long-hand sentences, and the result is a map that reflects the person's understanding of a topic's key concepts and how they inter-relate—so although it is possible for other people to pick up some information from a mind map, the map is likely to mean a lot more to the original author.

In the Custard proposal it was suggested that a computer-based medium for mind maps could be implemented, in which the computer could perform operations on the keywords used. The scenario elaborated here shows the proposed application of search techniques within Custard, based on literal matching of strings to suggest associations between the various occurrences of any keyword that appears in more than one place[2].

5.2.2 Scenario

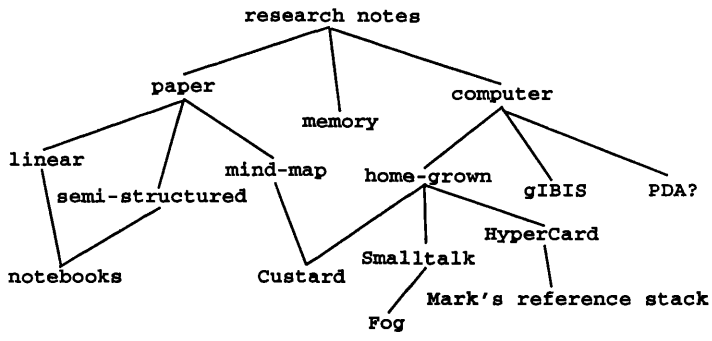


Figure 5.1 A new set of concept connections added to Custard

The context of the scenario is that of a user who has already been using Custard for some time, and is now adding some new notes on the topic of—as it happens—what kind of notebook or computer system to use to store general research-related notes. The new set of notes includes the small region of connected concepts shown in figure 5.1.

Underlying the provision of keyword-matching facilities in Custard is the idea that by highlighting concepts that have already been used elsewhere, and showing the regions of the map space where they are found, the system can remind the user of common ground shared with other topics. This is intended to help the user to realise how the current topic relates to others, and thus engender a more thorough understanding of the issues at stake.

[1]In contrast with the diagrams used here, Buzan suggests a more compact representation in which the terms are written *along* the connecting lines. Also not shown here, but recommended by Buzan to help increase the memorability of each mind map, is the incorporation of different colours, memorable sub-layouts such as lists or other shapes, and small diagrams.

[2]For this simplistic example it is assumed that the map author can be completely consistent in the choice of keywords to describe particular concepts.

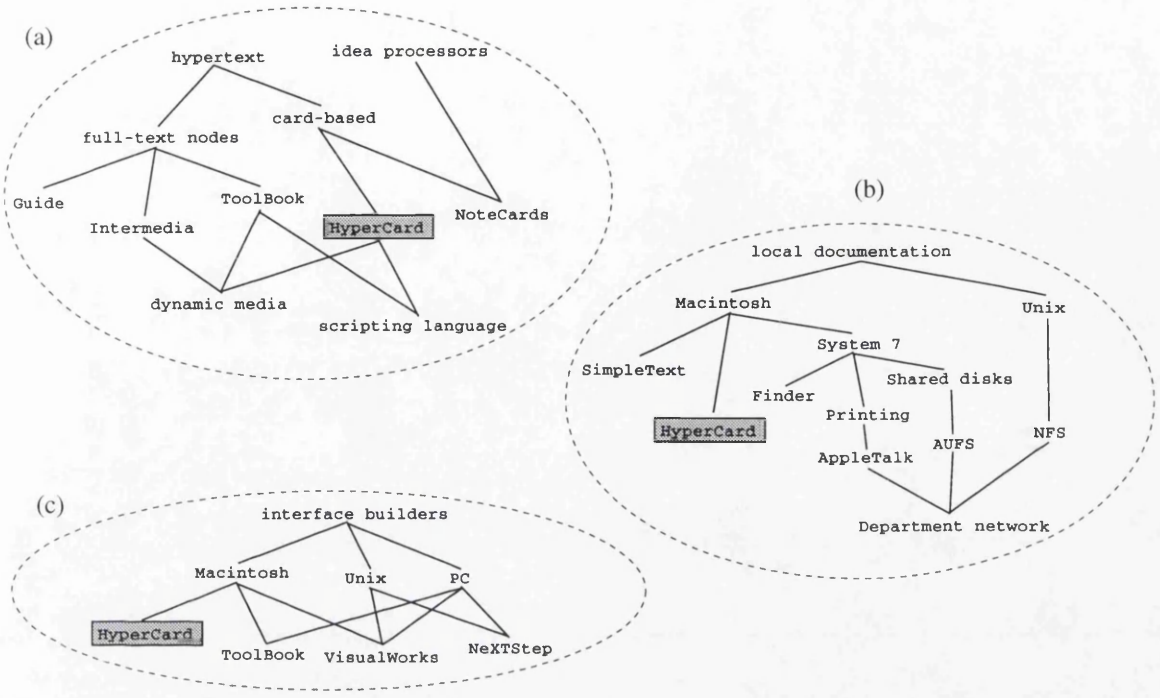


Figure 5.2 Some suggested connections based on one term from figure 5.1

Having entered the new set of connected concepts, the user requests a retrieval of other topics that might relate to them; the system scans the note space for other occurrences of keywords that appear in this 'query' set. Figure 5.2 is a representation of some of the results generated in this case, that were found because of a match with the keyword 'HyperCard'. The figure shows three separate parts of the existing note space: a) some notes on hypertext systems; b) some documentation available within the user's department; c) notes on various interface-construction tools.

The results would not necessarily be presented in isolation, as shown here. Where a match falls at a point within a large map of concepts it may be best to show a form of 'fish-eye' view (e.g., Sarkar and Brown, 1992) of the whole map, focussed on the matching concept(s) but allowing the user dynamic control over the extent of the region that is displayed in detail.

Of these results, the user decides that only the first is actually relevant to the current topic. The result includes a mention of the concept of 'idea processors', that the user decides merits further investigation. Selecting that term in the presented diagram, and specifying that it is of interest, leads Custard to find and display the further region shown in figure 5.3.

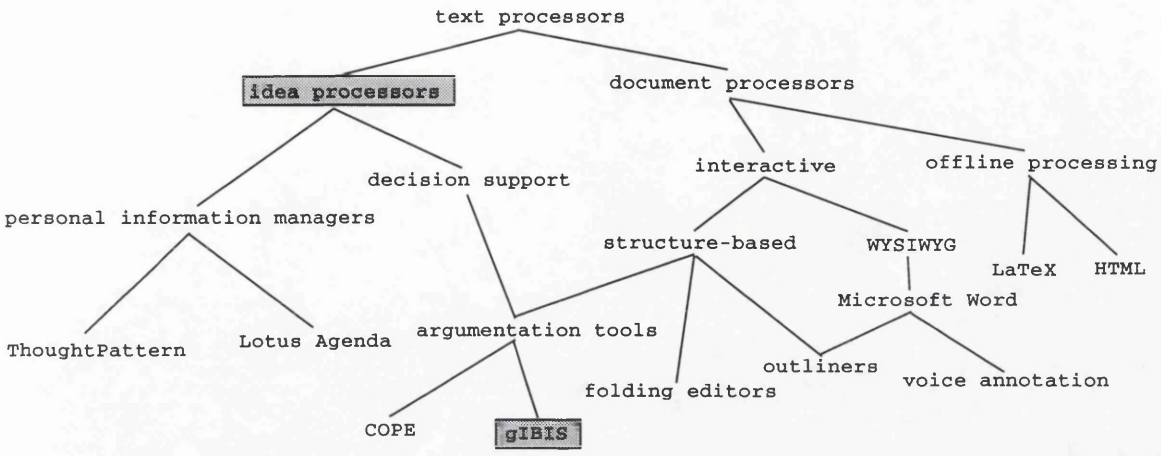


Figure 5.3 A further result, reached after refinement of interests

In this figure Custard highlights two of the terms: the recently selected 'idea processors', and the term 'gIBIS' that appeared in the user's original entry. The occurrence of the 'gIBIS' term implies that this region would have been seen anyway in the first round of results—if the user had continued to look through them. A real implementation would require an efficient strategy for ordering the presentation of results, and for the user to decide when to follow up discovered concepts. This section of the notes relates to various forms of text-manipulation tools, and helps to remind the user that some tools provide facilities for outlining and voice annotation, both of which might be useful with regard to keeping research notes. Custard has served its purpose in guiding the user to an expanded set of concepts that are related to the original ideas.

5.2.3 Applicability of delegated illumination

The aspect of Custard that is illustrated here is its potential for supporting the broadening of a user's ideas on a topic that has not been addressed in the notes before, but has some overlap with topics that were considered and recorded in the past. By doing this, Custard is intended to perform the role of a 'creativity assistant'.

Note, however, that the aim is not to promote 'divergent thinking'—an approach that involves consideration of as broad a range of concepts as possible, emphasising precisely those that are *not* evidently related to the original idea. Techniques based on this approach have been seen as useful ways to induce creative thought (e.g., Adams, 1987), but their effectiveness has been questioned by, for example, Weisberg (1986):

> 'Perhaps the most important assumption supporting the belief in brainstorming is the belief that divergent thinking is crucial in creativity. However, there is much evidence to indicate that this is not correct.'
>
> (p.67)

Weisberg uses the work of Koestler (1964) to describe an alternative approach to the achievement of creative synergy of ideas:

> 'Koestler proposed the term *bisociation* for the process whereby previously unrelated ideas are brought together and combined. Bisociation is contrasted with association by Koestler, since association refers to previously established connections among ideas, while bisociation involves making connections where none existed before.
>
> 'According to Koestler's theory, ideas exist in interrelated sets, or matrices. In normal conscious, associative, thinking, one idea leads to another idea within the same matrix. In situations demanding creative thinking, however, the thinker must move from one matrix to another.'
>
> (p.22)

Correspondingly, the aim of Custard is not to shower the user with concepts picked at random, but to attempt to encourage bisociation between whole topic areas that appear to have some concepts in common.

Requesting illumination

The kind of illumination performed by the system is a search among the body of existing notes, bringing to the user's attention those maps, or sub-regions of larger maps, that appear to deal with some of the concepts that the user has indicated as being of interest. Rather than the statistical methods used in modern information retrieval systems to estimate a 'score' for the relatedness of portions of free-form text, Custard was intended to work only by finding explicit matches between keyword concepts. Any match would indicate a region of potential interest to the user, and the system would not be designed to attempt to differentiate results on the basis of some calculated prediction of likely level of interest.

The automation of the search allows it to be pursued exhaustively, matching against each in turn of the query keywords. In a system containing notes that have accumulated over some time (amounting, perhaps, to tens of thousands of concepts), the search may give rise

to a large number of results. Many results might be based on spurious matches between concepts that appear the same to the system, but aren't, and many others will be valid but of no interest at all to the user on this occasion. A user hoping to benefit from the potential connections that are highlighted therefore needs an effective way to filter the results, cutting out all the ones that are seen to be irrelevant.

My hypothesis was that the structure of the 'mind maps' would provide an appropriate and sufficient generalisation mechanism for filtering the results. For example, when presented with an uninteresting region of an existing map, such as result (c) in figure 5.2, the user can set an explicit range on the map as being of no interest in this search. That will cause the system to filter out of its results, including the results it may generate in subsequent steps of the search, any segment for which the focus falls within the rejected region.

This specification mechanism is a form of *relevance feedback,* similar in some respects to the approach used in the News Retrieval Tool described on page 46. But I felt that the Custard approach, based on explicit rejection of regions of the potential result space, would have an effect that was easier for the user to understand than the statistical term-weighting mechanism employed by NRT, and would be endorsed by users as a mechanism that maintained their confidence in the thoroughness and clarity of the search.

5.2.4 Tractability

Testing out the use of an exhaustive human-controlled search in this domain posed the following challenges:

Supporting consistent, understandable matching

The attraction of the clarity of the computer's actions in performing only precise matches between keyword concepts had to be weighed against the difficulty for the user in maintaining a consistent vocabulary of keywords. One of the benefits cited by Buzan for the mind map technique is the fluency with which ideas can be recorded, since the writer does not need to take the time to form the thoughts into grammatical sentences; this fluency would certainly be compromised if the system required that each concept be chosen to fit into a carefully maintained vocabulary. It is possible, however, that this kind of selection might be acceptable if it were carried out not at the time of the initial generation of the ideas, but in a later 'consolidation' stage. This would fit with Buzan's recommendation that after an author has completed the rapid initial recording of a map of connected ideas, the result

should be transcribed into a neater form of map—this time with the benefit of hindsight to guide the layout, and to allow the author to increase the salience of the key ideas.

Providing obvious and adequate return on investment in using the system

As well as the technical challenge in providing a means to view a large note domain on a standard computer screen, it was clear that making the user examine in detail even a constrained sample of regions of the space might prove overwhelmingly tiring. In terms of the effort–accuracy tradeoff, even with the computer assistance the user would be required to expend a large amount of effort for an uncertain improvement in accuracy; it was apparent that it would be difficult to design experiments in which the accuracy motivation was sufficiently strong for the technique to show its benefits.

5.3 Study 2: Perspectives by reformulation

5.3.1 Context

I observed that there is a broad class of tasks in which a user has a collection of computer-held data items, each possessing a number of formalised properties, and needs to assemble a view of the items that reveals some information about the collection. But it is not always easy to know which view will reveal the information that is wanted—as reported by Rieman, Davies and Roberts (1992) with regard to their task of helping choose the papers to be accepted for a conference:

> '... we generated a bevy of final reports showing the ratings of each paper, sorted and selected every way we could imagine might be useful. Which were the papers that all reviews praised? Which were the ones that no one liked? Which had produced conflicting reviews? Which were borderline? How did the data break down by subcommittee? Were there any subcommittees that were overloaded?
>
> 'It's important to understand that we had only a vague appreciation of these questions during design of the database. Custom reports defined at that time would have missed the mark by a wide margin. During the month of data entry, the subcommittee composition and definitions had shifted, the ratings had raised new questions, and the plan for managing paper selection had become more concrete. But the reports we finally generated were appropriate and, as the committee members later told us, very useful.'

A computer-generated structured view can serve many of the roles that would be served by a physical organisation of the items if they were concrete rather than stored on the computer. The following are some examples of such roles, and some advantages that can accrue from working with a computer-held representation:

- distributing the items into groups whose members usefully 'belong together'

  An example is the common working practice of dividing one's mail into topic-specific folders. On a computer-generated display, one way to present this kind of organisation is using a linear list that is partitioned into sections representing the groups.

  An advantage of working with data held on a computer is that any organisation that can be described in terms of formalised item properties (such as tags identifying the sender, topic and urgency of an e-mail note) can be built dynamically by the computer system. Thus items do not need to reside in a single location, as they would in a physical organisation, but can take their place within any number of alternative organisations that can be assembled quickly on demand. The system could maintain a library of rules or procedures for generating different organisations, and allow the user to switch rapidly between different views—without requiring the effort that is involved in rearranging physical items, and hence reducing the risk of feeling stuck with an imperfect organisation because it would take too much effort to move all the items again.

- supporting simplifications or generalisations about the item collection

  One kind of simplification is to work with a subset of the items—for example, a group that has been arranged to contain just those papers for which some referee reports are still missing. Placing items into some form of ordering can also be a good way to simplify their handling—for example, sorting items by date to make it easy to locate the most recent ones, and to allow generalisations such as 'all items beyond this point are overdue'.

  Again, if the construction of views can be described in formal terms, a computer system can allow the user to see a variety of views on demand. This is the capability that turned out to be so useful once the referee reports had been gathered in the scenario quoted above.

- arranging items so they form a coherent whole

  Examples of this kind of organisation include event schedules (e.g., for a conference or a teaching timetable) that avoid problematic clashes, or the layout of newspaper articles to fill the available space in a convenient way.

> This is a form of task in which computer support can be especially useful: if the desired constraints can be described formally, the computer can perform a search among a large number of alternative possibilities and show only the valid arrangements it discovers.

Note that even where the *construction* of views is mechanisable, their *interpretation* might not be; the system does not necessarily incorporate any formal model for what the user is trying to do with the information on display. So in cases in which it is not known in advance which view will reveal information that is of interest, the user must embark on a search among the available mappings. My aim, in this case study, was to investigate how computer assistance might be provided for reducing the burden of this kind of search.

5.3.2 Scenarios

First domain: Note management

To try a different approach to the domain of computer-supported note management, I examined the support provided in the Agenda personal information manager (PIM)[3].

One of the activities supported explicitly by Agenda is the assignment of individual note items into user-created categories, and the use of these categories to define views containing subsets of the notes. A view is built by specifying the subset of notes it is to include, and the basis for ordering them. The subset is defined by selecting the categories of which an item must or must not be a member in order to be included; the ordering is defined by selecting the categories to be used in sorting or partitioning the view. Items can also be sorted on the basis of properties such as an associated date. So, for example, one could build a 'To Do List' view by requesting all items that are categorised as a Commitment but are not in the category signifying Low on the importance scale; the view could be partitioned into separate regions for items that belong to the Teaching, Research, or Home categories, and within each region the items could be sorted according to the Due Date property so that the most urgent appear at the top.

But what about the kind of view structure that could not be defined in advance, but would require some experimentation? For example, given a set of items referring to articles relevant to my thesis, could I easily explore different ways of selecting and ordering these items in search of a suitable structure for the 'Related Work' section of a paper or report? I implemented a system with some of the capabilities of Agenda, called Fog, on which to

[3] Agenda is a product of Lotus Development Corp., and is described in (Kaplan *et al.*, 1990).

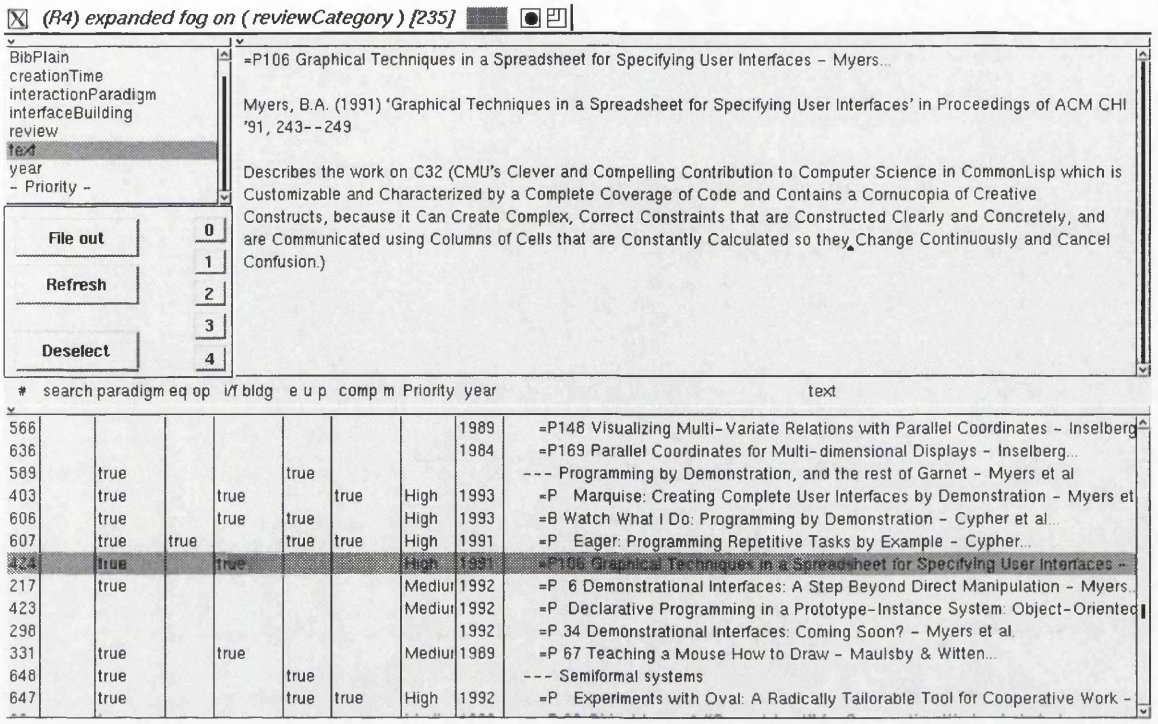


Figure 5.4 A Fog browser, set up for easy item categorisation

perform experiments—and also to serve as the medium for maintaining my real day-to-day Ph.D. notes. Figure 5.4 shows a Fog browser view that has been tailored for the specific task of assigning article reviews among six topic categories that I felt represented a worthwhile partitioning of the related work. The columns containing entries flagged as `true` allow category membership to be toggled on and off just by mouse-clicking in the cell relating a given note and category. As can be seen from the list in the figure, my choice and allocation of categories leads to substantial overlap between them (items appearing in more than one category)—so here is a potential opportunity for search assistance: perhaps the system, by examining the unions and intersections between the category membership, can derive a structure that partitions the collection into manageable families of related items. These families would guide my construction of sections of the review.

The suggested task was certainly a candidate for mechanised generation of alternatives, but it was not clear that many of the potential results would be meaningful, or that there would be a natural way for the user to steer a progressive search. The doubt arose from the large conceptual gap between the formally specified task (perform set operations on the members of a number of categories) and the informal goal (find a good order in which to discuss

related pieces of work); building a framework that would bridge this gap was a challenge in itself, and was beyond the scope of what I was aiming to demonstrate. So I decided to focus the case study onto a domain in which the user's goals would map more directly onto a formalisable specification—as described below.

Although I eventually decided against using it as the basis for my experimental work, I have continued to rely entirely on the Fog system throughout the Ph.D. At the time of thesis submission the system provides instant access to approximately 700 items amounting to 3MB of text, including all article reviews and the LaTeX source for the thesis itself. I have also continued extending Fog's capabilities over the years; the latest extension was the implementation of an 'outlining' facility to ease the editing and re-ordering of subsections within large documents. Fog runs in the Smalltalk VisualWorks environment, from ParcPlace Systems, Inc.

Second domain: Calendar management

In working with a computer-based version of one's appointment calendar, it may be useful to be able to construct various kinds of constrained view—such as the following:

- appointments within some date or time range—e.g., being in the same day, or same week, or within the coming month;
- a summary showing just one repeating time slot—e.g., only the Mondays, or only the instances of a particular research group meeting;
- only the appointments that belong to some category, such as the teaching commitments;
- only showing the days that satisfy some calculable criterion—e.g., having at least three hours of unbooked time during office hours.

Having observed that many computer-based calendars provide very limited facilities for taking advantage of the potential flexibility offered by dynamic view construction, I investigated ways to provide greater choice. It appeared that one way to do this, without making the facilities so complex that users would have difficulty understanding the formalism, would be to support a *programming by demonstration* approach—although I would need to take care to ensure that the conclusions reached by the system were clear to the user.

A forerunner of programming-by-demonstration was the Basil drawing-by-example system (Maulsby & Witten, 1989). Basil allows users to compose figures from geometrical objects

and constraints, but without having to learn in advance any constraint-expression formalism. Instead, the user proceeds by constructing the visible elements of the figure using simple graphical primitives, and Basil adds the constraint specifications implicitly. But sometimes the system detects that two or more alternative rules could account for what the user had done. For example, a user may draw a line for which it is ambiguous to the system which of the supported forms of constraint is supposed to determine the line's end-point position—is it meant to be at that absolute location, or constrained relative to the start of the same line, or constrained relative to one of the nearby objects? Basil presents a dialog box containing a list of the possibilities, and waits for the user to select the one that best describes the intention on this occasion. All but the first option are alternative ways to generalise the user's specific action, i.e., the drawing of a line between two particular points on the diagram. Note that the active intervention of the system does not save the user from having to specify the desired rule explicitly, but merely from having to think up how it should be expressed.

Similarly, I felt that it should be feasible for a user to build structured views of appointment subsets without having to know the necessary rule language. I considered scenarios for a system that would allow a user to build a view, and would attempt to find all the alternative generalisation rules that were consistent with the user's actions. Were there to arise some ambiguity the system, like Basil, would present the alternative options for the user to select what was intended.

As a simple example, consider a user who wishes to build a view containing all the appointments that fall on Mondays. A way to demonstrate what is wanted is to start building a view containing some of the items that will eventually be members, and to wait for the system to suggest the intended rule. If the user starts by selecting a single appointment from each of two successive Mondays (and if they happen to be instances of the same weekly commitment) the system might offer the following choice of rules:

```
Category
    absolute: category = Teaching        They all (both) have this exact category
    relative: category1 = category2      or is it just that the categories are the same?
Time
   absolute: startTime = 1400            Further valid but irrelevant generalisations
   absolute: duration = 60
   relative: startTime1 = startTime2
   relative: duration1 = duration2
```

```
Date
   absolute: weekday = Monday          This is the rule the user was after
   absolute: month = November          Followed by lots more red herrings
   absolute: year = 1992
   relative: weekday1 = weekday2
   relative: month1 = month2
   relative: year1 = year2
   repeat: 7 days
```

Even this trivial example produces a lot of alternative rules from which the user has to pick the intended meaning. Like the approach in section 5.2, of filtering results by excluding whole regions of notes, a way to boost the efficiency of user control over the generation of candidate rules is to make use of hierarchical categories of rule—for example, that both `startTime` and `duration` evaluations come under the `Time` heading. Since the user can specify that the rule of interest has nothing to do with—in this case—either the Category- or Time-related properties, a large proportion of rule checks can be suppressed. In a more realistic scale of domain, with dozens or hundreds of object characteristics to check, hierarchical constraint on rule-checking would make a big difference.

But on examining further scenarios, including some based on inequalities rather than identity comparisons, it became clear that it was futile to attempt to base the system on an exhaustive search for matching rules (whether or not a satisfactory hierarchical rule classification could be found). To provide support for any of the more interesting cases the rule search would have to be unbounded. So the challenge was to find a way to let a user obtain useful feedback from a search that, if left to proceed of its own accord, could not be guaranteed to generate the desired result in a finite time.

Assuming that one can successfully classify the search paths hierarchically, one way to tackle this problem is for the search to be structured as a tree that is visited breadth-first, and for results to be presented as soon as they are found. Then the user can be given the opportunity to steer the search by deflecting it away from those branches of the tree that count as false leads on this occasion. This kind of steering could be supported using an interface similar to that of the *retrieval by reformulation* systems such as Helgon (described in section 3.2.3), since the domains share the following features:

- At each iteration the system may find and present a long list of potential results, but the user may decide how to refine the search on the basis of detailed examination of just one or two of those results. The rest of the list would typically be ignored.

For an unbounded search the 'rest of the list' might not be enumerable in full. But as soon as enough results have been generated for the user to decide how to refine the search specification, the enumeration at the current level of specification can be suspended.

- In the early stages of the search the user's expectation in examining sample results is not that the desired final result will have been found, but that the results will reveal the factors that are available for steering the search.

  In Helgon the samples reveal the structure of the items held in the database—in particular their property names, types and value ranges. In a rule-generation search the revealed factors will be the names of the rules and predicates that have been tested successfully against the user's actions.

Figures 5.5 and 5.6 show some features of PerspEx[4], a prototype framework that I built for experimenting with my proposals for rule-based retrieval. As a sample domain I used a collection of objects representing my regular weekly appointments, and implemented a simple set of primitives for building different perspectives (ordered or partitioned lists) of those objects.

Figure 5.5 shows two stages in the use of the part of PerspEx that deals with straightforward retrieval by reformulation:

- In the top part of the figure the user has selected as the search domain the collection called `allAppts`, that contains all the appointment objects, and has not applied any other constraints. All the members of the collection therefore appear in the list box titled `candidates`. Within that list the user has selected the appointment with the name `gist_extra`, causing all known properties of that appointment object to be shown in the `candidate properties` box. Some of the displayed properties are part of the object's definition—such as the startTime and finishTime—but others are derived, such as the `overlaps` property that is true in this case for just the two other appointment objects that this one overlaps. Every appointment object additionally has the `isAppt` and `isObject` properties. The user has selected the derived property `pm`, which means that the appointment is in the afternoon, and is about to transfer this into the selection criteria as a property that is to be mandatory.

[4]The name stands for Perspective Explorer—a tool for exploring the construction of alternative perspectives of a given set of data. As usual I built the interface using the Smalltalk VisualWorks environment, and for a logic-based search engine I found a free beta-test implementation of Logic and Objects (an object-oriented extension of Prolog) that I was able to control from Smalltalk by way of an inter-process 'socket' connection.

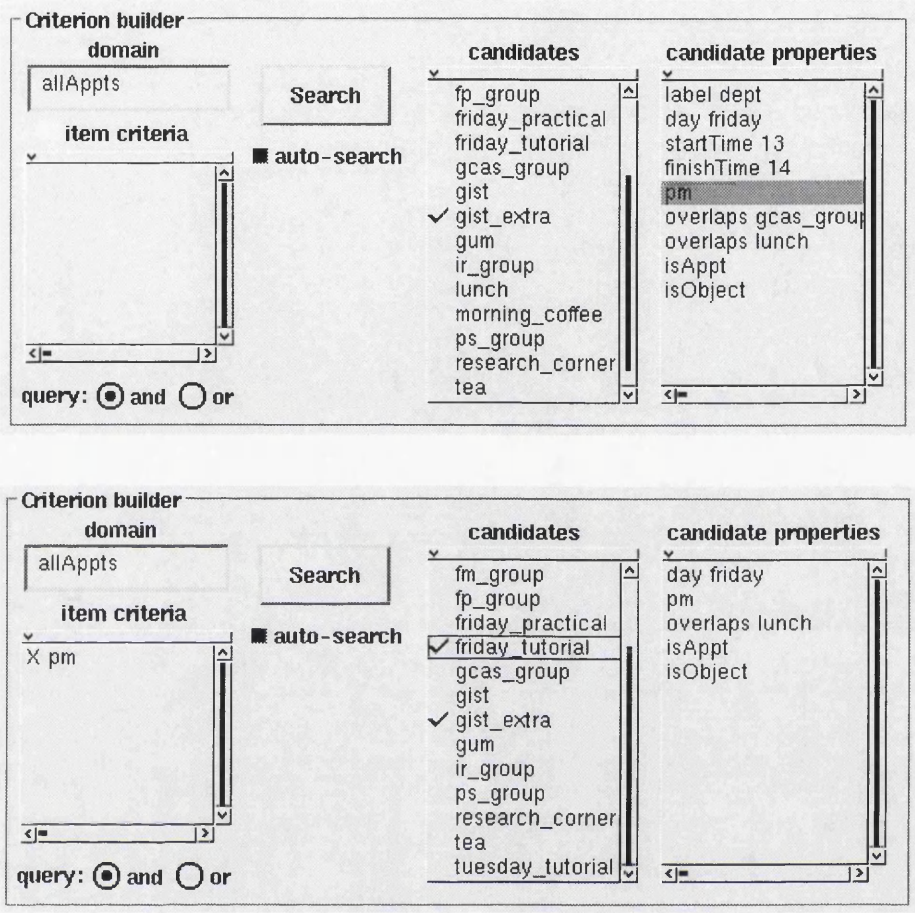


Figure 5.5 PerspEx: Retrieval by Reformulation for finite searches

- The lower part of this figure shows the outcome of the above action: `pm` is now listed as a criterion to be satisfied by any object X for it to be included in the candidates list, and the list is correspondingly depleted. This time the user has selected two of the appointments, so the `candidate properties` view only shows those properties that are true for all (both) the selected items: the facts that they are both on Friday, both in the afternoon, and both overlap lunch.

Figure 5.6 shows a different search as seen in the full PerspEx window (figure 5.5 shows views of just the top part). The main feature of interest is the list on the left entitled `perspective criteria`, whose contents represent a sequence of logic-programming clauses that can express a complex meta-logical search. The criteria in this example have the following effects:

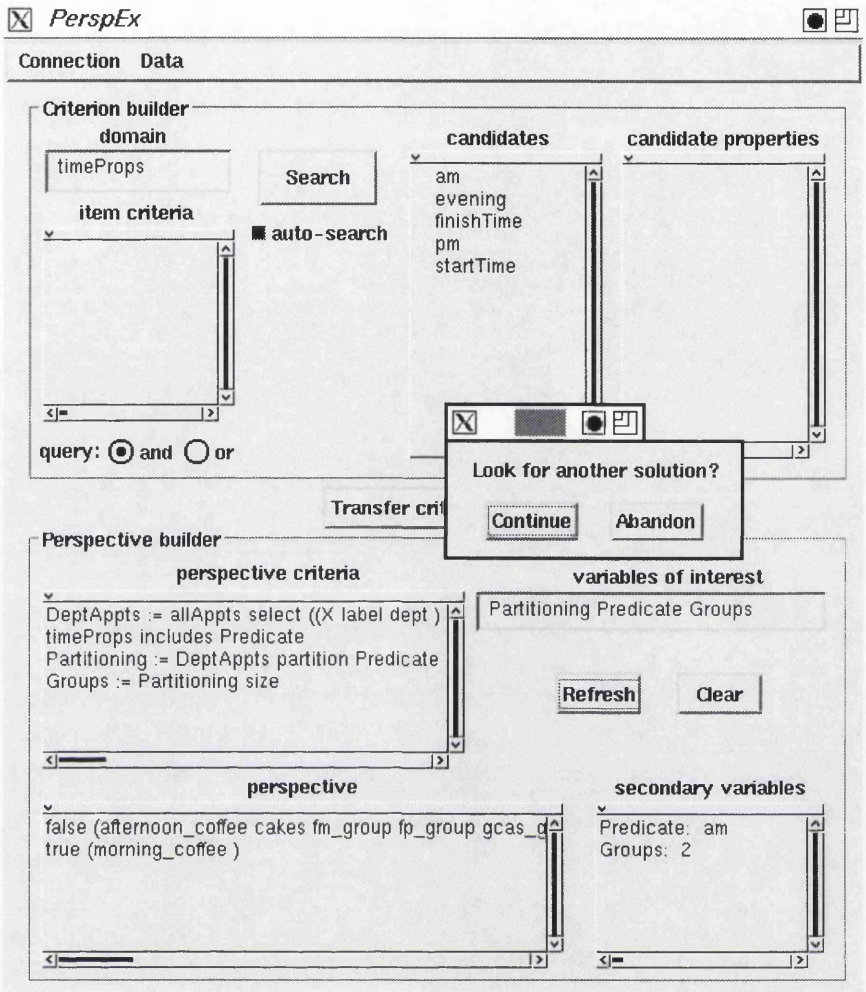


Figure 5.6 PerspEx: Perspectives by Reformulation for potentially unbounded searches

- `DeptAppts := allAppts select ( (X label dept) )`

  This clause assigns to the variable `DeptAppts` a collection containing just the appointments that satisfy the bracketed list of item criteria—in this case, each must have the value `dept` as its `label` property.

- `timeProps includes Predicate`

  For this clause to be satisfied, the variable named `Predicate` (because of how it will be used below) must be unified with one of the values found in the collection `timeProps`. As it happens, the contents of this collection is currently on view in the upper half of the tool window; they are the names of the various known time-related predicates.

- `Partitioning := DeptAppts partition Predicate`

  This is the clause defining the perspective itself: it causes the `Partitioning` variable to hold the representation of a partitioning of the selected appointments according to the currently chosen predicate. A partitioning is represented by a set of (key, list) pairs in which the key is the value of the predicate and the list is all the items with that particular value. There are therefore as many pairs as there are distinct values of the chosen predicate among the sample items.

- `Groups := Partitioning size`

  This assigns to the variable `Groups` the size of the partition set—i.e., the number of pairs.

- (not shown in the figure) `Groups>1`

  Although not shown on this example, it is also possible to add pure Prolog clauses to the list. Since all the clauses must evaluate successfully for a result to be declared valid, this statement would cause the search to reject any partitioning in which all the items fell within a single group.

The state of the tool in the figure is that it has just generated one perspective that meets all the criteria, and is showing what values were assigned to the variables of interest to allow this to occur. Pride of place goes to the `Partitioning`, displayed in a list window at the bottom left. The list at bottom right displays the two other variables that the user has requested—showing that this perspective is the one obtained by partitioning the department-related appointments according to their satisfaction of the `am` predicate, and that it contains two groups.

Overlaid on the main window is a dialog box allowing the user to control whether the system should attempt to find another solution or abandon the search immediately—in this version of the prototype, results are only generated one at a time. One way to use the system, like the standard reformulation approach in Helgon, would be to stop the search as soon as the solution being examined has a property that sparks an idea for an additional criterion to help narrow the search.

5.3.3 Applicability of delegated illumination

This study suggested that there is a class of domains for which 'perspective exploration' could be helpful, having the following features:

- The domain consists of discrete information items, and a way of mapping them into a perspective under user control.
- The user cannot tell in advance how a given mapping will work out.
- The user can instrument some aspects of the generated perspective to provide measures of goodness.
- In cases for which there is a formalisable way of comparing a measure of goodness, the system can be instructed to provide illumination in which the perspectives are ranked according to that measure, or even to select the 'best' available from a batch.
- In cases for which there is no formal criterion for judging a measured aspect, the summarised illumination still saves the user from having to inspect the perspective itself to obtain the value of those aspects.

5.3.4 Tractability

Perspective exploration is clearly a rich area of research to investigate, and PerspEx was useful as a vehicle for starting these investigations. I examined some aspects of the task and the tool in detail, but eventually reached the conclusion that, like the mind-map domain, the multiplicity of novel aspects to the approach would make it difficult to amass evidence relating specifically to the search-control ideas.

5.4 Study 3: Trading-off perspectives by batched processing

5.4.1 Context

One characteristic of the reformulation approach described in the previous study is that it is still difficult for the user to compare the layout and properties of a number of alternative perspectives. The PerspEx interface allows the user to select just one perspective at a time to be viewed in detail, and although it might be extensible to allow the arrangement of a few perspectives side by side on the same display, supporting comparison in domains that give rise to dozens of alternatives would still be impractical.

The hypothesis that I wished to explore was that the side-by-side comparison issue would be easier to solve in a perspective-exploration task in which the alternatives can be considered to lie in a regular multi-dimensional space—for example, when each perspective is defined by a particular combination of values for a fixed set of variables.

5.4.2 Scenario: Perspectives of a shared calendar

A shared calendar has the potential for needing perspective exploration—for example, to help locate a suitable time to hold a meeting between a given collection of people.

This scenario assumes the existence of a simple centralised calendar-management system through which all appointments of the members of a university department can be inspected. Each appointment is identified with a particular time and place, and is tagged with a category such as teaching, laboratory demonstrating, departmental meeting or other meeting. It is assumed that nobody's schedule includes any double-bookings.

One of the staff members, John Psmyth, has been asked to set up a weekly meeting for a new interest group that has been created, scheduling it to avoid conflict with the intended participants' existing commitments as far as possible. This is not a task that can be defined formally in advance, since it is not known how difficult the scheduling is going to be. Psmyth's task will be very easy if there turn out to be one or more times of the week when all the proposed attendees are currently free. But in a busy department it is more likely that even the least disruptive slot still requires one or two people to alter an existing commitment, and there might be a value judgement to make as to who might be asked to put up with what kinds of inconvenience, or some consultations to make before arriving at a decision. In addition the competing commitments might not be the same throughout the term, so some potential slots would only create an occasional conflict.

Psmyth starts by specifying the people who need to be present, and requests a mapping of the working week in which each potential hour-long slot is shown in white if it is free of bookings, and black where any of the required people has an existing commitment. The result (shown in figure 5.7(a)) contains a few free slots, but Psmyth can see at a glance that all the indicated times would be unacceptable. He doesn't have to tell the system that the proposed slots are unacceptable, let alone explain why; he simply ignores them, and continues by trying to shed some light on the areas currently shown in black.

Rather than the yes/no question of whether *any* of the required people has a booking for each time slot, one way to indicate the degree of inconvenience involved for each time would be to show *how many* of the people have conflicting appointments. If it's only one person, perhaps he or she can be asked to rearrange that commitment.

This involves specifying a translation from the number of clashes to the shading of the slot's representation. Figure 5.7(b) shows a possible representation for this; if there are ten people, the number of existing bookings can be shown on a scale from zero to ten. The interface can allow the user to colour portions of the scale to correspond to the way each

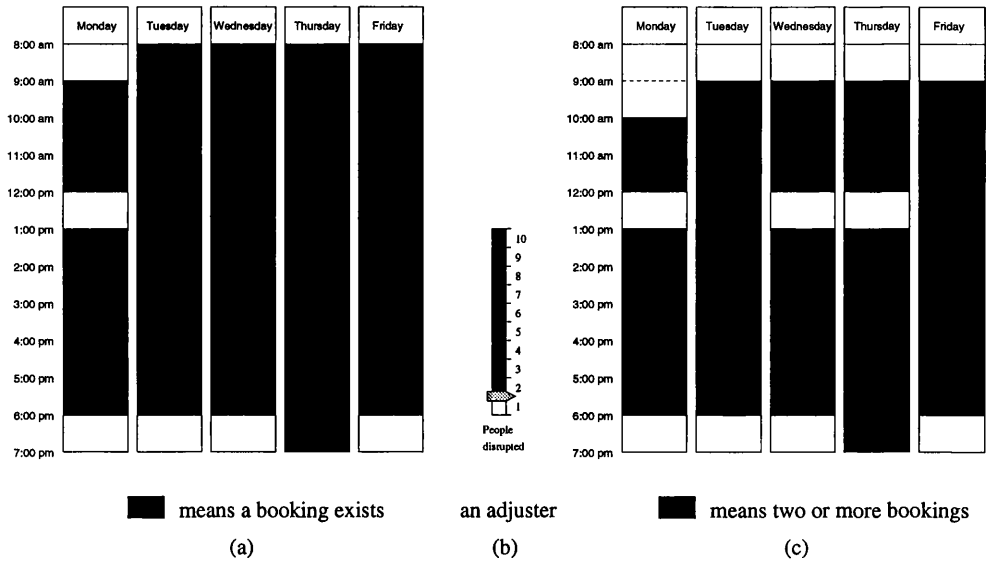


Figure 5.7 Early attempts: no acceptable slots

slot is displayed; this simple example has just two colours divided by a movable pointer. When Psmyth sets the adjuster as shown—meaning that any value of 2 or above (i.e. any slot for which there are two or more bookings) is mapped to black, while the slots with only one booking are now left white as well as the unbooked ones—the result is figure 5.7(c). This turns out to offer no useful alternatives either.

Psmyth continues moving the adjuster up, and the view of the working week is updated accordingly (this being a form of *dynamic query,* as reviewed in section 3.2.5). Very little changes at first, but when the white area of the adjuster reaches 5 the display suddenly contains large areas of white, as shown in figure 5.8. So now he has a new problem: how to discover which of these slots, each involving up to five clashing appointments, might in fact cause less disruption than the others.

To help discriminate the clashes' seriousness, Psmyth calls on two additional factors:

- **apologies acceptable**

  This is the number of occasions, during the timescale of interest, for which a clashing appointment is permitted.

  Without this factor, the system is registering a clash for anyone whose existing appointments clash with the candidate slot in even one of the weeks of term. Setting the 'apologies acceptable' to 2, say, will prevent any clash being registered for an attendee who would only has a previous engagement on one or two of the meeting's slots.

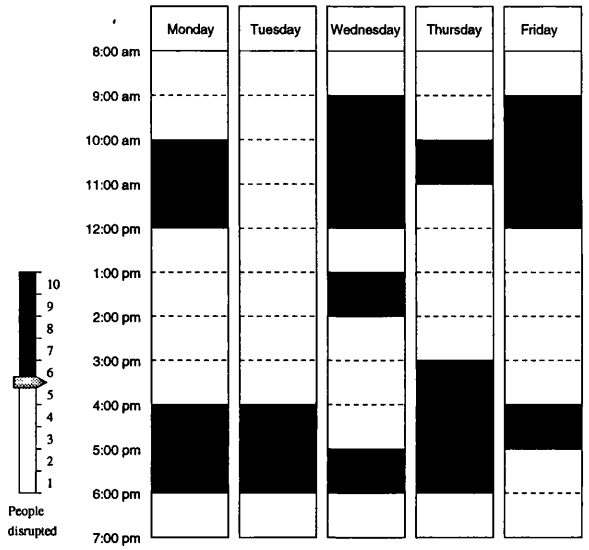


Figure 5.8 Later: too many indistinguishable alternatives

- **flexible commitments**

  This governs which kinds of existing appointment are to be considered potentially reschedulable, and therefore not a serious form of clash.

  Again, without this factor the system is taking the pessimistic view that any clash at all is a serious rescheduling problem. But this need not be the case—for example, asking someone to move an existing appointment is more likely to be successful if the activity is a personal 'housekeeping' task than if it is a university teaching commitment.

  In this case Psmyth decides to distinguish four categories of commitment, in the following order of decreasing importance:

  1. teaching
  2. department meeting
  3. laboratory demonstrating
  4. other

  Deciding on an order of importance allows the categories to be placed on a linear adjuster, like the other criteria. In this case, the black area applies to those commitments that the user considers are not worth trying to reschedule for the sake of the meeting—such as a teaching slot. The white area represents kinds of appointment that the user feels stand a good chance of being reschedulable.

Psmyth might have known in advance of these criteria for narrowing the search—for example, that clashing appointments come in a variety of categories—but he could equally well have encountered them when examining the details of the early results, as in *retrieval by reformulation* systems.

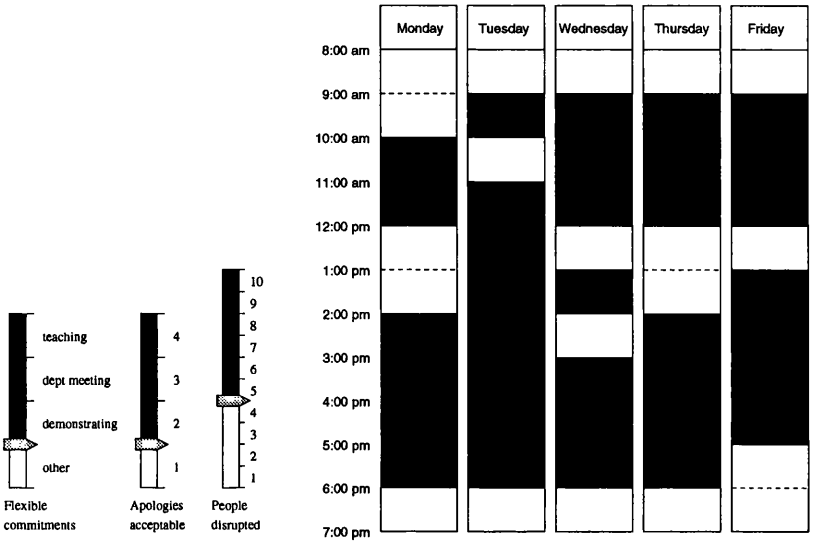


Figure 5.9 Three criteria used in combination

An example of setting the three criteria he now has, and the resulting perspective, is shown in figure 5.9. The way the criteria are combined to determine the displayed colours for each slot is as follows:

- only appointments more important than 'other' are considered disruptive
- iff an attendee has disruptive appointments that overlap the proposed time on more than one occasion during the term, he or she is considered to have a clash
- iff more than four people have a clash the slot is coloured black

Psmyth examines the new results. He knows that the early, late, and lunchtime slots would be extremely unpopular with the group. By selecting in turn each of the other apparently free slots he can see the details of known clashes. For example, he discovers that the slot at 9 on Monday is only a problem for one of the group members, on just three occasions this term—but since the person is the group's organiser and the conflicting appointments are important meetings, that slot cannot be used. Similarly, the other suggested slots each involve important commitments for vital members of the group.

To find other candidate times Psmyth is going to have to change his criteria, one way being to relax the degree of constraint on some or all of the scales he has set up. The question is, which form of constraint-relaxation will give the most useful benefit? If he put the number of acceptable absences up to three, would the view become flooded with available times? If it did, could he narrow it productively by tightening the constraint on the number of people with other forms of clash? There is no way of telling, from what he has seen so far, which slots are going to turn out as the easiest to accommodate.

But the system can help. It can do the job of trying different combinations of constraints, figuring out automatically which slots become available in each case.

Before asking it to do so, Psmyth selects all the slots that he knows are not of interest—all the 8 a.m., 12 p.m. and 6 p.m. times, and the six others that are white in the latest view—and marks them as unavailable. He then calls up an evaluator, which is similar to an adjuster but is used for deciding the validity of a result rather than the output format, and links it to a function that counts the acceptable slots—i.e. the ones that would be white if displayed. He sets this evaluator to accept any value from 1 to 10 (there's no point being left with more than ten appointments to sort out). In addition he sets maximum levels of tolerance for the other three parameters. Figure 5.10 shows this setup, and figure 5.11 the outcome of running the lookahead facility.

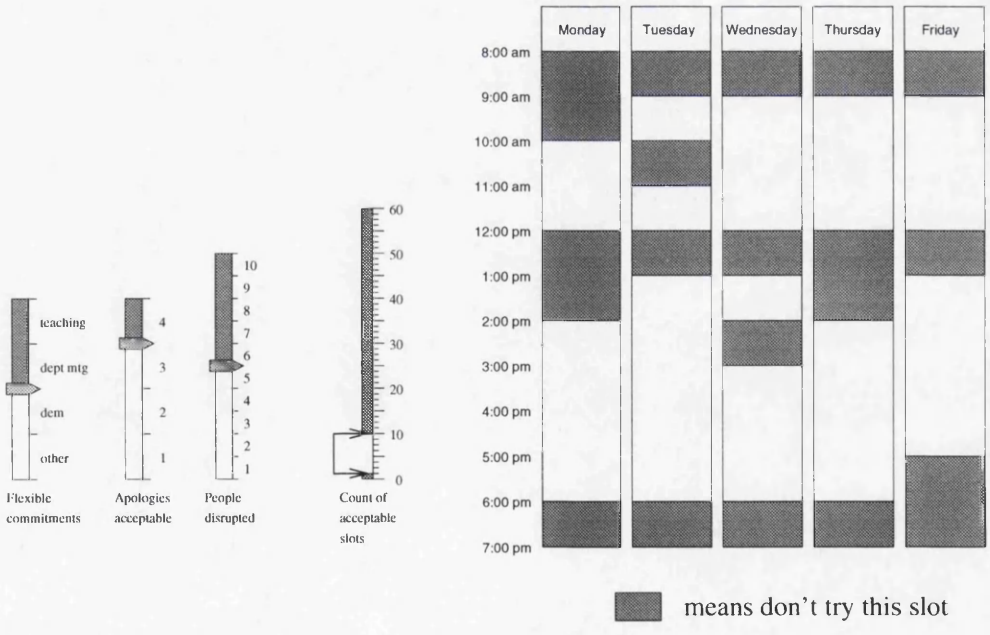


Figure 5.10 Setting up for lookahead

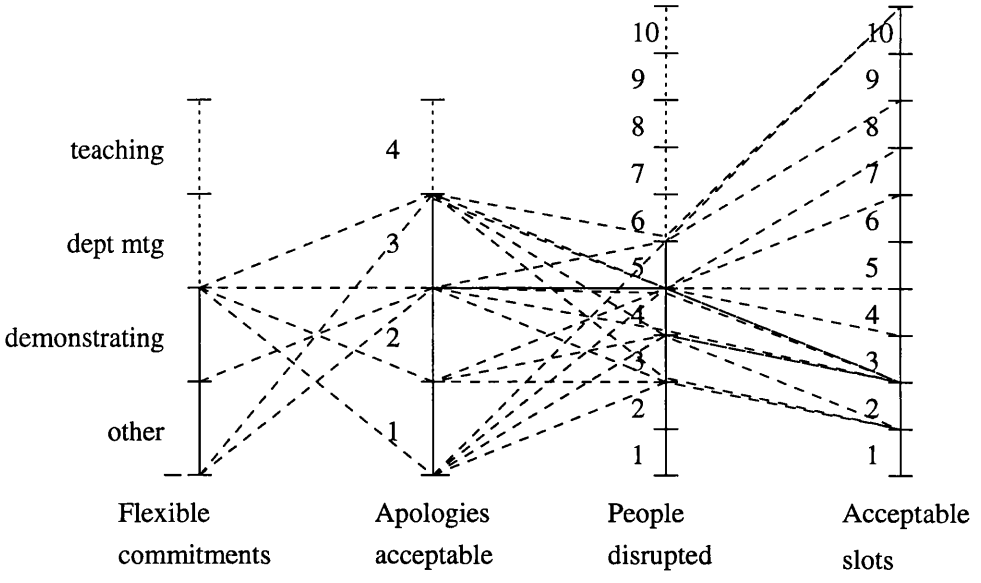


Figure 5.11 Results from lookahead (adjuster details omitted)

The result is displayed on the adjusters themselves, in the form of a parallel coordinates display. Each line connecting the adjusters represents a consistent set of parameter values. For example, the current state of the calendar has generated a line that starts half way up the left-hand scale and passes through the remaining scales just above the 1, 4 and 3 points respectively—this showing that, if 'demonstrating' and 'other' commitments are to be regarded as open to negotiation, and people with one (or no) known conflicting appointment are not a concern, then there are three slots that each involve disruption for four people. Plotting all the lines on top of each other makes it difficult to see all these individual relationships but points out the generalisations—for example, that whatever allowances are made there is no slot for which fewer than two people's schedules are disrupted. The tool should let the user inspect individual results, for example by highlighting just the results that pass through selected points, and should also make it easy to obtain the corresponding perspectives so the user can see which slots are being proposed.

One important aspect of this result, as far as Psmyth is concerned, is the fact that there are lines passing through the 'people disrupted' scale at the '2' and '3' levels. These show that, given some combination of allowances, there are slots that will cause disruption to only two or three people. The system has done the work of trying all the combinations of criteria, so it can show that although these particular slots do require demonstrator commitments to be altered they do not involve making any allowance for occasional absence.

5.4.3 Applicability of delegated illumination

This proposal provided the further insight that it would often be valuable for the user to be free to use a given piece of illumination in a variety of ways. For example, the appearance of the free lunchtime slots could simply be ignored, or explored for clues on how to narrow the search, or used to specify directly that these slots are not of interest.

In addition it was attractive how easily the user could request a change from user-driven to delegated exploration, by virtue of the slider controls also being capable of indicating the desired illumination range.

The lookahead idea led directly to the idea of reconnaissance, of course. In particular this scenario suggested the ease of providing a useful kind of illumination in a domain for which the inputs and the measurements could each be shown on a linear scale such as a slider. These criteria were further developed into the concepts of *option spaces* and *measurement spaces*, as described below.

5.5 Option spaces and measurement spaces

In some computer-supported domains each result can be described by its value for each of a fixed set of options. Figure 5.12 represents the requesting of result illumination in such a domain by specifying a combination of option values or ranges. In the figure, option A is a choice the user might normally select from a menu, such as the required class of a flight; B is a boolean choice such as whether or not a text formatter should perform right-justification; C is a numerical value such as the width of text between margins. Many of these domains also allow the illumination request to include an option scale representing a measured property of any results that are found: for example, option D in the figure might be specifying the acceptable range of transfer times between connecting flights. Depending on the task domain and the specified option values a single combination of options and constraints may lead to zero, one, or many candidate results.

In such a domain, a simple form of reconnaissance range can be described by specifying a number of acceptable alternative values for each option. An example of this is the reconnaissance range requested in the scenario in section 4.2, which was expressed as three alternative departure dates, two alternative departure cities, and so on. I refer to the range implied by the combinatorial expansion of all specified values for a set of options as an **option space**.

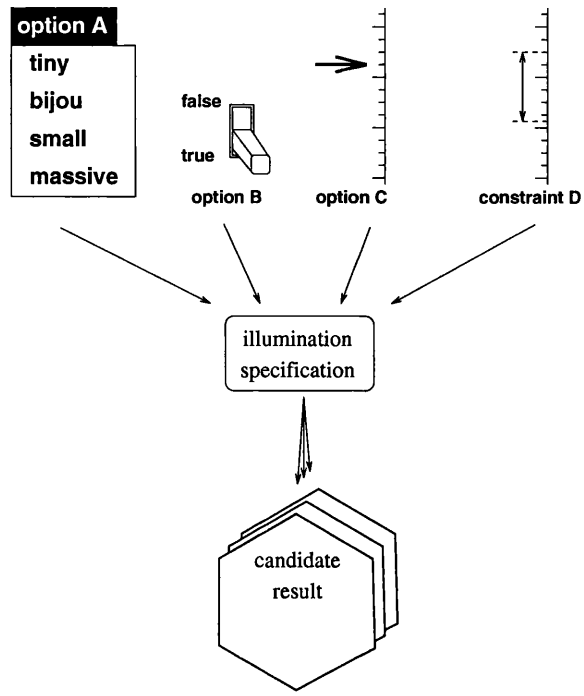


Figure 5.12 Specifying illumination as a combination of option values.

The specification of reconnaissance content can also be handled as a set of independently handled dimensions. Figure 5.13 shows the relationship between option selections made by a user and the properties of a corresponding candidate result. Now that results are being measured, each result's actual value for parameter D—which will be within the constrained range specified on D as an input—is likely to be one of the measurements of interest to the user. Measure E is numeric, such as the proportion of white space on some page of a formatted document; F is a discrete-valued measure for which there is an unpredictable but limited range of possible values, such as alternative airlines; G is a boolean test such as whether a candidate home includes air conditioning. The notional region defined by combinatorial expansion of all possible values for a given set of measurements can be called a *measurement space.*

Figure 5.13 also shows how each candidate result within an option space/measurement space illumination can be summarised using a multivariate data item that correlates the chosen option values with the discovered measurements. Production of a collated report of a set of candidate results summarised in this way can therefore be handled as simply a task of multivariate data presentation.

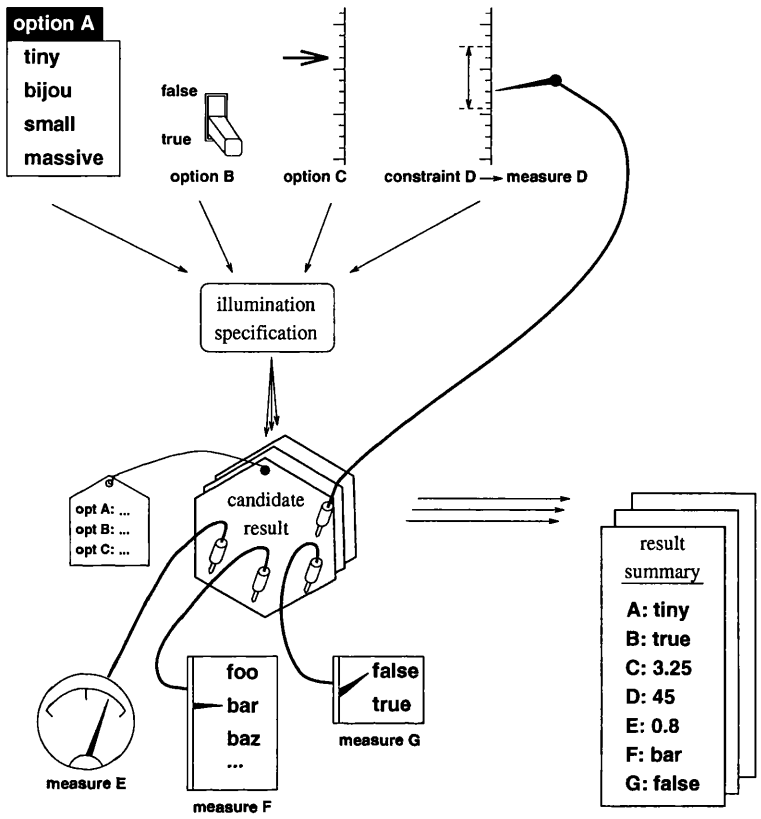


Figure 5.13 Specifying candidate-result evaluation in terms of a set of measurements.

5.6 Conclusions

This chapter recounts the various studies I pursued in the early stages of my thesis work, which can now be seen as domains for which reconnaissance would be useful but complex to implement and test.

One of the most valuable outcomes of these studies was the identification of option spaces and measurement spaces, a simple means for expressing reconnaissance in a promisingly wide range of task domains. The ability to express results from such domains as multivariate data items transforms the challenge of providing a collated representation of a reconnaissance foray into one of producing a multivariate data display. An example of a flexible multivariate representation is the parallel coordinates technique, that has already been used for illustrative purposes at various points in this thesis. The next chapter reviews the development of this technique and some of its existing computer-based implementations, to assess its suitability as the representation for a reconnaissance support system.

Chapter 6

Using parallel coordinates as a reconnaissance interface

6.1 Introduction

Having demonstrated that the results from reconnaissance in at least some kinds of task can be expressed as multivariate data items, and noting that the parallel coordinates presentation technique is a powerful way to handle multivariate data, it was decided to investigate the feasibility of producing a parallel-coordinates-based reconnaissance system.

The first part of this chapter is a review of the parallel coordinates representation, and various existing interactive embodiments of it. This is a detailed survey intended primarily to demonstrate the power of the representation. In particular, because use of this representation is only slowly pervading the commercial and academic domains, the survey begins at a basic level by describing some of the simple 'emergent' properties of parallel coordinate diagrams; maybe in a few years such explanations will have become as unnecessary as it would now be to explain how to detect clusters and correlations in a two-axis scatter plot.

A further purpose in providing a detailed survey is that the interface implemented within the main evaluation exercise (described in chapter 7) inevitably contains only a subset of the latest interactive features developed for parallel coordinates. The survey allows the reader to appreciate the potential of the technique beyond the capabilities of the particular instance that is demonstrated.

Section 6.3 then shows how an interactive version of parallel coordinates, with various specialised facilities, could support the activities involved in pursuing reconnaissance.

6.2 The parallel coordinates presentation technique

This review of the existing work on parallel coordinates is structured as follows:

1. an overview of the development of the technique, and ways in which it is being used and developed;
2. a description of the fundamental data-analysis activities supported by a parallel-coordinate mapping—i.e., what kinds of relationship between mapped items appear as visual features that are easily detected by a person. This section includes a brief comparison with other ways of presenting multi-variate results;
3. a review of various existing exploratory data analysis tools (including two commercial products) based on interactive parallel-coordinates displays.

6.2.1 Development and applications

General development

The recent development of the parallel coordinates visualisation technique is mainly driven by the work of Alfred Inselberg and others, working at the University of Southern California and at IBM's Los Angeles Science Center. The primary emphasis of their work has been in the field of computational geometry (the early work of the group includes Inselberg 1981 and 1985a), but they have also explored various applications in decision support. Inselberg, Dimsdale, Chatterjee & Hung (1993) provide a bibliography of some of the most significant recent developments, while Inselberg & Dimsdale (1994a, 1994b) give a thorough review of the mathematical properties that underlie all uses of the technique. Another thorough but more application-oriented overview is (Inselberg & Dimsdale, 1991).

Further work in computational geometry

Chatterjee's thesis (Chatterjee, 1995) presents a parallel coordinates method for visualising multi-dimensional polytopes[1], and then explores the application of these techniques to linear programming problems. Also see (Inselberg, Reif & Chomut, 1987).

[1] Polytopes are generalisations of 2-D polygons to N dimensions, i.e., connected regions in N-space bounded by hyperplanes (definition from Inselberg, Dimsdale, Chatterjee & Hung, 1993).

Multivariate data visualisation and analysis

Many researchers have investigated the power of the parallel coordinates display for exploratory data analysis. For an excellent introduction to this area see (Wegman, 1990); other work includes the thesis by Chomut (1987), and some simplifications for expression of confidence intervals proposed by Magleby, Burton & Scott (1991). The WinViz *multi-dimensional data visualisation* tool, described in (Lee, Ong & Sodhi, 1995a) and reviewed below in section 6.2.3, has also been used to assist in visualising the rules generated by automated *knowledge discovery* in a large result database (Lee, Ong & Sodhi, 1995b).

Also described below are the Attribute Explorer demonstration (Tweedie, Spence, Williams & Bhogal, 1994) and its derivative, the Influence Explorer (Tweedie, Spence, Dawkes & Su, 1995). These incorporate a form of parallel coordinates presentation in which the result items are shown in stem-and-leaf plots along each axis.

Computer-based multivariate representations that exhibit some of the properties of parallel coordinates include Eick's (1994) *data visualisation sliders* and Chimera's (1992) *value bars.*

Process modelling

As well as supporting visualisation of multivariate domains, the parallel coordinates representation can facilitate various forms of calculation. Examples include decision analysis techniques such as *multi-objective programming* (Ng, 1991) and *data envelopment analysis* (Desai & Walters, 1991), although the latter should be read in conjunction with Inselberg's (1985b) insights into how the mathematical properties of the representation can help in modelling the decision task as one of staying within the bounds of a hypersurface.

At a more concrete level yet, Inselberg and Dimsdale (1994b) describe patents pertaining to a parallel coordinates tool specialised to the performance of route-planning calculations in air-traffic control.

Commercial products

Computer-based adaptations of parallel coordinates appear in various tools, including the statistics software packages BMDP, Systat and NCSS. Two of the most recent products to emerge are aimed at general markets—both are designed for producing visualisations of existing bodies of multiattribute data, and include various means for building displays and manipulating, marking and filtering results:

The Parallel Visual Explorer developed by the Software and Visualization Solutions group of International Business Machines Corporation (IBM), Yorktown Heights, NY.

WinViz developed by the Information Analysis group at the Singapore Information Technology Institute (ITI).

Some of the special features of these products are reviewed in section 6.2.3.

6.2.2 Fundamental properties

An intuitive use of parallel coordinates for representing a collection of multi-variate data points has already been illustrated in the examples in section 4.2.2, including the sequence of figures starting with figure 4.2 on page 82.

But to understand some important characteristics offered by a parallel-coordinates presentation one must become familiar with some of the basic properties of the mapping. In the following explanations I use the terminology employed by Inselberg, Dimsdale, Chatterjee and Hung (1993) and by Inselberg and Dimsdale (1994a):

> 'In the Euclidean plane R^2 with xy-Cartesian coordinates, N copies of the real line labeled $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_N$ are placed equidistant (e.g., one unit apart) and perpendicular to the x-axis. They are the axes of the *parallel coordinate* system for R^N all having the same positive orientation as the y-axis. A point C with coordinates $(c_1, c_2, \ldots, c_N)$ is represented by the complete polygonal line (i.e., the lines of which only local segments are shown) whose N vertices are at $(i-1, c_i)$ on the $\bar{X}_i$-axis for $i = 1, \ldots, N$.'

Figure 6.1 shows the representation of a single point C. Note that the polygonal line $\bar{C}$ includes the complete lines, extended through and beyond the dotted segments shown here, and not just the portions between the axes.

The line–point duality in mappings of two dimensions

The next important property is easiest to demonstrate for mapping 2-dimensional data, where it manifests itself as a duality between lines and points in the Cartesian and Parallel Coordinates plots.

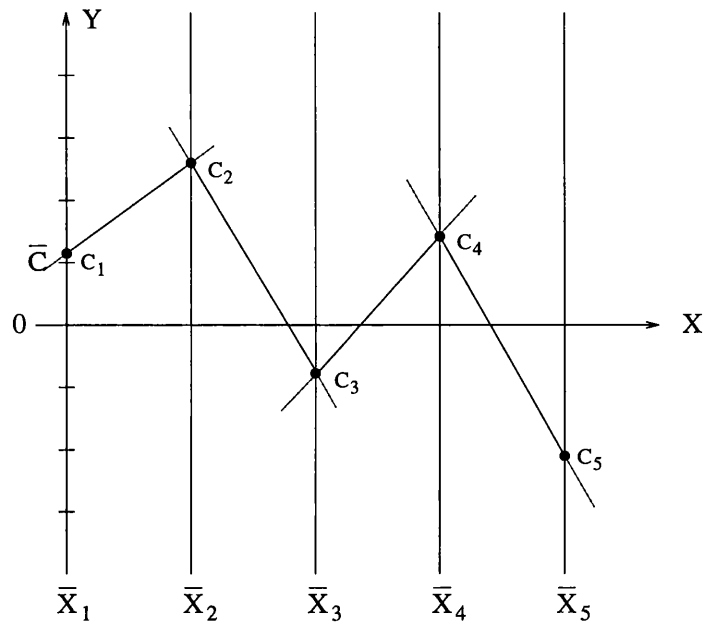


Figure 6.1 Parallel axes for R^N, when $N = 5$. From Inselberg *et al.*, 1993.

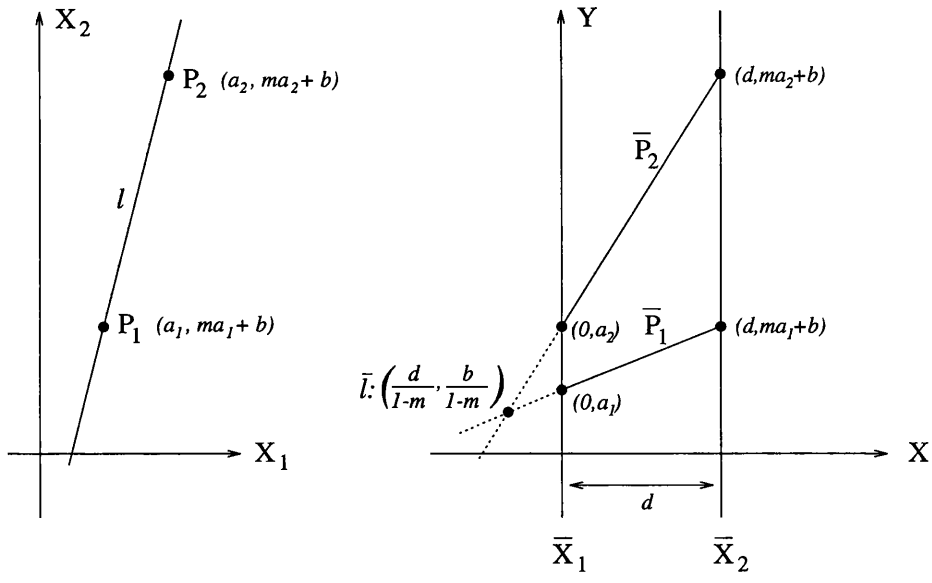


Figure 6.2 The line–point duality

Consistent with the convention adopted above, I refer to a point in the Cartesian space using a capital letter (e.g., P) and a line using a lower-case letter (e.g., l). Their representations in the Parallel Coordinates space are referred to as $\bar{P}$ and $\bar{l}$ respectively.

Figure 6.2 shows two points on a line

$$l : x_2 = mx_1 + b, \qquad |m| < \infty$$

and their corresponding lines on parallel axes. The point where those lines intersect, and where *all* lines representing points on l will intersect, is shown as

$$\bar{l} : \left(\frac{d}{1-m}, \frac{b}{1-m}\right)$$

where d is the distance chosen as the separation between the adjacent axes.

Spotting linear relationships

Notice that the position of the point $\bar{l}$ relative to the axes only depends on m and b, the determinants of the line l. In particular the gradient of l, determined by the coefficient m, governs the position of $\bar{l}$ relative to the axes $\bar{X}_1$, $\bar{X}_2$:

- For $m > 1$, as in the example in figure 6.2, $1/(1-m)$ is negative—i.e., the intersection occurs to the left of $\bar{X}_1$. As m reduces, approaching 1, $\bar{l}$ moves further and further left and the lines $\bar{P}_1, \ldots \bar{P}_n$ become closer and closer to being in parallel.

  In this region of m the values on the line are exhibiting a positive correlation between their two variables. When the lines are exactly parallel the variables change in lock-step; the common slope of the lines on the parallel coordinates only depends on the offset, b, of one variable from the other[2].

  In figure 6.3, case (a) shows a line with a slope close to 1 (in this case just greater than 1; if the slope were just less than 1, the meeting point would be to the far right of the axes).

  As $m \to \infty$, l is tending towards a vertical line at constant x_1, and $\bar{l}$ moves onto the $\bar{X}_1$ axis. This is shown as case (b).

[2]For the sake of consistency, parallel lines can be modelled as meeting at *ideal points* whose theoretical positions are determined by the lines' slope. See Inselberg & Dimsdale (1994a) for a description of one way to handle the parallel-coordinates mapping as a projective plane based on a hemisphere, which allows ideal points to be modelled as appearing on the hemisphere's equator.

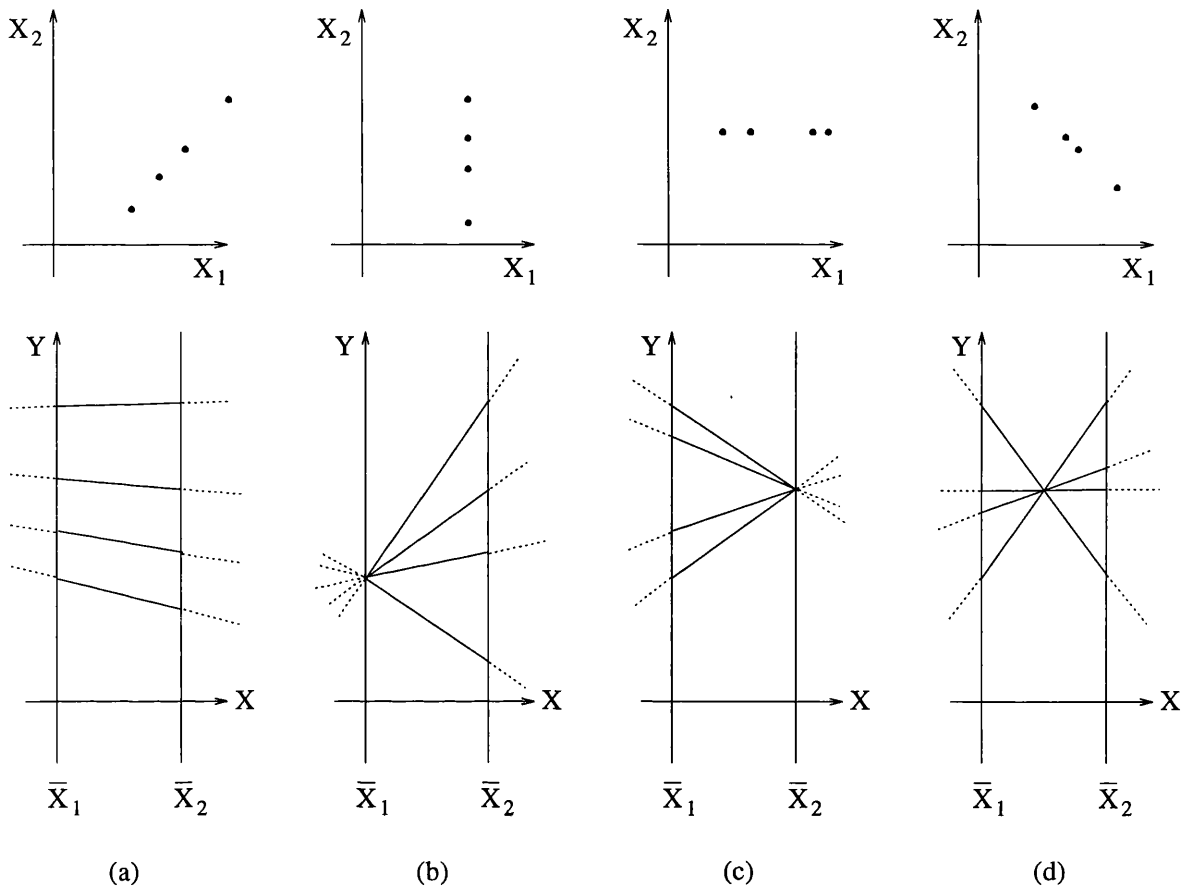


Figure 6.3 Examples of collinear points mapped onto parallel coordinates: (a) $m \approx 1.2$; (b) $m = \infty$; (c) $m = 0$; (d) $m \approx -1$. (The figures are not drawn to scale.)

- For $0 < m < 1$, the intersection occurs to the right of the axes. This is still a positive correlation, but in this case X_1 is growing more quickly than X_2.

  As m approaches 0, l is tending towards a horizontal line at constant x_2, as shown in case (c) in the figure.

- Finally, for $m < 0$ the intersection occurs between the two axes, appearing at the half-way point when $m = -1$. Such a situation, shown as case (d), corresponds to negative correlation: as one variable increases the other decreases, leaving a tell-tale crossover point between the axes.

A result of these conditions is that when plotting a number of data points that are approximately collinear with respect to two of their variables, the parallel coordinate mapping will

contain a region of lines that all pass through or near a particular point on the diagram. In a real-life data analysis task there may be many small independent regions of linearity, which will appear as groups of lines that pass through different common points. The person performing the analysis has to develop the skill of being sensitive to detection of such groups—unless interactive assistance is available, as described in section 6.2.3.

Basic statistical interpretations supported

Wegman's (1990) excellent review of the technique includes a basic vocabulary of 'statistical interpretations' supported by parallel-coordinates displays. They include:

- **linear relationships** as described above
- **overall correlation structure between adjacent variables** based on the degree to which obvious crossovers, converging lines, parallel lines are evident in the spaces between axes
- **inter-variable cluster relationships** both between adjacent variables and, provided the data lines are not packed too densely, traceable through to other variables
- **distribution** such as detection of the *mode* of the results, appearing as the most dense bundle of paths seen to flow as a group through the diagram.

The paper includes a compelling illustration of the application of these techniques in the analysis of a data set relating to 74 different models of automobile plotted on five data dimensions.

Support through interactive alteration

Although parallel coordinates can be used as a static, paper representation, many additional capabilities are opened up when the plot is performed on a computer and an interactive interface is provided.

Wegman notes the following kinds of support:

- **axis normalisation** For a given data set the user may find that all the points have a similar value for one or more of the variables, and therefore pass through narrow regions on the corresponding axes. This makes it hard to distinguish the results on the basis of those variables, so it may be useful to re-scale such an axis so the data of interest are distributed over the full axis height.

- **different permutations of variables** In a plot of n variables (without duplication), $n-1$ pairwise correlations are visible at any time out of the total of $n(n-1)/2$. Wegman shows that the minimum number of permutations needed to ensure that every pairwise combination has been seen is $(n+1)/2$—for example, for ten variables you would need six permutations. In a typical case, however, the user's understanding of the problem domain will rule out many of the correlations as uninteresting.

- **reversing the sense of some axes** Because of the tell-tale crossover points, a negative correlation between two axes can be easier to detect by eye than a positive one; conversely, in some cases a lack of crossovers will help to isolate distinct regions of correlation. Reversing the sense of alternate axes in the display will reverse the sense of all displayed correlations, and may bring to light some relations that were previously obscure.

Wegman also introduces variations on the plotting technique, using aggregation and *density plots,* that are suited to working with very large data sets for which the user's goal is less likely to be the identification of individual results than the discovery of overall correlation structures between variables.

Relationship to other multi-variate presentations

Many techniques have been developed for helping people to visualise, and hence understand, multivariate data. It is a vast area of study, into which I can only hope to provide a few pointers. The techniques considered first are those that help the user to spot individual results with characteristics that are salient, or are simply of interest at the time, out of a large sea of data. Others emphasise the search for overall trends in the data.

One of the most memorable techniques emphasising the representation of individual items is the use of faces with varying features, as proposed by Chernoff (1973) and Flury & Riedwyl (1981)[3].

Tufte (1990) provides an attractive and informative historically based review of information presentation techniques—but his examples are all *static* displays, in which there is a clear goal to communicate a particular known message in the data (and perhaps to obscure other possible messages). By contrast, in the tradition of *exploratory data analysis* launched by Tukey (1977), the idea is that through the manipulation of the way a data set is visualised

[3]More recent work includes an evaluation of Flury-Riedwyl faces by De Soete (1986), and an interface developed by Curtis and Scarfone (1992) using which they claim to be able to map either 17 or 34 distinct variables to a face depending on whether or not it is constrained to be symmetric.

users of data may discover information they did not realise was there. Since the nature of the information is not known in advance its detection and presentation cannot be automated; the presence and interaction of the person is crucial, as Bertin (1981) stated:

> 'Graphics offers the means of going beyond what can be automated. We can find numerous methods for automating the diagonalization of a matrix, but we will never find methods for automating the conception of the *most useful* subsets. Graphic permutations can only be carried to their conclusions by the "decision-maker" himself.'
>
> (p.21, added emphasis)

Bertin's own technique was based on a tabular 'graphic' representation that makes it easy for the user to perform visual analysis to discern rows and columns with similar or inverse properties. By physically rearranging the table the user can build up large overall patterns revealing some trends in the data that were (a) originally hidden, and (b) of interest to this particular user. Bertin, at least initially, performed his work either on folded and marked strips of card, or using large arrays of domino-style pieces representing the data. Computer support for this approach has since appeared—e.g., Marsh's (1992) Interactive Matrix Chart.

So now we are clearly in the domain of techniques for spotting trends among data, rather than especially salient individuals. But the Exvis visualisation environment (Grinstein, Pickett & Williams, 1989) is claimed to combine the two approaches: users create landscapes of small iconographic representations of the individual data points, where each character-like 'icon' may vary in a number of abstract but salient features, such as the orientation of a main stem or the angle and shape of various branches. The user is expected to gain impressions of the data set from the overall 'texture' of the resulting display, as well as possibly having his or her attention drawn to individual icons that have interesting shapes. Iconographer (e.g., Waite, 1991) provides an interactive environment in which a user can experiment with the construction of iconographic displays such as these, altering the mapping of data variables to icon characteristics and placement until a revealing presentation is discovered.

As Tufte notes (1990, p.38) a display that conveys information at the panoramic as well as at the 'micro' level is a combination the human visual-processing systems seem well adapted to using. Parallel coordinate displays possess this property too, and correspondingly possess the constraint that where results are very densely packed the task of discerning individual entities becomes difficult.

There is also a branch of representations that condense the display of multiple dimensions by using projection, whether onto a single 2-D display (e.g., see Vetschera, 1994) or onto

a matrix of scatterplots each showing correlations between different variables (e.g., see Cleveland and McGill, 1984). Scatterplot matrices, like the parallel coordinate presentation, can be drawn for an arbitrarily large number of dimensions—as long as you have the display space. An important landmark in an alternative direction—the use of computers to provide *dynamic* visualisation techniques—is the collection of papers assembled by Cleveland and McGill (1988).

Wegman (1990) provides some direct comparisons between the parallel coordinate and other representations, in particular pointing out the way it avoids some pitfalls in projection-based techniques. But overall he regards the parallel coordinate representation 'as complementary to scatterplots. ... Because of the projective line–point duality, the structures seen in a scatterplot can also be seen in a parallel coordinate plot.' He suggests that a major advantage of the parallel coordinate representation over the scatterplot matrix is the linkage provided by connecting points on the axes, which is difficult to duplicate in the scatterplot matrix. In favour of the scatter plot he notes that it uses a comparatively small amount of ink for each result, so one would expect results to be less prone to becoming cluttered and indistinguishable than the mass of lines on a parallel coordinate plot. But he briefly describes some of the computer-based parallel coordinate *density plot* techniques that he implemented to overcome this problem.

6.2.3 Applications in data visualisation

The IBM Parallel Visual Explorer

Figures 6.4 to 6.7 show views of an IBM visualisation and data analysis package that was announced at the end of 1994[4]. The package is a straightforward implementation of the parallel coordinates method, with an environment that supports various query and manipulation tools for analysing the presented data.

The information used here is taken from an overview on the IBM World Wide Web site, and from the User's Guide for Version 1, Release 1.0 of the product.

Figure 6.4 shows a plot of 384 ten-dimensional data items corresponding to various financial measures collected each Monday over a seven-year period. The user has applied two separate 'range' queries on the YEAR scale, to highlight the results corresponding to the years 1986 and 1992. The corresponding result lines are re-drawn in a colour designating each query,

[4]IBM Announcement Letter No. ZP94-0897 dated December 20, 1994: Program Number 5765-469 IBM Parallel Visual Explorer for AIX. Note that the product makes widespread use of colour for distinguishing data items; here the colours can only be shown as shades of grey.

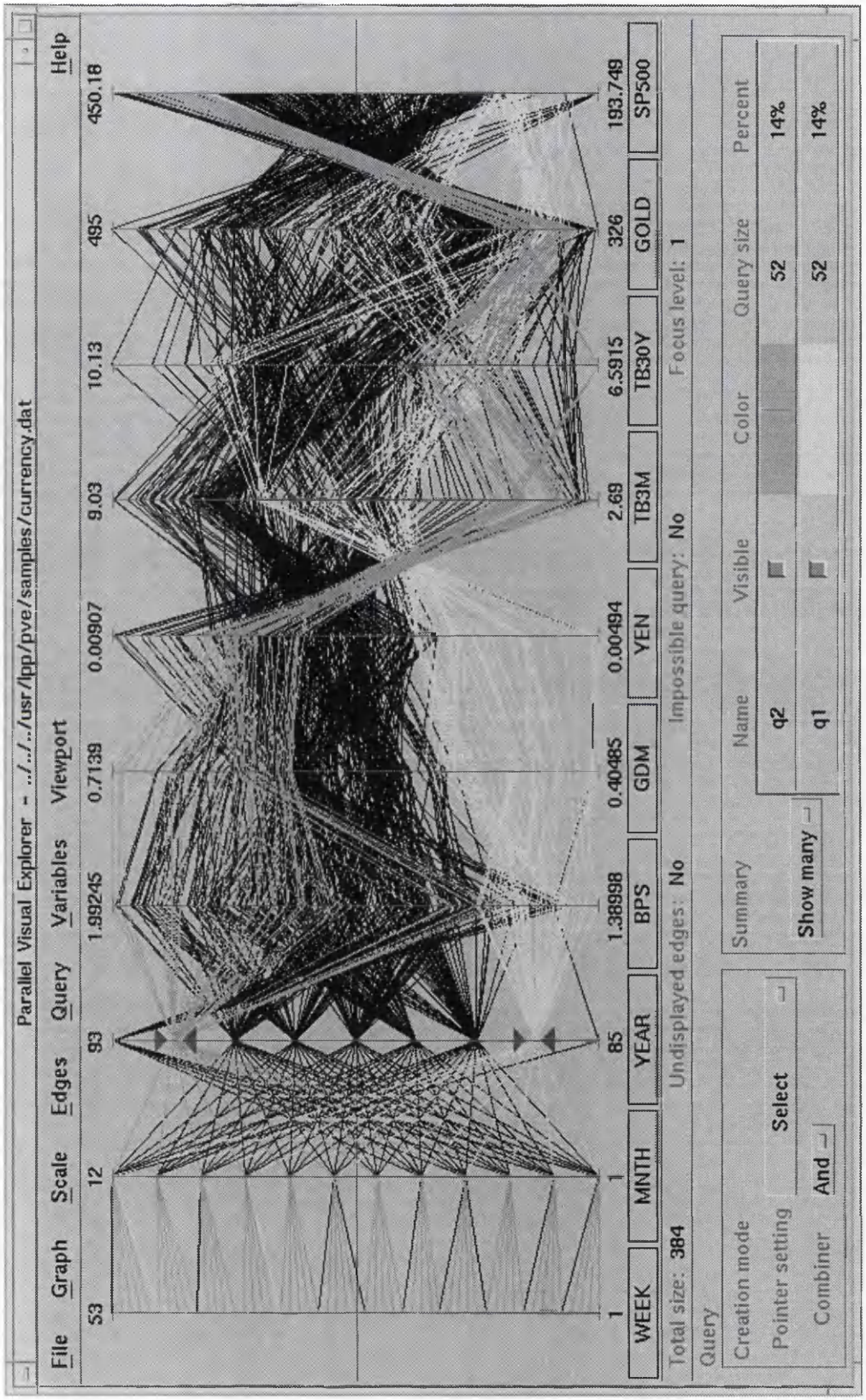


Figure 6.4 The IBM Parallel Visual Explorer, working on a set of weekly financial measures from the New York currency markets covering the years 1985 to 1993. On this display the user has specified two independent queries.

while the other results remain plotted in black. In this example the user can see that 1986 (actually plotted in yellow) was characterised by a strong dollar—showing low values on the exchange-rate scales against BPS (British Pound), GDM (German Deutsche Mark) and YEN—and low stocks, as shown by SP500 (the Standard and Poor's Index). By contrast, in 1992 (plotted as light blue) the dollar was relatively weak and the British Pound highly volatile, and throughout the year gold prices were low and stocks high.

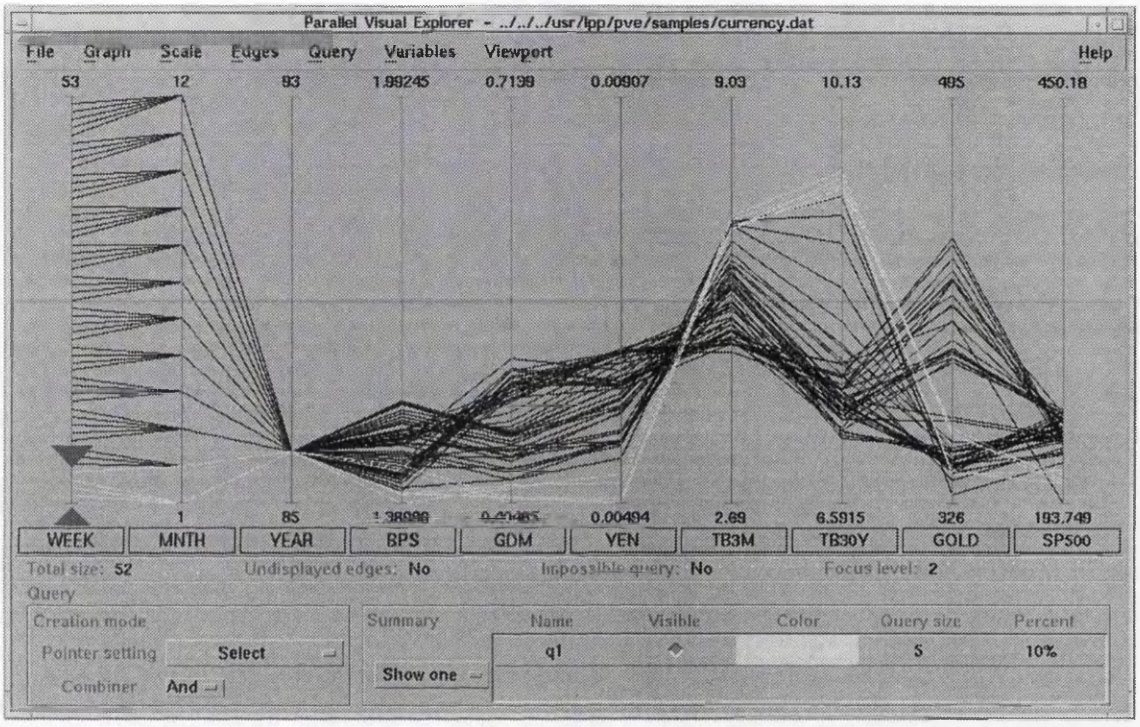


Figure 6.5 The data narrowed to contain just the results identified by query `q1` in the previous figure. A new query has been applied at this lower 'focus level'.

In figure 6.5 the user has isolated the 1986 data points by 'focussing in' on the results of the query previously shown as `q1` and plotted in yellow. Now a fresh set of queries can be applied. Here the user has set up a range query bracketing a five-week section of the WEEK scale, and can slide this bracket up and down the scale interactively to highlight the results from different times in the year.

The queries shown so far are examples of 'range' query, enabling selection of a range of values on one or more axes. PVE also supports other forms of query specification:

- **slice:** a more general form of range, in which the markers can be placed in the space between axes

- **wedge:** in which the markers are not constrained to be vertically one above the other. The results selected are those whose lines pass above the upward-pointing marker but below the downward-pointing one.
- **flow:** for selecting results whose lines, between two particular axes, have a *slope* within a specified angular range

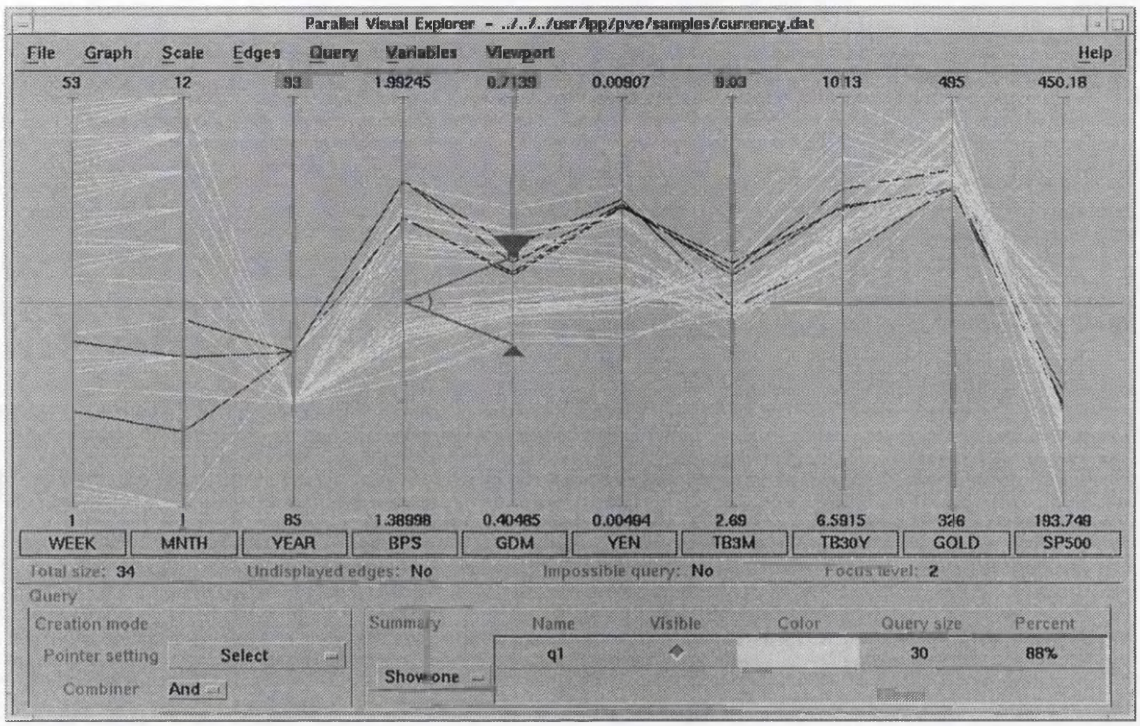


Figure 6.6 A 'flow' query, selecting results whose line slopes fall within the specified range.

Figure 6.6 shows an instance of a 'flow' query placed between the BPS and GDM scales. The highlighted lines appear to be converging, suggesting a linear relationship. This can be confirmed by requesting a scatter plot of these two variables, as shown in figure 6.7. Note that the slope of the linear relationship is just a little less than 1 (i.e., just below 45 degrees), which is what one would expect from a convergence point to the far right of the two axes concerned.

PVE also contains many other facilities supporting the specification of queries and derived variables, permutation of variables, automated clustering, etc. One powerful interactive feature is the facility for adjusting the specification of queries either on the parallel coordinate display or on a derived scatter plot.

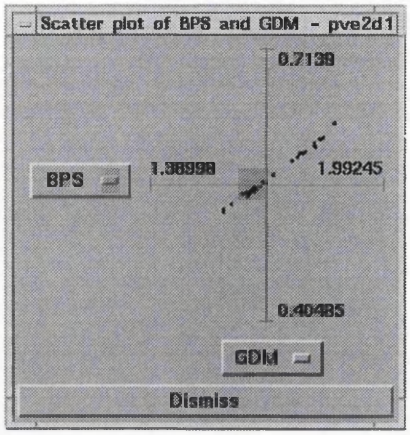


Figure 6.7 A scatter plot of the results selected in figure 6.6

An insight into the range of data analysis activities for which PVE is expected to be of value is provided by the following guidelines that appear in the PVE User's Guide[5] for users undertaking exploratory data analysis. The main headings and recommendations are as follows:

1. Set a goal: What are you trying to find in the data?

   It is noted, however, that you don't need to establish formal hypotheses regarding what you might find: 'the PVE method can be efficiently used to identify hypotheses from data sets without the need for establishing a hypothesis beforehand.'

2. Organise the data

   The guide recommends separating the dependent variables from the independent ones, and placing variables that are very closely related consecutively.

3. Correct the scaling and locate missing values

4. Identify and explain outlying values

   If using the tool to discover qualitative trends in typical data, it is important to remove the values that represent unusual cases. 'A valuable feature of PVE is that the specification of what constitutes an "outlier" does not need to be specified formally (such as "more than two standard deviations from the mean"). Rather, data points that appear to be visually far away from other data points can be selected for examination and analysis.'

[5]IBM Publication no. SC28-9742-00

5. Permute variables and look for relationships

   Look for variables that have high correlations—positive or negative.

6. Identify and explain categorical data

   In other words, find sets of data that appear to form a category—appearing on the display as 'bands' of lines either meeting an axis or crossing the space between axes.

7. Determine effect of different flow angles between consecutive variables

   This is just one particular hint for finding categories of data, by using the provided facilities for selecting groups of results according to the slope of flow between two axes.

The guidelines include some further practices that are suggested to be particularly useful in data sets containing clearly distinguished independent and dependent variables:

- Cluster identification—looking for separate clusters of lines that are closely correlated over a range of independent variables.
- Range queries on dependent variables, to discover which clusters tend to lead to which dependent-variable values.
- Identifying alternative independent variable values that arrive at the same dependent variable values.
- Inversion of some variable scales to make trade-offs more salient. Depending on the data set it might be useful to give all the dependent variables the same sense with respect to the user's goals (e.g., so good results are high on every scale), or conversely to invert the senses of alternate variables, since crossings can be easier to detect than regions of parallel flow.

WinViz

The WinViz Multidimensional Data Visualisation tool is now bundled as part of the Lotus 1–2–3 spreadsheet[6]. WinViz is also shipped as part of the Business Intelligence Suite, a set of data analysis tools marketed by Information Builders Inc. (IBI) of New York, NY.

The following information is taken from an overview available at the Singapore Information Technology Institute's World Wide Web site, and from two papers (Lee, Ong & Sodhi 1995a, 1995b)[7].

[6]Release 5 for Windows, initially for Asia/Pacific markets only; a product of Lotus Development Corp.
[7]Again, the tool's display makes use of colours for distinguishing results. Here they appear as grey.

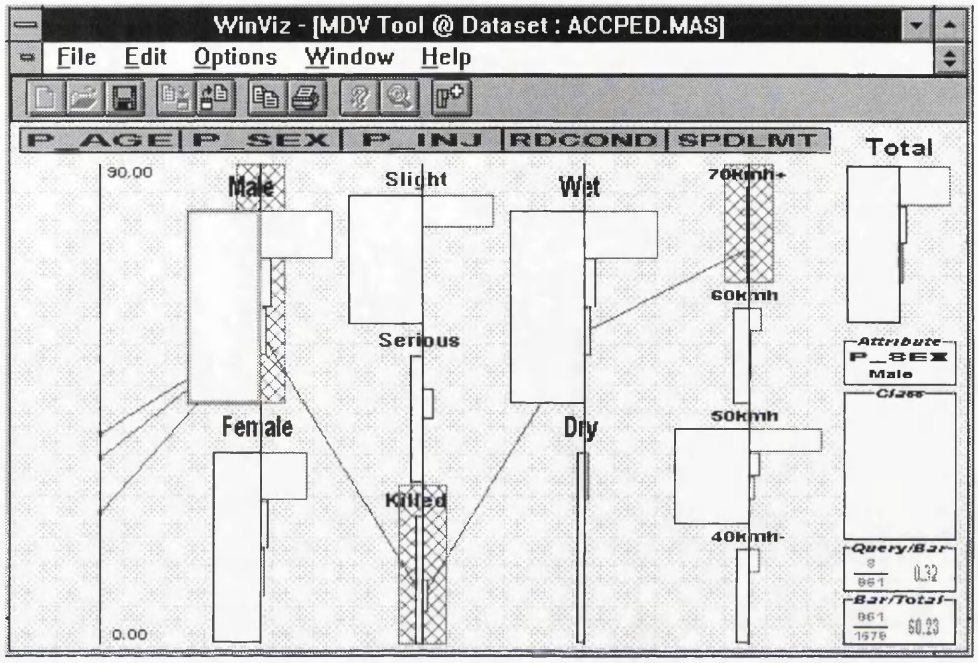


Figure 6.8 WinViz, showing a query on a set of accident statistics.

Figure 6.8 shows a set of 1579 records of traffic accidents involving pedestrians. In total there are seventeen attributes, of which only five are currently on display: the age and gender of the injured pedestrian, the extent of injury, the road condition (wet or dry) and the speed limit at the accident location.

Apart from the pedestrian age, the displayed dimensions are all discrete attributes with just a few values. If every result were plotted as a polygonal line the display would become very hard to decipher, so although WinViz does support such plotting its facilities are optimised instead for dealing with 'group bars' containing multiple results. The tool provides an interactive 'partitioning' feature for dividing a scale such as 'age', with an arbitrary level of detail, into a useful set of groups if wanted.

Each scale represents the distribution of the results among its discrete groups by the width of the group bars to the *left* of the centre-line of the axis. For example, the user is currently pointing to the 'Male' group bar, which can be seen to represent a slightly larger portion of the results than does the 'Female' bar. Accurate results relating to the bar currently under the mouse pointer are displayed at the far bottom-right of the picture, here showing the ratio of the number of results in the bar (951) to the total number of results (1579), expressed also as a percentage.

The bars to the *right* of each axis are for showing correlations between each group and a set of alternative 'classes' identified by the user. In this figure the user has identified the three alternative levels of pedestrian injury as the classes of interest, so every group has three bars on the right of the axis with widths showing the proportion of the overall data set that falls within each of the classes. It is apparent that the predominance of 'slight' injuries applies universally over the data set, although in this particular form of presentation the '70kmh+' bar is too narrow for us to make any visual judgement. One way to look in more detail at this part of the results is to create a query:

Three of the group bars in the figure are overlaid with cross-hatch boxes, having been selected by the user: 'Male', 'Killed' and '70kmh+'. Their selection constitutes a query, selecting only those results that have all the chosen properties. The box labelled Query/Bar near the bottom right-hand corner of the display shows that just eight out of the 951 'Male' victims were killed on high-speed roads.

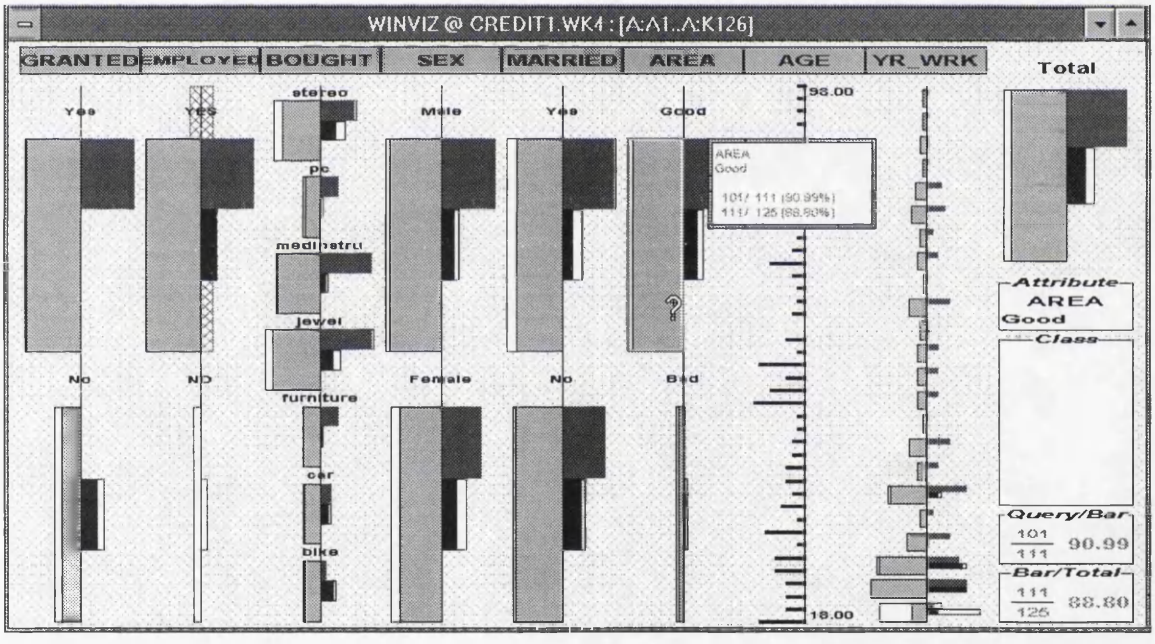


Figure 6.9 WinViz (for Lotus 1–2–3): A set of loan-decision statistics.

Figure 6.9 shows a more complex display but on a smaller set of results, relating to loan-granting decisions. In this case the specified query (results for which 'Employed' is 'Yes') constitutes the majority of the results. Each bar in the display shows the proportion of its results satisfying the current query as the shaded portion, while results not satisfying the query appear as the white strip at the outer edge of the bar. The classes selected for correlation are the 'Yes' and 'No' values of the 'Granted' scale. So, looking at the 'Sex'

scale, for example, we can see that a greater proportion of the female than male applicants were non-employed, and that all (or virtually all) the females who were not employed are in the 'not granted' category. The full query capability of WinViz is described in (Lee, Ong, Toh & Chan, 1995).

As well as supporting fully user-driven exploration, Lee, Ong and Sodhi (1995b) show the results of integrating WinViz with a tool for Knowledge Discovery in Databases[8] that derives generalisations from a data set and displays them on WinViz for evaluation by the user. For example, the program might develop the rule: 'IF P_AGE $<$ 41 THEN P_INJ = Slight [83.6%]'. This rule would be displayed using the P_AGE constraint as a query criterion, and highlighting the expected P_INJ result so the user could see at a glance which of the results did or did not conform to the generalisation.

The Attribute Explorer and Influence Explorer

An important alternative to the group bar concept used in WinViz—although not as well-suited to dealing with large data sets—is the 'stem-and-leaf' style of plotting used in the prototype Attribute Explorer (Tweedie, Spence, Williams & Bhogal, 1994) and the Influence Explorer (Tweedie, Spence, Dawkes & Su, 1995).

The stem-and-leaf plot is a well-known data presentation technique, in which—as described by Tufte (1990, p.46)—each data point simultaneously states its value and fills a space. Tufte shows an example of a train timetable:

```
hour      minutes
5         06 18 31 46 58
...
9         06 12 15 19 24 30 34 40 45 49 53 59
...
```

... in which it is clear at a glance that there are fewer trains in the hour following 5 a.m. than 9 a.m., before you even think of looking in detail at the precise times.

Figure 6.10 shows the Attribute Explorer, a system for data exploration (already described on page 58 in chapter 3). Each of the small house symbols that contributes to the histograms on the scales corresponds to an individual house instance, providing visualisation of trends and, when results are selected using the sliders, correlations between the scales. A symbol can also be selected individually, causing the corresponding data instance's polygonal line to appear joining its positions on each scale.

[8]The C4.5 inductive learning program (Quinlan, 1993).

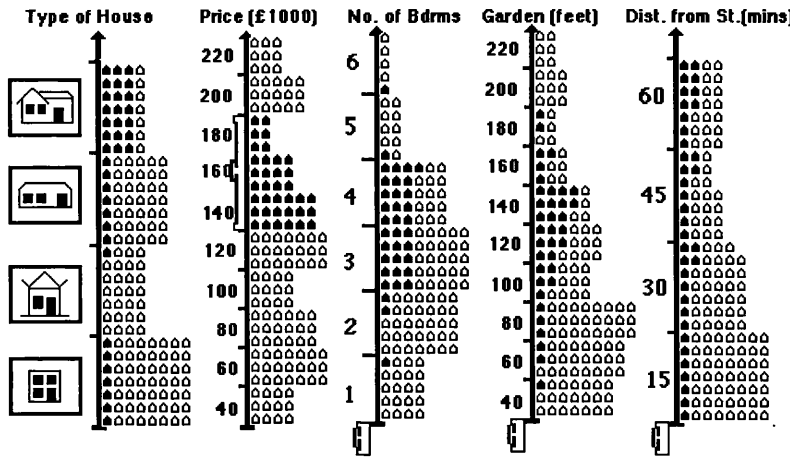


Figure 6.10 The Attribute Explorer (also shown on p.58).

The Influence Explorer, shown in figure 6.11, takes this style of display further. It includes rules for colouring the result blobs according to their satisfaction of multiple range criteria—with special salient colourings for those that satisfy *all* the criteria relating to the 'parameter' scales, or all those relating to 'performance' scales, or all of both sets. A detailed description of the meaning of the colourings seen here is given in (Tweedie, Spence, Dawkes & Su, 1995); a general explanation is not yet available.

6.3 Requesting and manipulating reconnaissance results

The foregoing review shows that the parallel coordinates presentation is a powerful formalism for displaying, manipulating and analysing multivariate data. But a reconnaissance system requires some specialised facilities beyond those needed for analysis of an existing body of data. In accordance with the needs identified in section 4.6 it is now shown how an interactive parallel-coordinates formalism might provide a substrate for supporting the following user tasks:

1. specifying a reconnaissance range, and opportunistic extension of the range
2. specifying the reconnaissance content—i.e., the measurements to be taken from each result—and opportunistic extension of the content
3. viewing and comparing result summaries to identify available trade-offs
4. selecting particular candidate results for detailed examination

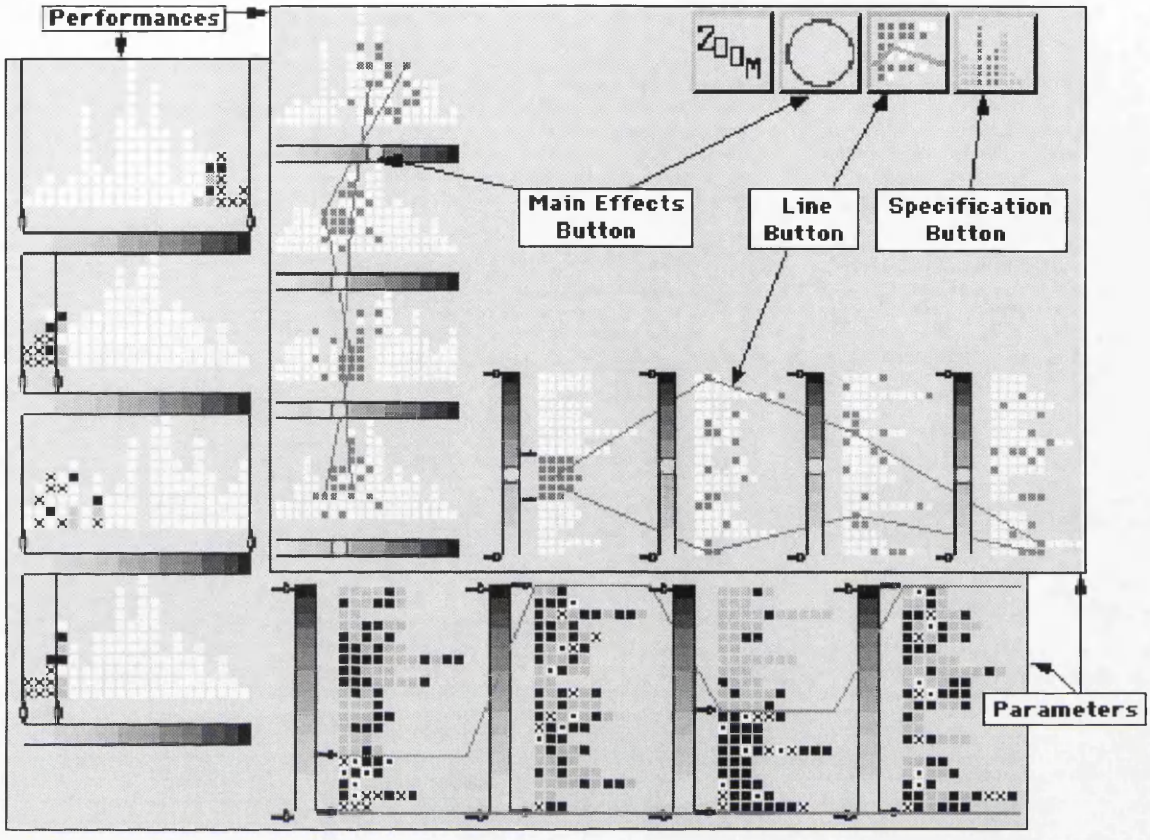


Figure 6.11 The Influence Explorer. This diagram shows two stages in the exploration; the larger (overlapped) window is the later stage, when the user has specified some parameter and performance criteria.

6.3.1 Specifying a reconnaissance range

Since the reconnaissance support is for use in a domain that can be explored in terms of option spaces, a reconnaissance range corresponds to a range of option-value combinations. In the domain represented in figure 5.13 on page 126 the user can choose values for each of the options A, B and C; one form of search that may be useful is an exhaustive permutation of alternative values for these parameters. Figure 6.12 represents a case in which a user has shown interest in three out of the four values for option A, both values for option B, five discrete points on the scale for option C, and two acceptable ranges for constraint D. Selecting these values from a menu, a 'switch' control, and two numerical ranges respectively can be achieved with standard dialog-box style controls; the selected values can then be mapped, as shown in the figure, onto independent scales that will appear in the parallel coordinates representation of the search.

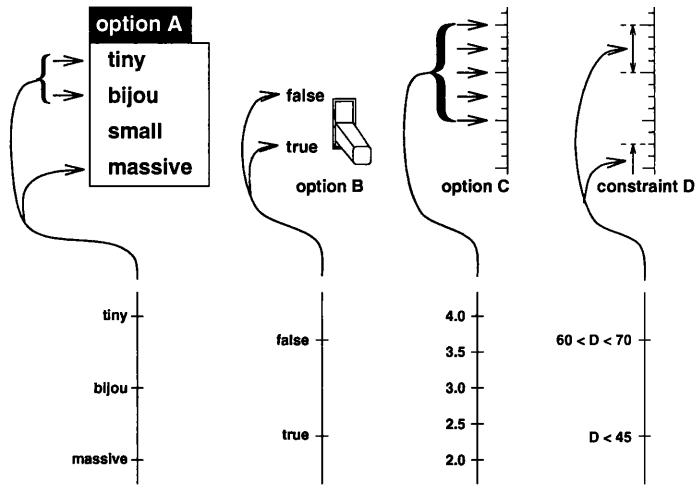


Figure 6.12 Representing option specifications on parallel axes

The most straightforward way for the system to undertake a reconnaissance foray using the selected values is to explore all combinations, in this case totalling $3 * 2 * 5 * 2 = 60$ alternatives. Alternatively the user can be given facilities for controlling the order and extent of evaluation, as suggested below.

Opportunistic extension of reconnaissance range

The user may be given various forms of control over the range of option-value combinations for which reconnaissance is obtained, such as:

- incremental search over the selected option values

  As described above, the initial bounds of the search may be specified by selecting a set of options to vary, and a hand-picked set of values to try for each option.

  The system can be designed so that it is easy for the user to request the reconnaissance necessary to illuminate *all* unexplored value combinations. Alternatively there may be facilities for the user to request partial searches, which in some domains would allow the avoidance of some needless exhaustive search in regions that can be discovered early on to be unproductive—e.g., when certain combinations of options A and B are found to be uninteresting, no matter what values might be specified for C and D. A simple way to do this is to allow the user temporarily to switch option values into or out of the search. For example, in figure 6.13 the user has switched off three in-between option values so that the reconnaissance illuminates just a broad selection of

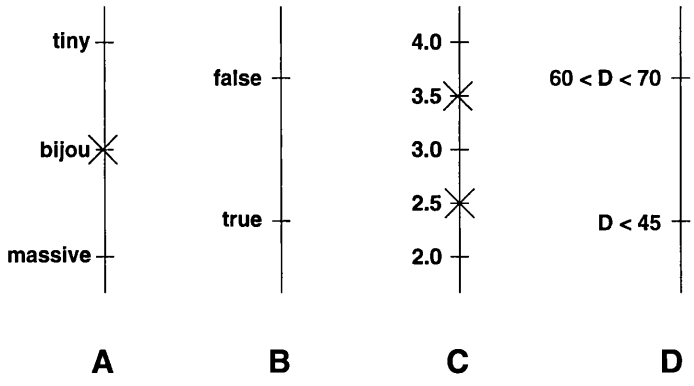


Figure 6.13 Temporarily switching off option values to reduce reconnaissance range

the option space; when the results are analysed they may be found to indicate that some of the switched-off values hold more promise than the others.

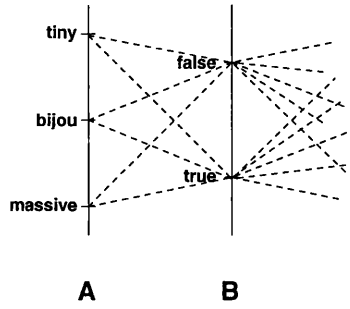
This example of reduced search examines $2*2*3*2 = 24$ of the total 60 combinations, so if processing each combination takes an appreciable amount of time the incremental reconnaissance strategy can reduce dramatically the delay in obtaining some results. It is up to the user to balance the time savings against the physical and cognitive effort of selecting options to switch into and out of the search, and the possibility that important results will be missed. Note, however, that the need to expend effort to *reduce* search thoroughness acts as a counterbalance to the normal effort–accuracy trade-off; the most thorough reconnaissance foray is the one that requires the least effort to request. This observation alone suggests an opportunity for a set of experimental investigations into how users prefer to handle reconnaissance under various conditions.

A more sophisticated way to refine the range of search would be to allow the user to express constraints on the combinations, such as specifying that only when option A has the value 'tiny' is the range '$D < 45$' of interest. Supporting such extensions would present further interface design challenges—but this is to be expected: after all, a fully expressive form of reconnaissance support would need the power of a fully expressive query language, and there are still many active research issues in the provision of visual-programming techniques for such languages.

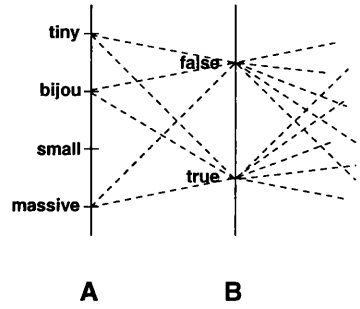
It should also be noted that the description here is influenced by the commitment, declared in chapter 2, to systems in which the user has explicit control over every aspect of progress. There are many opportunities for interesting extensions based on more adventurous forms of cooperation, such as reconnaissance refinement driven by

automated adaptive response to trends in the early results. Such extensions could be powerful, and presentable in a way that does not confuse users—but they are currently relegated to the domain of 'further work'.

- adding further values to existing option scales



before: reconnaissance completed



"small" added: reconnaissance has a new gap

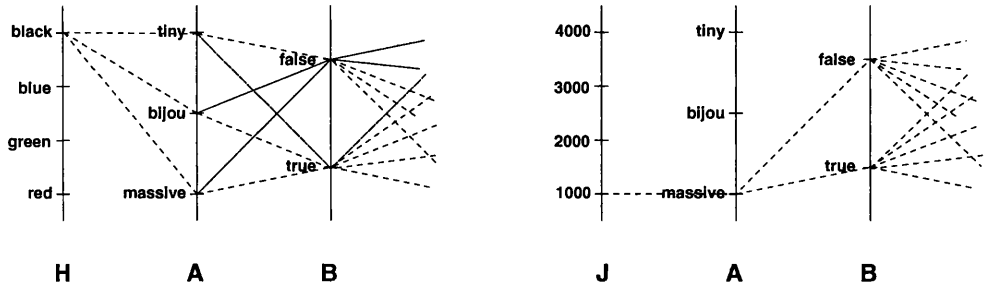
Figure 6.14 Adding a value to option A for further reconnaissance

Continuing with the same abstract search example, the user may decide to extend the reconnaissance beyond the three values of option A that were originally chosen for illumination. The addition of the previously ignored value 'small' does not affect any of the results illuminated so far, so the extra value can simply be added to the option A scale, as shown in figure 6.14. The value can be added in its normal place in the ordering of possible values, as shown in the figure; using the same ordering as the option menu is probably a good idea for helping the user understand the display.

The gap in the reconnaissance coverage that is opened by the addition of the value is of the same form as when a user switches in a value that had temporarily been removed from consideration. Facilities for requesting the reconnaissance necessary to fill the gap should be the same in both cases.

- adding further option scales

Having viewed early results the user may decide to take control over additional options in the search. In the case of a database retrieval this would correspond to constraining the value on some previously unmentioned parameters of the candidate results, such as the date of construction or the floor area of homes for sale. In a task such as document formatting the additional options would typically be customisations of parameters that are otherwise defined to have default values, such as the typeface or the inter-line spacing.



inclusion of previous default

partial overlap with previous values

Figure 6.15 Two examples of adding a new option to the search

Depending on the details of the exploration, adding a new factor might correspond to a subtle extension to the result-space region to be illuminated, or it may conversely correspond to moving the entire region. In the very worst case it would have the effect of making the previously discovered results irrelevant to the continuing direction of exploration. Going back to the castaway example: if the decision makers were suddenly to abandon the assumption that the groups was to settle in the new place, and decided instead that they should investigate striking camp somewhere where they could build boats in which to sail for home as soon as possible, any existing survey information would at least be evaluated with a radically altered set of priorities; some of the reports from scouts who had been sent inland would suddenly become uninteresting. By contrast, if the additional consideration were merely a particular interest in examining all hill-top sites, the reconnaissance may have already covered most of these places.

It should be left up to the user to determine what happens to existing results when a new scale is added, by making provision on the new scale for the display of the previous searches. Figure 6.15 shows the outcome of two different scale additions on the search as represented at the 'before' stage of figure 6.14. On the left, option H represents the addition of colour to a search in which the default assumption corresponds to the use of black. Since there is a 'black' node on the new scale, all the existing results are retained and can be plotted as passing through that point. On the right, the addition of option J marks a decision to specify in numerical terms the quantity that was previously represented in discrete categories by scale A. In this case the lowest value on the new scale corresponds to the value implied by 'massive', so the previous results corresponding to 'tiny' and 'bijou' no longer have any place in the display and have been removed. Scale A itself will have to be discarded to allow the larger values on J to be reported, but at least part of the earlier reconnaissance can be retained.

6.3.2 Specifying reconnaissance content

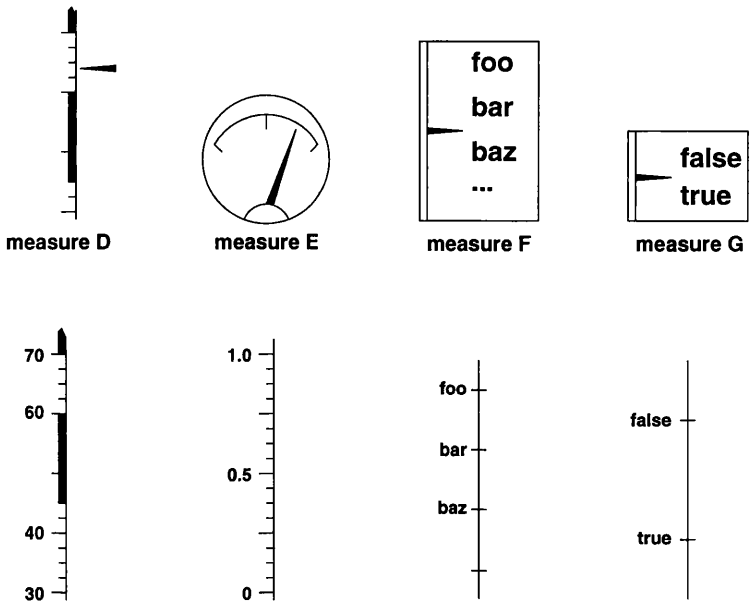


Figure 6.16 Representing measurements on parallel axes

As shown in figure 6.16, the various suggested kinds of measurement can be mapped onto scales for presentation in a parallel coordinates display. Many exploratory systems provide some standard result measurements, such as the total page count in a WYSIWYG word processor, or at least the facilities for requesting measurements, such as the area ratio of two rooms in a candidate floor plan. So an environment may already have the facilities for requesting the measurements that are of interest to the user; the requirement for reconnaissance is to let the user identify the measurements that are to be automatically derived, isolated and plotted for each candidate result.

An example of an interface facility for letting a user request reconnaissance measurements would be a suitable entry in a pop-up menu available on fields displaying measurement values.

Opportunistic extension of reconnaissance content

Pursuit of an exploration also requires that the user have facilities to change the scouts' instructions regarding the result measurements to be taken. Like the illumination facilities, support takes various forms:

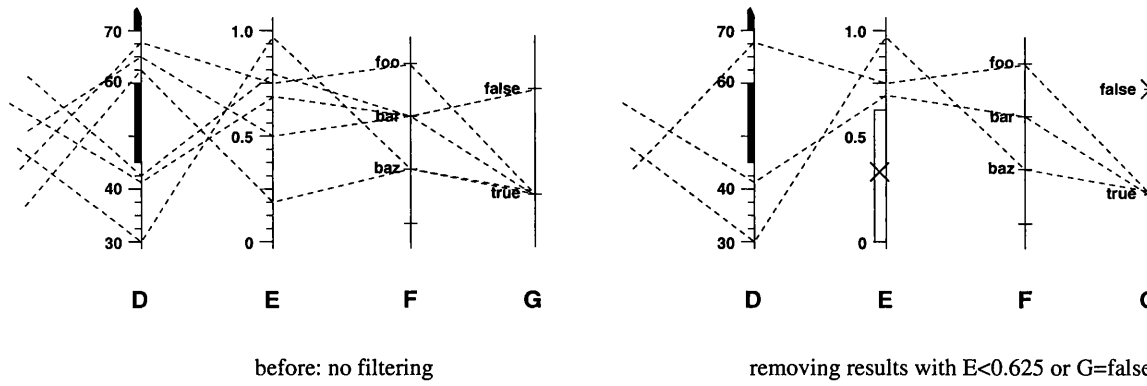


Figure 6.17 Using measurement values to exclude results from display

- result filtering by measurement value

  Although initially the user may specify measurements to be reported, without knowing whether or not they will provide useful evidence for discriminating between the value of available results, once some results have been seen the discriminatory power of some measurements may become apparent. In the castaways' scenario the decision makers may decide to remove from consideration any reports that score badly on these measures, and to instruct future scouting missions not to submit their reports for collation if they fail to meet the criteria. Note that because these are criteria based on the discovered results the scouting missions will still have to be sent to find those results—by contrast with a decision to reduce the range of options considered, as discussed in section 6.3.1, which allows the decision maker to cut down the number of scouting missions dispatched. Nonetheless, the ability to delegate to the scouts the decision on whether the reports are worthy for consideration can save a lot of evaluation effort on the decision makers' part.

  Figure 6.17 shows how this may be represented on the parallel coordinates display: having seen that several results attain the desirable high values on measure E, and the 'true' condition for G, the user has requested to remove from the display those results that score badly on E or fail on G.

- altering measurement sensitivity

  Another aspect of the discovery of the effectiveness of measurements is learning how precise a value it is worth asking the scouts to measure and report. Figure 6.18 shows two changes that a user might request: aggregating the range of measurement D into a few discrete categories (which are still independent of the constraints on D

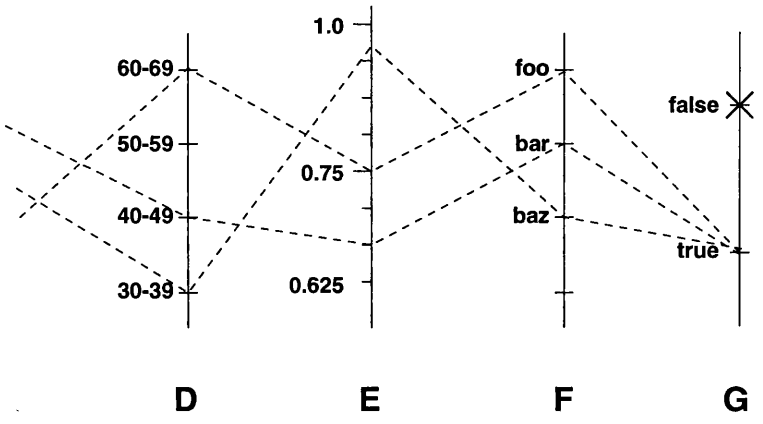


Figure 6.18 Changing the sensitivity of scales D and E

specified in the retrieval), and focussing scale E on just the reduced region of interest so that results' relative performance within this range can be compared more easily. Requesting the changes must automatically update the display of existing results, as well as affecting how future results will be plotted.

Scale F which, as defined in the previous chapter, represents a measurement that can take 'an unpredictable but limited range of values' (such as airlines offering a service on some route), must be capable of extending itself automatically as previously unseen values are found. Clearly this is only practical as long as there are no more possible alternative values than can fit comfortably onto a displayed scale.

- adding further measurement scales

  Like the addition of option scales, the decision to make additional measurements can have a dramatic impact on the usefulness of results that have already been obtained. In some cases the new measurements may already be available from the previous reconnaissance, merely needing to be expressed in the new way. In other cases the previous reports will not serve to provide the new measures, so either the user must manage without having those measures for all results or must dispatch reconnaissance to revisit the earlier sites and take the new measurements. For certain domains it would be feasible and appropriate for the system automatically to dispatch the necessary reconnaissance to fill such gaps; in others—especially where gathering reconnaissance is expensive in terms of resources—it would be better just to indicate the gaps but to make it easy for the user to give the necessary orders.

6.3.3 Trade-off analysis

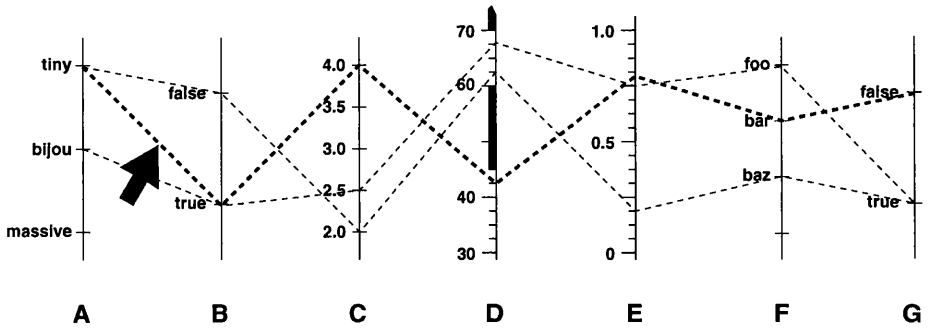


Figure 6.19 Three result summaries on parallel axes

Figure 6.19 shows a simple plot containing just three results, derived from different option combinations. The user can make some comparisons by eye from this plot, and can see correlations such as the apparent inverse relationship between factors C and D, but for detailed analysis some interactive assistance is bound to be necessary: for a start, the figure indicates a basic facility for highlighting a result by pointing to it (using the arrow pointer), which can help to resolve display ambiguities such as that caused by the convergence of two results on scale B. Additional interactive facilities such as those found in the systems reviewed in section 6.2.3 are likely to be useful: for example, the ability to highlight or colour selected ranges of results, or to reorder the scales to show alternative correlations.

The display shows the result-generation options in the same form as the result measurements. This suppression of the implementation-imposed distinction between options and measurements can be helpful, since they all count as properties of the results: for example, a user is likely to want to correlate flight cost against travel date, regardless of the fact that the date had to be supplied as an 'option' value while the cost is only available as a 'measurement'.

This is just one example of the (desirable) practice of providing *inter-referential input/output* (Draper, 1986)—also a factor in the provision of *equal opportunity* interfaces (Thimbleby, 1990). The overall aim, as discussed in section 9.3.2, is to avoid unproductive distinction between what constitutes an 'input' to a system and what constitutes an 'output'; in this case the inputs are acting as a useful part of the output.

6.3.4 Detailed examination of interesting results

Part of the practice of reconnaissance-based exploration is that the decision maker needs to examine in detail some of the results whose properties have been summarised by the scouts. Since it is by interaction with the parallel coordinates display that a user identifies which of the results are interesting enough to merit such detailed investigation, it is desirable that the facility for requesting the investigation is triggered through the same interface: for example, there might be a pop-up menu that allows the user to request the full details for the currently highlighted result.

As was noted in section 4.6, the generation of detailed result displays will usually need to be handled by a domain-specific display engine—for example, showing the full WYSIWYG view of a document whose structure has been summarised on the parallel coordinates display.

6.4 Conclusions

This chapter shows how the parallel coordinates presentation technique is suited to the handling of the information generated by reconnaissance that can be expressed using option space/measurement space dimensions. It has also outlined how one might build an interactive framework supporting the necessary opportunistic construction of a display that supports the core facilities required to perform reconnaissance.

The next chapter reports on the implementation of such a framework and the associated pieces needed to support a particular approach to document formatting, and the tests that were carried out to confirm that existing users of the formatting tools found the reconnaissance a powerful and useful addition.

Chapter 7

InteracTEX: Adding reconnaissance to LATEX processing

7.1 Introduction

This chapter documents the implementation and testing of a prototype interface based on reconnaissance, carried out to explore the feasibility and impact of casting an existing activity into the necessary form for reconnaissance to be applied.

There had been some discussion as to whether it would be better to use the time available on a single large-scale implementation exercise, or a number of smaller ones covering a variety of domains. It was decided that a single domain would be addressed; since the addition of reconnaissance raises questions and challenges at many levels, a single project stood a better chance of throwing light on a worthwhile range of these issues.

The fundamental need was to find a domain in which users would be involved in explorations, and for which under-informed outcome would be a recognisable problem. Further criteria used in selecting the domain were as follows:

- **A non-retrieval domain**

  As illustrated by the examples suggested so far, reconnaissance can be applied either in a domain such as database retrieval that explicitly deals with exploration, or one in which the exploration is only implicit. Having already explored some of the issues in providing reconnaissance in the travel agent (retrieval-based) scenario, the alternative represented a more interesting challenge and would show the wider applicability of the illumination-zone analysis.

- **Extending an existing industry-strength tool**

  In chapter 5 (particularly section 5.4) are discussed some design issues relating to contrived example domains of my own invention. When it came to implementing a test domain, however, there was a concern that any system I could build from scratch in the available time would necessarily support only a simple level of tasks. By contrast, finding an activity that was already supported with an 'industry-strength' tool, that could be extended to provide reconnaissance features, would offer the opportunity to apply reconnaissance to tasks of realistic complexity.

- **A domain with existing local practitioners**

  Working in a domain for which there was an existing local community of users would give the opportunity to canvass their opinions on the kind of reconnaissance support that would be appropriate, and to obtain feedback on the impact of reconnaissance, once implemented, on their typical modes of use.

Section 7.2 summarises the course of the development and evaluation of InteracTEX, showing the assistance obtained from existing experienced LATEX users. Then section 7.3 begins the analysis of what was discovered, starting with a brief overview of the features that made document-formatting in LATEX a suitable domain to address in this thesis; readers familiar with LATEX may wish to skip this section. Sections 7.4 to 7.6 then address the findings of the study, in terms of the various challenges posed by the reconnaissance proposals—namely:

- Challenge 1: Can reconnaissance serve a useful role in this domain?

  Section 7.4 examines this with respect to the requirements noted in section 4.5, namely:

  - Is there an opportunistic exploration to be performed?
  - Is the user motivated and able to specify a range of results to illuminate?
  - Can the user specify a set of system-measurable result properties?

- Challenge 2: Can a suitable reconnaissance interface be built?

  Section 7.5 reviews the requirements noted in section 4.6, using the interactive parallel coordinates approach proposed in chapter 6:

  - Can useful illumination be handled by defining an option space to be explored and a measurement space within which to place all the results?

  - Can the results be collated to allow the user to perform the kinds of trade-off analysis that are wanted? In addition, can the user view and compare a small subset of results in detail?
  - Can the system support exploration in the directions the user wishes to cover, collating results from disjoint parts of the result space if necessary?
  - Can the system provide convenient control over the progress of the exploration?

- Challenge 3: Does reconnaissance encourage accuracy?

  Section 7.6 reports on the test subjects' overall reactions to the addition of reconnaissance to a system with which they were already familiar:

  - What impact is there on the users' typical mode of use?
  - How is the extra effort involved in pursuing reconnaissance perceived in relation to the extra accuracy it can bring?
  - What further incidental changes did the addition of reconnaissance bring?

Finally, section 7.7 summarises the findings from this work, and the implications for future efforts to build systems with reconnaissance facilities.

7.2 Implementation and evaluation procedure

Figure 7.1 shows the three main stages of progress in developing and evaluating InteracTEX (also portraying their significance towards the three challenges declared above):

- The first stage was a feasibility study to confirm that LaTeX processing could be controlled satisfactorily, and aspects of its output instrumented, from a tool built in the Smalltalk development environment.

  This nascent reconnaissance framework was biassed to maximising the range of options and measurements that could be requested, by using the full expressive power of raw Smalltalk expressions. The use of Smalltalk syntax, which is not widely known, as well as the absence of convenient abstractions, meant that at this stage it was impractical to set up tests in which subjects could use the system for themselves.

  To obtain feedback without having to run such tests, the system was explained and demonstrated to two experienced LaTeX users—using a simple contrived formatting example. The subjects were canvassed for their opinions on the kinds of option and measurement they believed would be required to make up a useful 'toolkit' for general LaTeX use.

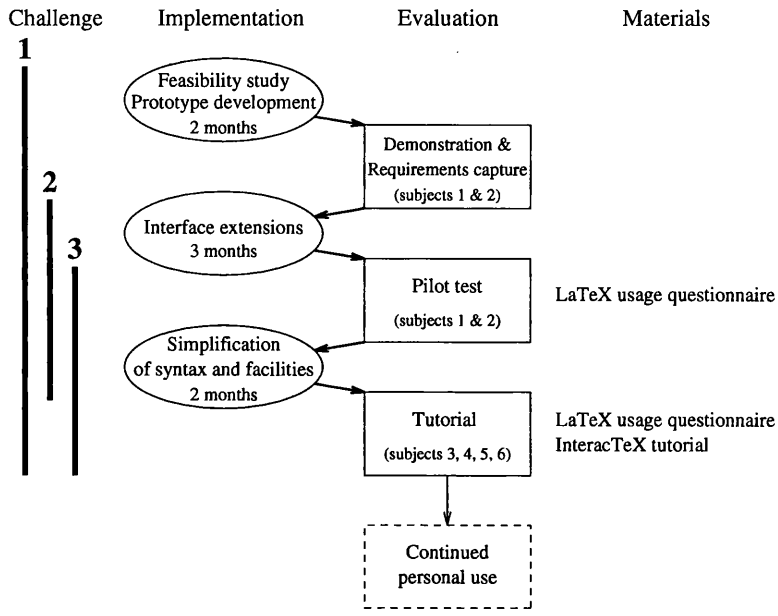


Figure 7.1 Stages in InteracTEX implementation and testing

- On the basis of the demonstration feedback the main period of implementation began, adding a basic level of facilities to enable the most important of the requirements that had been suggested for 'real' formatting tasks. Eventually the system, although it still required Smalltalk incantations that were too obscure to explain to test subjects, was sufficiently flexible to be demonstrated at work on the subjects' own documents. A pilot test was run, calling on the same two users as before. The test involved the following phases:

  A. A pre-arranged set of questions was put to the subject, to obtain a measure of his level of use and experience with LATEX.

  B. The subject was asked to call up one of his own previously formatted LATEX documents, and to show how, using his normal LATEX setup, he would go about re-formatting the document to meet some contrived but reasonably realistic new constraint—i.e., a dramatically altered page size.

     For this phase I had drawn up a second set of questions, that were fairly open-ended and aimed to provide a view of the subject's attitude towards the costs and benefits of producing well formatted documents. However, having encouraged the subjects to 'think aloud' while performing the task it was found that there was no need to ask all the questions formally: some were answered along the way.

C. In the absence of the subject, the InteracTEX framework was extended to include specific support for as many as possible of the processing options and measurements that the subject had been seen to use. This typically took a few days.

The subject was then shown the extended facilities, demonstrating how reconnaissance would offer a different way of going about the same formatting task, and was prompted to answer a third set of questions regarding the perceived effectiveness of the approach.

The sets of questions for which I wanted answers are in appendix B. Some were answered in the subjects' thinking aloud, while others had to be posed directly.

Basing the demonstration on one of the subject's own documents, and a task that had already been attempted without the help of reconnaissance, had the desired effect of impressing the subjects with the potential power of the technique. They were able to suggest further facilities that they believed would be useful. But both subjects stressed that for any serious analysis of the impact of reconnaissance they would have to be able to drive the system for themselves; that the top priority in further development should be a dramatic simplification of the interface to make this possible.

The results from phases A and B contribute towards the overall evaluation of challenge 1: the existence of a role for reconnaissance in LaTeX use. These results are therefore reported in detail in section 7.4 below. The phase-C results, on the other hand, are found in section 7.6 dealing with challenge 3: the overall impact of reconnaissance on result quality.

- The feedback from the pilot test was sufficient to guide an extensive re-design of the system, to allow the most commonly used facilities to be made much easier to specify. This made it feasible to develop a hands-on tutorial based on a fairly complex formatting scenario, introducing all the main capabilities of InteracTEX so subjects could reach an understanding of what it might do for their own work.

  Four further experienced users of LaTeX were approached. As in the pilot-test stage, each subject was first asked to demonstrate the facilities he or she typically used in LaTeX, and to answer the questions from phases A and B above. Then the tutorial was worked through in 'think aloud' fashion, followed by a further solicitation of comments in similar fashion to the use of the phase C questions.

  There was a lot of material to cover. Subjects typically spent 3.5 hours in total on the evaluation, split into two sessions to prevent excessive fatigue and to allow the novel ideas to 'sink in' a little. Again, the results from this section are split between sections 7.4 and 7.6 as appropriate.

Calling the test participants 'subjects' perhaps fails to give them sufficient credit for the contribution they were being asked to make: these people were assisting in the iterative design of a complex extension to a rich existing environment. The fact that they were all experienced users of LaTeX ensured that they were familiar with the basic task being supported, and—as it turned out—had their own catalogues of typical formatting problems. As well as validating the comprehensibility and usefulness of the implemented InteracTeX facilities, these subjects were therefore able to provide additional ideas on further development directions for reconnaissance support in this domain.

The facilities described in section 7.5 are from the final stage of development, as used in the tutorial.

7.3 Suitability of LaTeX for addition of reconnaissance

7.3.1 The need for result-space exploration in working with LaTeX

To help clarify why the use of LaTeX is more like an opportunistic exploration than an imperative definition of the desired output, I include here my interpretation of the main points of 'LaTeX philosophy'. The overriding principle is 'leave typographic design to the experts', so readers who are attuned to a more hands-on, WYSIWYG approach to document preparation might find the following a little disconcerting. But there is certainly room for debate; it is not clear what proportion of the LaTeX community is actually prepared to adhere to the letter of the law in pursuit of officially sanctioned formatting.

Leave typographic design to the experts

LaTeX is a software tool for allowing authors with little or no skill in typography to produce documents formatted in accordance with many of the complex and stringent typographic conventions that have developed over centuries of publishing. Some of these conventions are concerned with minute details of placement of each character within a line of text, while others relate to higher-level issues such as the formatting of paragraphs, sections and chapters.

The author of a document is saved from having to consider or even understand such details, by working at the level of *logical design*. In the case of a bulleted list, for example, the author merely needs to declare a set of items to be members of the same list, and the system will produce a list that conforms to design standards for such lists as previously defined and expressed by an expert typographic designer.

The alternative—tools that give authors direct control over the *visual design* of their documents—is claimed to cause many problems when the authors are untrained in typography. Such systems may allow an author to produce a document that appears visually pleasing, but this is not enough:

> 'Most authors mistakenly believe that typographic design is primarily a question of aesthetics—if the document looks good from an artistic viewpoint, then it is well designed.'
>
> (Lamport 1985, p.6)

Good typographic conventions, Lamport argues, have evolved to make a document as easy as possible for a reader to understand. Naively designed documents, even if visually appealing, are likely to contain many typographic mistakes that cumulatively burden the reader; even if the reader can discern the meaning, the effort required to do so will be needlessly greater than in the well-formatted case.

The system will need help in making design trade-offs

Every line of text produced by LaTeX is proportioned and laid out to the highest possible standards—and if those standards cannot be met a warning message will be produced. But the conventions it embodies are only those that can be expressed as mechanistic generalisations, so unlike a human typesetter (and in common with all other computer-based text formatters, as far as I am aware) LaTeX cannot make higher-level decisions based on the meaning and intent of the material being formatted. For example, if a list does not fit onto a single page, LaTeX cannot detect whether it would be less disruptive to a human reader for the page break to occur between a given pair of items rather than one item earlier or later.

For this reason the user sometimes needs to provide hints to LaTeX as to how to resolve problems that might arise. LaTeX includes various commands for coercing the formatting algorithms to give priority to constraints that are additional to those implied by the document's logical structure. These can be used, for example, to encourage or force line and page breaks to occur in places that the author feels would be the most appropriate.

If used heavy-handedly, these coercions can give the author excessive control of the visual design of the document—potentially sabotaging some of the efforts of the typesetting algorithms. Used appropriately, however, a beneficial synergy of subjective judgement and typographic precision can result.

Create first, polish later

Lamport quite reasonably claims that:

> 'LaTeX was designed to free you from formatting concerns, allowing you to concentrate on writing. If, while writing, you spend a lot of time worrying about form, you are probably misusing LaTeX.'
>
> (p. 8),

...and he later goes further, suggesting:

> 'Don't worry about line and page breaking until you're ready to prepare the final version. Most of the bad breaks that appear in early drafts will disappear as you change the text.'
>
> (p.87)

...although here it is not clear whether he is claiming that the bad breaks tend to 'disappear' because of changes you were going to make anyway, or simply reminding you to get used to the idea of changing your text to make this happen.

7.3.2 An extensible, industry-strength system

Some software systems are much easier to extend than others; with the programming tools at my disposal I needed to find a system that had some form of powerful external programming interface.

Unlike the highly self-contained text formatting programs available on the department's Macintosh computers, LaTeX offered great promise of extensibility. The key to this is that the TeX processing engine[1] is controlled entirely by the contents of files that the user instructs it to consult, and that since these are all plain text files they are easily editable. Figure 7.2 shows the standard LaTeX setup.

[1]Note: LaTeX is not a separate processing tool, but a simplifying layer of commands defined using the TeX language. As far as TeX is concerned, a document that uses LaTeX is simply calling on the `latex` macro file as well as any others it requires.

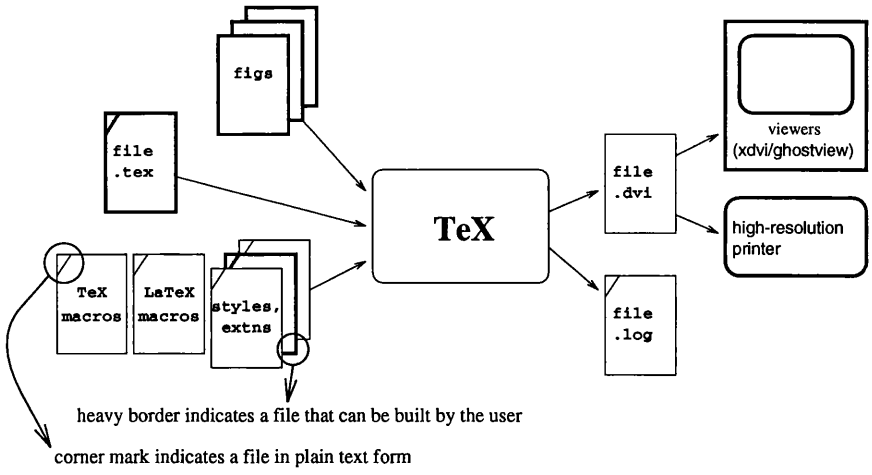


Figure 7.2 Components of a TeX/LaTeX processing setup

Specifying alternative input options

In normal use of LaTeX, all the user's text and formatting instructions for a given document are contained within a single file (the `.tex` file). This file can contain all of the following kinds of instruction to the TeX engine:

- the document text, including its structure information (headings, lists, etc.);
- instructions to pull in additional files containing other elements of the document, such as figures, that have been prepared independently;
- instructions to pull in files that specify the formatting style to be used;
- instructions to set values of parameters used by the selected style;
- instructions to override macros defined by the style.

The valuable implication of this is that the full range of customisation that a user will typically use can be offered simply by controlling the content of this one file.

Taking measurements of the output

The main result of processing a document is the `.dvi` file, containing typeset pages in a 'device-independent' format that can then be interpreted by further tools to drive printers or

online viewers. In addition TeX produces a log file (`.log`) containing diagnostic information regarding the processing of the document. In normal use the main purpose of this file is to help the user determine the cause of any errors or warnings that may have arisen; when there are no errors the file contains little information. However, there are various adjustable settings (set using instructions within the document, as usual) that allow a user to cause additional information to be logged, such as detailed tracing of the actions of the line- and page-breaking algorithms. In addition the user can cause messages to be written to the log by embedding special macros in the input; these macros can refer to variables that are made available by TeX, and that reflect the position and state of the typesetting at the instant when the surrounding text is being processed.

Having these logging facilities seemed likely to be all I would need to provide rich measurement facilities. In fact, as reported in section 7.5, there were some unexpected limitations in the power of both the range and measurement specifications.

Providing a reconnaissance-control front end

Although there are some front-end systems that are specialised to the creation and editing of LaTeX documents, our department's Unix installation of TeX does not include any. I was therefore free to create my own front-end to the processing engine, which I did—as usual—using Smalltalk. As well as being a general-purpose application development environment, Smalltalk has all the facilities necessary for launching and communicating with other processes such as TeX.

7.3.3 Availability of subjects who understand the domain

In the department of Computing Science there is a large community of users of LaTeX, with varying levels of experience and differing principal reasons for using the system—such as:

- the opinion that its typesetting is of higher quality than that of the alternative tools available here;
- a need to typeset large amounts of mathematics, for which LaTeX has particularly powerful facilities;
- a frequent need to conform to explicit style guidelines, e.g., for conferences, which are often expressed in the form of a LaTeX style.

I was already a LaTeX user myself, but having access to people with much more experience of the system was of great assistance throughout the development of InteracTeX.

7.4 Challenge 1: Can reconnaissance serve a useful role?

The first task in the investigation was to seek confirmation that users of LaTeX typically engage in opportunistic exploration of a result space, and that there is a risk of under-informed outcome—i.e., that users are aware that greater accuracy might be obtained through greater effort, but are not motivated to be thorough. In addition I had to confirm that, as defined in section 4.5, the user is (a) motivated and able to specify a range of results to illuminate before making a choice, and (b) able to specify a set of result properties that could be measured to provide informative summaries of the results.

7.4.1 Is there an opportunistic exploration to be performed?

To supplement my own experience with LaTeX, all test subjects were interviewed about their patterns of using the system. As described in section 7.2, they were also asked to show how they would carry out a sample formatting task on one of their own documents. They were asked to select a document that was typical of their use of LaTeX and whose formatting fitted, as far as possible, the following criteria that affect the level of likely difficulty in producing good formatting:

- **too large for the formatting to be predictable**

  Around 20 pages was found to be sufficient to make it impractical to predict the impact of any significant change throughout the document.

- **both subjective and formalisable result properties**

  The need here was for a mix of structures in the document—perhaps a few figures, lists, or formulae amongst the text—firstly to ensure that there would be some document-specific formatting issues rather than just the perennial concerns of widows and orphans, and secondly to make it clear that the reconnaissance was not aiming to reveal all details of the processing but just those that could be expressed with the available measurement facilities.

- **non-formalisable freedom of variation**

  The subject would need to think up aspects that could be varied to fix formatting problems. To motivate this, the sample document should be for the consumption of people other than its author.

The test procedure used for each of the subjects was as follows:

1. First the subjects were asked to report on the main issues that had arisen in the formatting of the document, and how these had been handled. They were also asked to comment on how satisfied they were with the formatting as it stood.
2. For any aspects with which they felt dissatisfied, they were asked to explain what had led them, or forced them, not to find fixes to these problems.
3. Then a simple but significant change was applied to the task goals: a substantial change in the size of printing area available on each page. This alters the page-break positions TEX will select throughout the document, and was intended to return the pagination to the kind of arbitrary state users are likely to find when they first seriously address a document's formatting.
4. The subjects were asked to format the document with the new page size, but no other alteration to the input, and to comment on what—if anything—they now felt needed to be fixed.
5. They were then asked to make the fixes they felt were necessary, until the point at which they were satisfied with the outcome.

The subjects were also asked to comment on the extent to which this sample of activity was representative of their normal use of LATEX. Most of the subjects identified strongly with the test scenario, having experienced the anguish of completing the formatting of the 'final version' of a document only to be hit later by a significant and unavoidable imposed change to its style or content. These experiences normally related to conference papers.

One subject noted that because the test scenario did not have a realistically high external motivation he was not as motivated as normal to find the best possible way of presenting his document (a conference paper). In particular, when he has to squeeze a paper to contain only the most important points in the limited available space he normally uses a strong mental picture of the paper's topics to decide how the text can be expanded or contracted to best express these points. So whereas at one point in the test he adjusted the layout simply by adding a blank line to an example code segment, he would normally have wanted to use that precious space for another line of terse explanation. He still insisted on fixing the problems he found, but using the approach that he would normally apply to a less important form of document.

The following sections describe some of the characteristics that were observed to lead to a need for opportunistic exploration.

User cannot predict and pre-empt all potential result problems

The complexity of the typesetting task makes it very difficult for a human to predict with any accuracy how a bulk of information will turn out, once laid out on the page in accordance with good typographic conventions. This is part of the reason for Lamport's exhortation not to worry about layout while creating a document's content: it takes time and energy, and most of the places where a problem *might* arise will turn out to be problem-free anyway.

In standard implementations of TEX, a document is prepared in a *source file* that contains both the text and the formatting instructions. The intended result is a document printed on a very high resolution printer. To get an accurate representation of how the document will appear once typeset the user must run the TEX processing engine on the source file, although it is not usually necessary to go to the extent of printing each attempt on paper: a representation of typeset text and pictures can be viewed in some detail on a graphical computer display, allowing some text processing tools to provide WYSIWYG displays of the formatted output.

However, whether or not a tool offers incremental re-formatting of text as it is changed, the effect of even a small change to the input can have unpredictable impact for the following kinds of reason:

- **sensitive algorithms**

  There are many hidden trade-offs involved in TEX's calculations of where to break lines and pages. The addition of a single character may tip the balance determining where to break a line, and hence perhaps a page. Some of the test subjects noted that they find LATEX's placement of 'floating' items such as figures especially brittle.

- **style changes**

  Any change to a style parameter that is used in many places in the document, such as the indention at the start of a paragraph, has a multiplied chance of causing the kinds of disruption described above.

- **knock-on effects**

  Clearly a change that has any significant effect on local formatting may also have knock-on effects in any later portion of the document. In some cases changes can affect earlier parts of a document, too—for example, in a document that includes cross-referencing to named sections or figures a change to one of these names will affect all places where it is referenced.

Typical formatting problems that the subjects reported experiencing (and considered important to try fixing) include the following:

- (Subject 2) exceeding a publisher-imposed page count
- (S1, S2) code examples and figures not on the same page as their references
- (S1, S2) code examples that were broken across pages, in inconvenient places
- (S5) proofs broken across pages, although they are often long (and their position in the text is fixed) so it's hard to see how they could be improved
- (S5) some pages had large white-space gaps at bottom
- (S5) some pages were sparsely filled, because of default additions around list items and examples; would have been wanted to compress them a little
- (S5) bad line breaks within the formatting of the bibliography

Problems may require indirect, user-chosen modifications

Because of the goal of leaving as many as possible of the formatting decisions to LaTeX, most suggestions the user can make should be specified as hints rather than commands.

For example, if the author is especially concerned at finding that a paragraph of just a few lines is broken across a page boundary, the recommended approach is not to force a page break just before the offending paragraph but to mark the text as an important block *not* to break. When TeX comes to the point at which it weighs up the decision on where to end the page in question, the author's strong hint may tip the balance in favour of a different layout: TeX may be able to squash or expand the page to include or exclude the whole paragraph, or there may be some more radical alternative such as changing the position of some nearby 'floating' item such as a figure.

If the hint fails to have any impact, the author has to decide whether to take responsibility for the visual layout by using one of LaTeX's firm commands, or to make some other attempt to appease the formatter's punctiliousness by changing the order or content of the information to be typeset.

An alternative the author *might* consider is changing some global parameter. For example, if the document has the problem that it is taking up a little too much space—just spilling over onto a part-filled page, or perhaps filling half a page in what will be a double-page

spread—the user may be prepared to try using a smaller font or wider margins throughout the document. Of course, many such adjustments would go against the spirit of LaTeX. As Subject 1 put it: 'The whole point of LaTeX is that we [as document writers] are naive about formatting; we don't know as much as the providers of styles.'

The subjects' opinions on suitable modifications included the following:

- (S2) reluctant to override any features of the externally-imposed style, although felt that subtly adjusting the spacing between major sections would probably be acceptable
- (S1) finding problems relating to the use of someone else's macro, was reluctant to interfere with it
- (S1) considered turning code examples into figures, so they float and therefore don't get broken across pages (but raising the issue of becoming separated from their descriptions). Opinion: 'floating figures tend to turn up in the right sort of place'
- (S2) figure placement could sometimes be fixed by moving the figure's declaration relative to the text
- (S5) faced with a short list (5 items) split across pages, might try forcing a page break before it or finding earlier text to expand or contract
- (S5) faced with a widowed list item, would try removing some text from some previous section where it looked like room could be created—possibly shortening the abstract four pages previously!

Problems cannot necessarily be solved in isolation

The changes an author makes to fix one problem are likely to have effects on other parts of the document. This is, of course, always true in the case of a global change, but even a local change can have knock-on effects of unpredictable magnitude depending on the degree to which the area around the 'fix' can absorb the change.

It is therefore true that a change in one place may cause some other problem to disappear—and perhaps some new problems to emerge. This can be especially dramatic for global changes; in a small document it is sometimes the case that some combination of global settings will produce formatted output that is almost or completely free of problems. But how likely is the author to find that combination, even if it exists?

The approach of making only local changes, and working through the document strictly in sequence—re-formatting after each change, to see where the next fix is needed—will certainly clear up problems. But it is still worth staying in the realm of 'hints', rather than explicit commands, in case there is a change imposed in the future—such as the need to remove a section—that affects a large section of the document. An example of the kind of problem to avoid is the addition of explicit page-breaks, which can result in unexpected breaks part-way through pages if there is a later change.

Subject 2, having made various changes to fix some problems, then found on a later page a problem that he perceived as more important than the others. He therefore undid all the previous changes, and made different ones that avoided the side effect that had caused the particularly bad problem.

Trying large numbers of alternatives is expensive

Each change to the document needs to be specified, submitted for processing, and the results viewed. Even if each iteration takes only a few seconds the author is unlikely to have the patience to try more than a few dozen attempts altogether.

Subject 5 noted that if an experimental change doesn't make the required difference she usually wants to restore the document to its previous state. This makes experimentation expensive.

7.4.2 Is the user motivated and able to specify an illumination range?

The options the user may be prepared to vary, and the acceptable ranges for these options, are likely to be specific to the document and could not be predicted by a tool's designer. For example, in some cases the user has great freedom in setting style parameters such as line spacing, font size, margin width, etc., but cannot alter the content; in other cases the style may be tightly constrained by an external requirement, but the user is prepared to change the order or even content of the document to produce what the user judges to be a preferable overall outcome.

Many of the adjustments the subjects commonly used were capable of being supplied mechanically, rather than the user having to make every change by hand:

- **Standard global formatting choices** such as picking a font name, size, and document style to be used for the document as a whole.

- **Ubiquitous presentation choices** such as turning on or off individually a forced page-break before any of the major sections, or a frame box that can be added to each figure.

- **Discrete steps in a numerical value** such as line spacing, margin sizes, tab positions, scaling values for a figure.

  (S5) had spent many hours on formatting, dominated by maths alignment: trying different line break positions, tab positions, font size (within a single proof).

  None of the subjects in my sample were in the habit of adjusting the parameters that are used to fine-tune LaTeX's various control algorithms. But some of the formatting options offered by LaTeX, such as `\linebreak[`n`]` and `\pagebreak[`n`]`, and `\sloppy` and its partner `\fussy`, are implemented simply as coarse variations in penalty and threshold values. There is no reason why an author should not create half-way versions of these commands using explicit numerical values, and use InteracTeX to try many different values to discover when the balance of the formatting alternatives changes. Depending on the details of the calculation being performed at the particular point being adjusted, such changes may sometimes be swamped by other factors. But in some cases it may turn out that the formatting is sensitive to subtle changes in the weighting, so the author will discover a number of additional potential outcomes.

- **Alternative placement of text fragments** where the fragment might be some text or a formatting command, thus allowing simple reordering of items (entire chapters, if wanted), or alternative places to break a line of mathematics, etc.

  (S2) experimented with moving a single-line paragraph from the start of one section to the end of an earlier one.

- **Specialised alternative versions of a given text fragment** when the user does not mind which of the versions is used, but is interested in whether a desirable knock-on effect is induced. Or wanting to retain the original form of a sentence that has been savagely cut to fit—in case the shortening turns out not to be worthwhile.

7.4.3 Can the user specify a set of system-measurable result properties?

What concerns the user regarding the output will depend on the content of the document, as well as its structure. For example, although a formatting tool might generally discourage page-breaks from occurring where they would maroon a single line of a paragraph or item in a list, the level of the user's concern is just as likely to relate to juxtaposition of concepts. A computer-based tool, with no ability to 'understand' these issues, might at least allow

the user to provide hints—overlaid on the document structure—for example specifying that page-breaks would be highly undesirable in certain regions.

The following are some categories of features in the formatted output that commonly caused concern for the test subjects, and that could easily be measured by the system rather than requiring human inspection.

- **Position of page breaks relative to particular document regions** such as whether sections that the user wants to be continuous (e.g., code examples, lists, short paragraphs) are broken across pages, or whether points that ought to be seen together (e.g., a figure and some text that refers to it) are separated by having to turn the page.

  (S5, S6) it would be good to have some more standard measurements, e.g. start and end pages (or number of pages) of each major section

- **Position of salient document elements relative to page breaks** such as noting when a major section heading appears near the bottom of a page, or that a section ends near the top of a page thus leaving a large (and measurable) area of unused space.

  (S5) it would be good to have standard measurements checking the various ways in which the positioning of, say, a single-line equation falls at a bad point

- **Degree of 'badness' of typesetting warnings** so that rather than having to look through all warnings that may be reported, or to adjust the sensitivity of the warning generator, the user can see a summary of just how bad the warnings are.

7.5 Challenge 2: Can a suitable reconnaissance interface be built?

Having obtained a large amount of evidence pointing to a role for reconnaissance in document formatting, I was faced with the challenge of implementing the necessary software components to provide reconnaissance support. As defined in section 4.6, these are:

- a result-space illuminator that can accept the kind of range specifications that the user is prepared to supply, and can make the kinds of measurement requested in the content specification;

- a display generator and interactive display that can present a collated summary of the results to allow the user to evaluate them, and in particular to make tradeoff judgements between them;
- a command processor to handle the user's requests for illumination, summary-display manipulation and access to detailed displays, and to maintain an illumination zone that can hold however many results the user wishes to illuminate and to evaluate alongside each other.

7.5.1 Specifying a reconnaissance range

Principal requirements

As reported in section 7.4.2, the initial studies of LaTeX usage revealed that it would be useful to support the following kinds of option:

1. overall document style choices, such as font size;
2. alternative, equally acceptable wordings for parts of the text, e.g., to change the size of a paragraph if it will help the overall layout;
3. adjustment of numerical values used to specify figure sizes, tab positions, etc.;
4. optional characteristics of individual document elements, such as the constraints on placement of each 'floating' item (e.g., figures), or whether a given section of example code may be split by a page break;
5. alternative orderings for sections of text, e.g., the items in a list or even sections in a chapter;
6. correlated changes in different parts of the text.

To be represented as a dimension in an option space, each kind of option must be controllable by selecting a single value out of many (as if from a menu of alternatives). The principal characteristic of LaTeX that makes this easy to achieve is that all the text and formatting commands for a document are specified in a single source file in a plain ASCII format. This allows alterations to be specified merely by changing some sections of the character sequence that makes up the file. Figure 7.3 shows how a facility for allowing specified places in the file to take one from a menu of alternative character strings is sufficient to handle the first four requirements in the above list, and figure 7.4 shows how the other requirements can also be controlled by selection from menus.

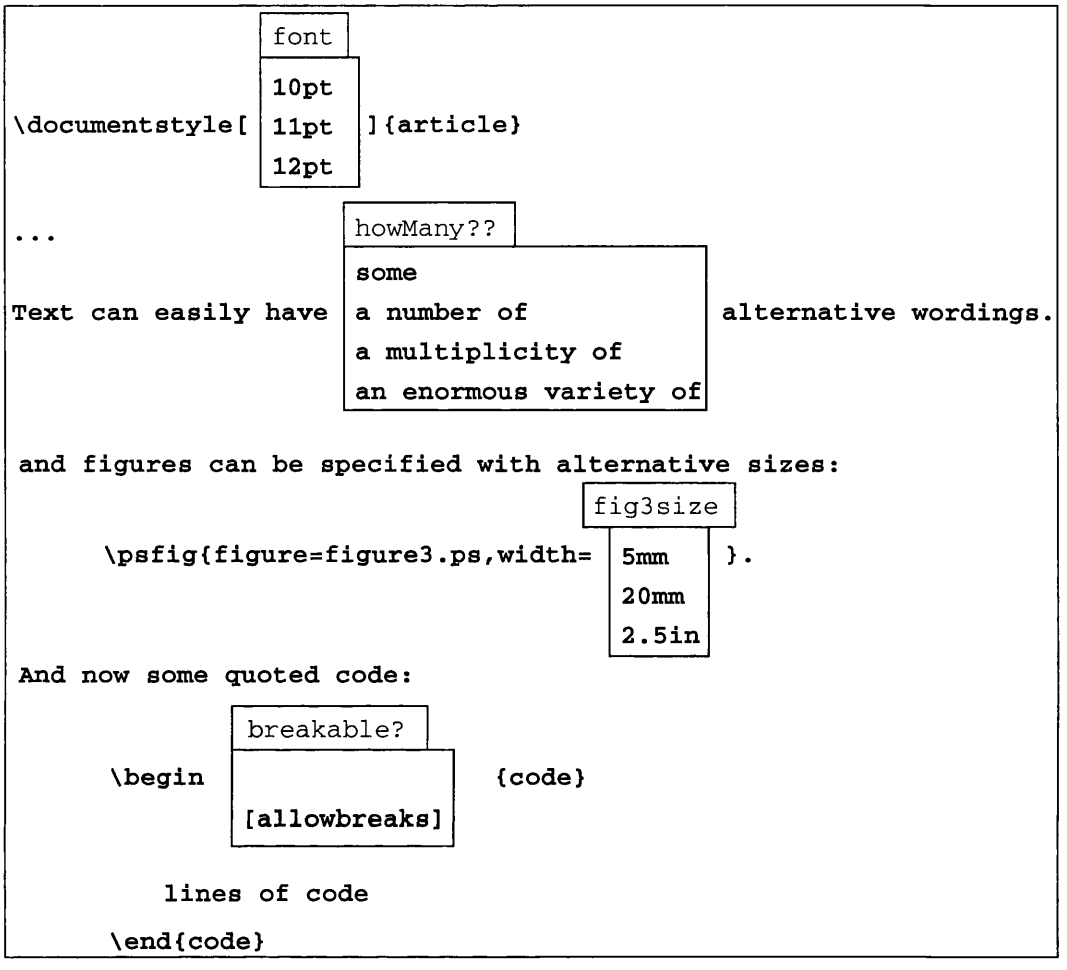


Figure 7.3 Document variation by insertion of alternative strings. (In this and subsequent examples, the strings that are destined to end up in the LaTeX file are shown in the heavier font.)

Facilities provided in InteracTeX

Since LaTeX users are accustomed to specifying the content and formatting of their documents as markup in a textual source file it seemed logical and convenient for the InteracTeX facilities to be specified in the same way. However, to allow the maximum flexibility in varying any part of the markup the InteracTeX specifications are handled using an independent syntax that is interpreted and stripped out in a pre-processing stage, the outcome of which is a standard LaTeX markup file. In addition, since some instructions to InteracTeX do not have any obvious place within the main document markup the user can place them in a separate 'expression' file.

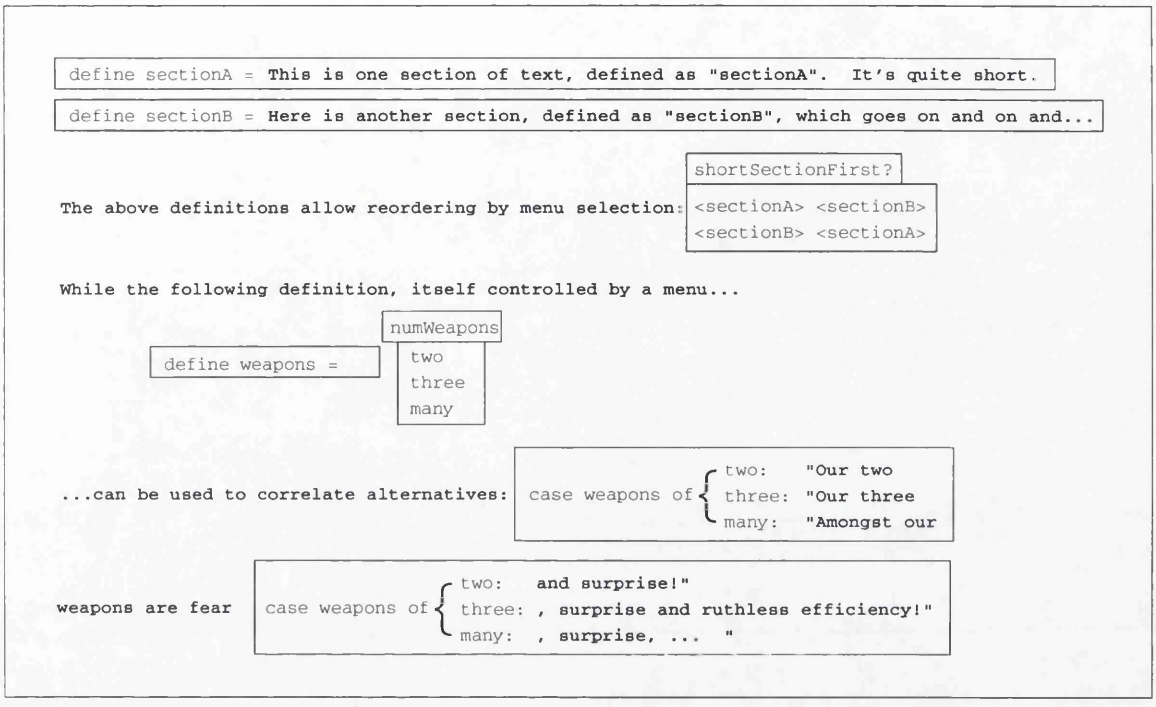
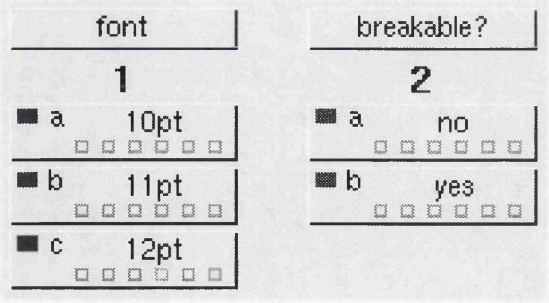


Figure 7.4 Document variation by definition and use of variables

As examples of the syntax that is used to define options, here are the definitions for the alternative values of 'font' and 'breakable?' shown in figure 7.3, and the interface controls that InteracTEX generates as a result:

```
\documentstyle[ |< %choose "font"
                10pt 11pt 12pt
                >| ] {article}

\begin |< %choose "breakable?"
           |< no |>|
           |< yes |[allowbreaks]>|
       >| {code}
```



The nested |< ... >| bracketed sections in the second example are for associating the mnemonic names 'no' and 'yes' with the alternative text substitution values being defined: in this case the 'no' alternative is the empty string, which results in the default unbreakable format, while the 'yes' alternative adds an optional parameter to allow breaks.

Limitations

While it is true that taking control of the content of the `.tex` file gives access to all the customisation normally available in building a document file, this level of customisation is itself rather patchy. For example, it is straightforward to adjust the amount of white space to be inserted before each paragraph in a document, with a single-line macro invocation such as:

```
\setlength{parskip}{4pt}
```

By contrast, the specification of the amount of white space to be inserted before the first member of a list of items—intuitively a similar kind of customisation—is not available for independent adjustment, since it is embedded within the definition of the `list` environment. That definition appears in one of the style files, and although it is possible to gain control of a factor such as the spacing by overriding the entire definition, that presupposes that the author of the document knows exactly which style is being used, and hence the full form of the definition that must be replaced. Then if a user decides to try an alternative document style, the previous way of adjusting the list definition might no longer work.

Of course, the standard LaTeX styles are written from the viewpoint that authors should not be seeking arbitrary control over such related properties. But it is not an inherent limitation in TeX, so it would be perfectly possible for a style designer to develop a suitably anarchic style that provided a large number of parameters for control by the author of a document.

7.5.2 Specifying reconnaissance content

Principal requirements

The variety of measurements that my test subjects suggested the system could take was at least as rich as the range of options they wanted to specify—but a lot more challenging to implement, since LaTeX was designed to provide an algorithmically perfect output and to leave it to the user to make any subjective judgements about whether the rules had failed to suit the document. It became clear, however, that providing the following simple kinds of measurement would at least constitute a powerful starter set:

1. reporting the page number, and the position on the page, occupied by a given piece of the document. There are many kinds of information that can be derived from these measurements, including:

- the overall size of the document, and whether it finishes on a mostly-empty page;
- the positions of salient elements such as the start and end of major sections, and hence warnings about a section having its heading near the bottom of a page, or ending near the top of a page;
- the separation between related points in the document, such as a figure and the place(s) where it is referenced, or a set of items that should all appear on the same page, or an item that the user wants to appear on a single line.

2. measuring the length of each typeset page, to reveal whether the formatting has left large white-space gaps;
3. analysing any warning messages that are generated, so the user can judge whether the degrees of 'badness' reported warrant concern.

Facilities provided

I decided that all the measurement facilities, like the processing options, would be specified in textual format. There are two main categories of measurement: 'global' measures that apply to the document as a whole (e.g., the length of the document, and the number of warning messages), and site-specific measures that present information about individual features within the document. The latter require the insertion of 'anchor points' within the source markup to identify the features to be measured.

```
|< %measure "Global" "Last Page %" >|
\begin{verse}
|< "limerick 3" |An old man who came from...
...
As the spelling makes poets so waoghaire.>|
\end{verse}
|< %measure "limerick 3" P= >|
```

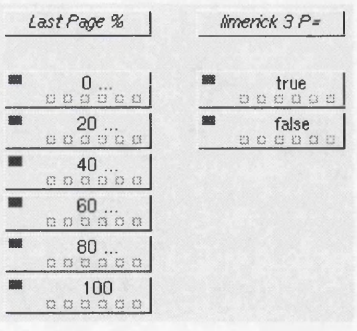


Figure 7.5 Requesting a global and a local measurement

Figure 7.5 shows, firstly, a request for the global measurement called 'Last Page %' which measures how full the last page is (as a percentage of the available text area). The figure also shows the insertion of the anchor points identifying a section of the text as 'limerick 3', and the corresponding request for the measurement 'P=' concerning this section, which generates a boolean value that represents whether or not the two ends of the section appear

on the same page of output. On the right of the figure are the two scales generated by InteracTEX to show these measurements. For simplicity of implementation all InteracTEX scales are made up of discrete elements, so the nodes on the percentage scale represent ranges of possible values; there are facilities for the user to request a customised set of gradations, if wanted.

Limitations

Like the limitations in freedom to adjust formatting parameters, there turned out to be some unexpected constraints on the information that can be made available by way of the log file. In particular the log file cannot report reliably the eventual physical position, on the typeset output, of any part of the document; this is a corollary of the method by which TEX determines the best position for each page break.

It was clear that this kind of measurement is the most valuable of all, so an alternative means of finding the information—making use of the DVI output file rather than the log—was developed after much experimentation (and much help from Alastair Reid, then working in the department of Computing Science, Glasgow).

As for the means of requesting the measurements, the need to insert and edit complex expressions within the document was regarded as laborious. Subject 2 suggested alleviating the problem by providing macros in a tailorable editor such as `emacs`. Subject 6 suggested that the commands should be generatable using commands in the graphical reconnaissance interface itself. I believe there is a need to support both, to allow for different user preferences; the challenge is to make the alternative facilities compatible with each other.

7.5.3 A suitable reconnaissance display format

The sections above have included some examples of the graphical representation used for option and measurement scales. In this section I show how the scales are assembled in a form of parallel-coordinate display, and how the results of processing are presented on that display. The overall presentation is derived from a mix of features found in the interactive parallel-coordinates systems reviewed in chapter 6, combined with some unique additional features designed to suit the reconnaissance task.

All features proposed in section 6.3 are supported. The following sections highlight some of the more important ones.

Collating multiple results

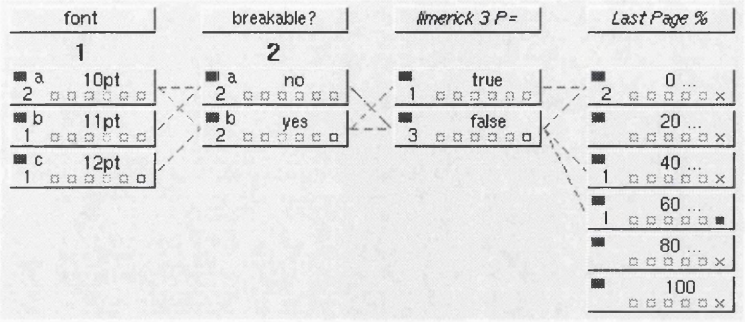


Figure 7.6 A display of four results

Figure 7.6 shows the display after the system has processed the document in four alternative forms. The combinations of input options that have been processed can be seen from the lines joining the first two scales; the availability of particular values for the two measurements of interest can be seen from the lines joining their scales.

Rather than following Tweedie *et al.*'s (1994) approach of using a histogram to show the distribution of results among the nodes on a scale, the number of results passing through each node is displayed on its left. The advantages of this is that it requires less space and is better for showing distributions that have wide variation (e.g., 3 results having one value, 100 results having the next). The main disadvantage is the comparatively low visual impact.

Viewing options and measurements in the same format

The option and measurement scales are represented in the same way, to allow the results to be evaluated according to trade-offs that can involve a free mix of both kinds of result property. As is seen in figure 7.6, the option scales are numbered to help the user remember which ones they are. This seemed to work adequately well.

Highlighting subsets of results

A mechanism for marking interesting sets of results is provided by the six marker buttons visible as small squares on each node, which allow independent control of six different colours of marker. The user has switched on the right-most (green, seen here as black) marker on the `60...` node, so all results passing through this node—in this case just one—pick up

the green mark, and all the other nodes that they pass through show a green outline on the right-most marker button. In this case the marking lets the user see that the result with the fullest last page was the one generated by the combination of '12pt' and 'yes' options.

Switching on the same marker on more than one node is the way to mark results that satisfy combinations of criteria, and by using the different marker colours in concert the user can explore many forms of trade-off. Additional display features that assist in such analysis include the ability to re-order the scales, and to 'switch off' individual nodes to filter their results out of the view (as suggested in sections 6.3.1 and 6.3.2) using the buttons that are seen in figure 7.6 as the large dark blobs at the left of each node.

Exploring alternative correlations

All scales can be 'dragged' sideways using the mouse, and 'dropped' between two other scales to reorder the view.

Viewing results in detail

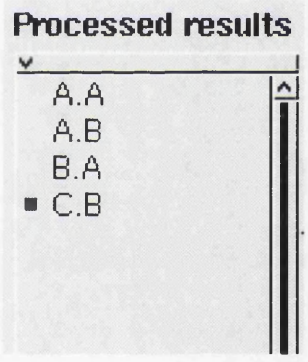


Figure 7.7 A simple result list, supporting result selection for viewing or reprocessing

To request to view a result in full detail—using a high-resolution graphical display tool such as `xdvi` or `ghostview`—the user first needs a way to select the result. In InteracTEX each result is given a name based on the option values used to generate it; the one highlighted in figure 7.6 would be named `C.B` since it passes through the 'c' and 'b' nodes of input options 1 and 2 respectively. Figure 7.7 shows the format of the list of presented results, which includes a display of the markers that currently apply to each one; in this case `C.B` stands out because of the green blob against its entry.

InteracTEX allows the user to create high-resolution views of any number of the results, and—subject to the limitations of the user's workstation screen—to compare them side-by-side if wanted. This is an advantage over the default setup of LATEX; unless the user makes a special effort to copy the DVI file at appropriate moments, only the latest version of the formatting of a document can be viewed at a time.

7.5.4 Exploration progress

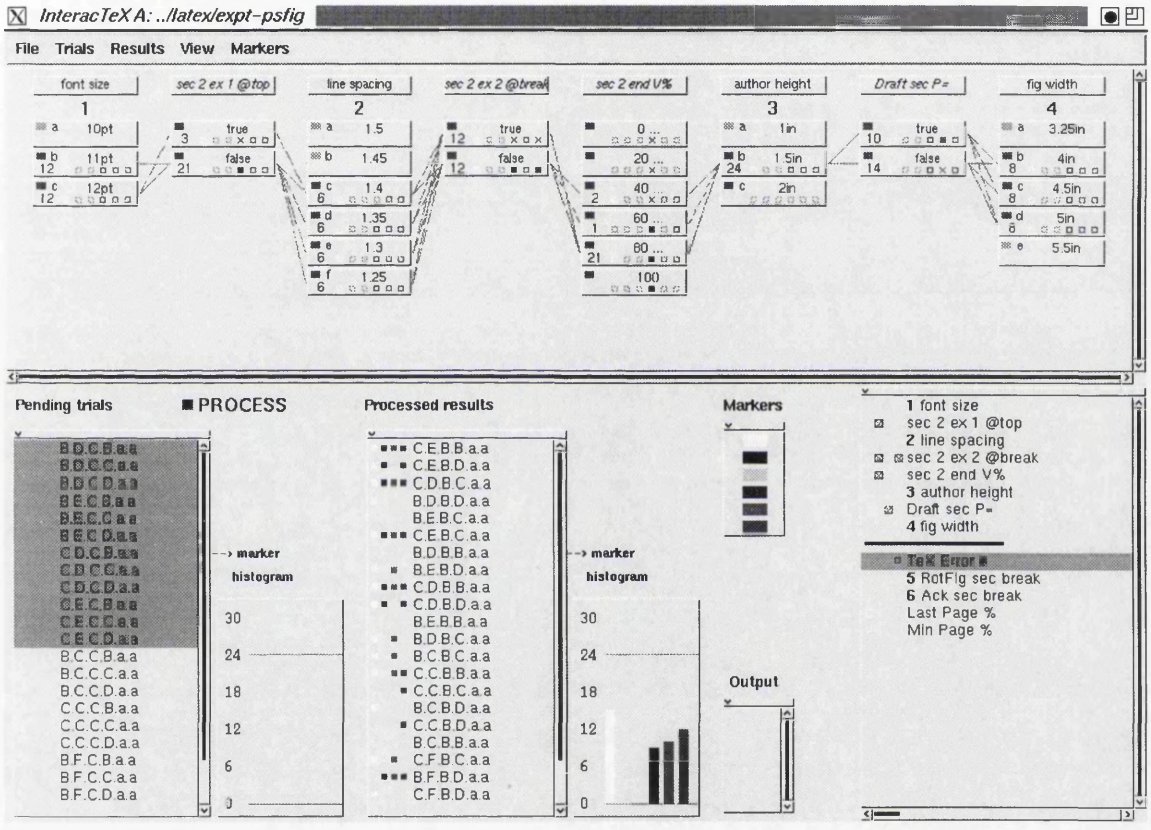


Figure 7.8 A complete InteracTEXdisplay, while working on the tutorial example

Specifying a result range

Each node on any scale can be enabled or disabled by the user. When an *option* node is enabled, InteracTEX automatically sets up a search range that includes the enabled node in combination with all combinations of enabled options on other scales. All these combinations are added to the `pending trials` list seen at bottom-left in figure 7.8.

Launching a reconnaissance foray

The pending list contains all option combinations that the user has enabled but not yet processed. Processing begins when the user selects some (or all) entries in the list and clicks the PROCESS button. Processing stops when there are no more selected entries in the list, or can be halted by the user pressing the PROCESS button again.

Interactive viewing and control of progress

All formatting is carried out in the background, so the interface remains active at all times. One benefit of this is that each result appears on the display, for examination by the user, as soon as its processing is completed. Another is that the course of the reconnaissance can be steered dynamically: InteracTEX picks the next trial to process by taking the top-most *selected* entry in the pending list, so the user can choose not to process some of the pending combinations either by disabling an option node (which removes the affected list entries) or, more selectively, by un-selecting specific entries in the list.

Display 'housekeeping'

A facility that was found to be important for helping the user to manage the display is the ability to hide scales from the display if they were temporarily not needed on view. The untitled list at the bottom right of the InteracTEX window contains *all* the scales; in that list, the entries above the thick black line are the ones currently on view in the main display. The user can drag-and-drop entries within this list as an alternative way of changing the order of scales, including changing the visibility of a scale by moving it from one side of the black line to the other—or even dragging the black line to a different position in the ordering.

7.5.5 Characterisation using the Illumination Zone Model

Figure 7.9 shows an adaptation of figure 4.5 (p.94) to show the components of the InteracTEX setup. Note the portrayal, at far right, of user interfaces that are independent of the reconnaissance facilities: one handling the editing of the two input files (as described on p.176—here shown as `xyz.text` and `xyz.expr`) and the other handling the 'previewing' of a document in WYSIWYG detail. As expected, on the left is the embedded connection from the reconnaissance engine to the document formatting services.

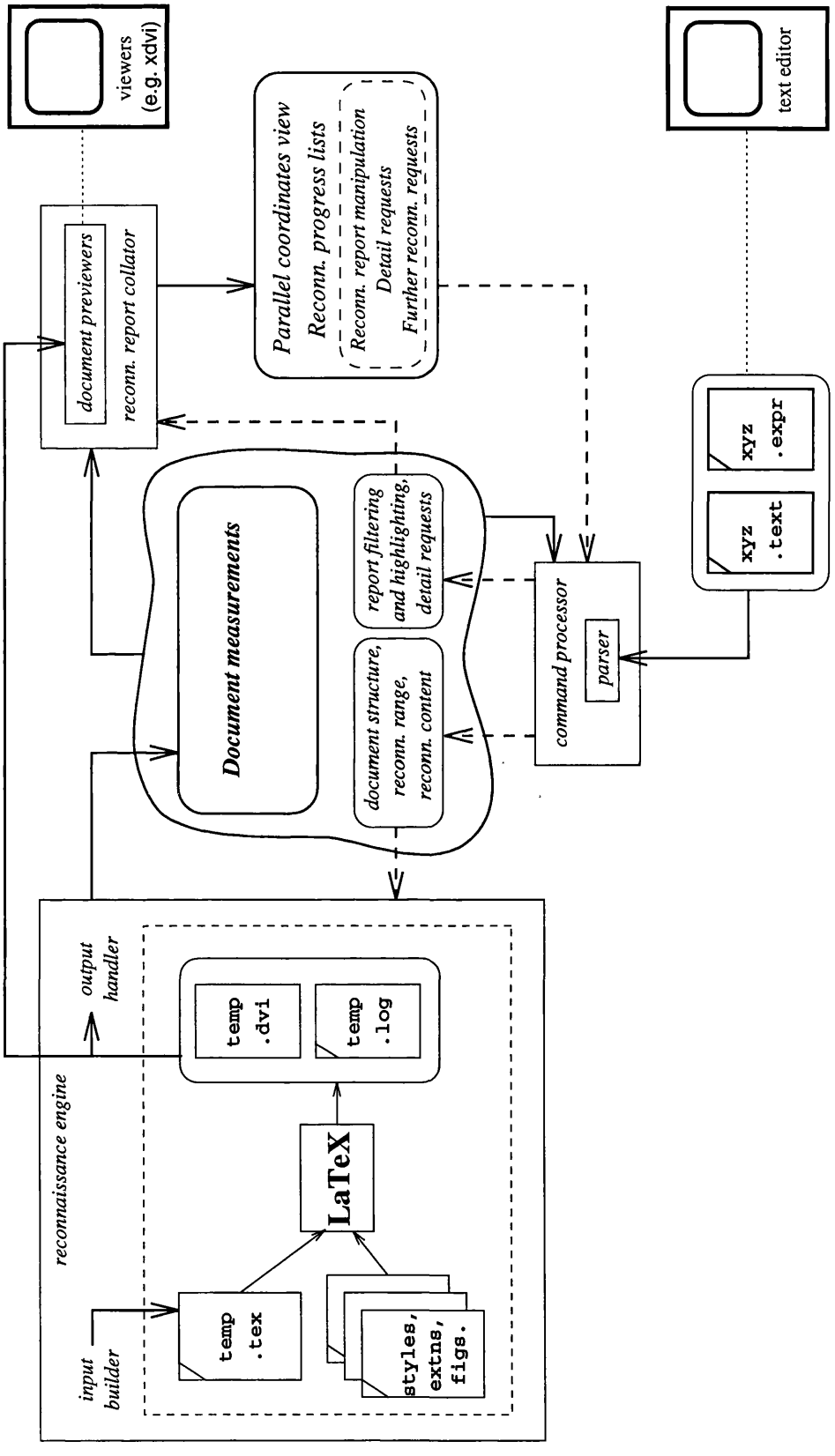


Figure 7.9 Major components of an InteracTEX setup

7.6 Challenge 3: Does reconnaissance encourage accuracy?

7.6.1 Impact on typical use of LaTeX

The need for exploration

All of the subjects acknowledged that LaTeX formatting has a frustratingly inconsistent nature. When a document can be arranged to fit pages in a convenient way the outcome is excellent; when the system cannot arrive at a way to do this the outcome is disappointing. Furthermore, whereas some problems turn out to be simple to resolve, others turn out to be much more difficult.

I observed that this unpredictability has led many of the subjects to develop somewhat defensive strategies for resolving formatting problems. For example, they tend to employ cautious modifications—changes whose impact is relatively predictable—and to work through the document carefully in section order so as not to disturb the pieces that are already 'fixed'. They are also wary of making changes that would be difficult to undo, in case they reach a dead end in their search and want to revert to the best compromise encountered along the way.

It was not surprising to see these defensive strategies, but I was impressed at the degree of pragmatic effectiveness some subjects appear to have achieved. This suggests that although reconnaissance makes it feasible to embark on explorations of arbitrary boldness, it is still likely that a strategy of educated, document-specific suggestions will be more productive than one of shaking everything up in the hope of landing on a fortuitous combination. Indeed, my own experience with InteracTeX included a few occasions of frustration on running an extensive range of experimental options and finding that, for the document in question, these options offered no improvements at all.

That said, all the subjects noted the support for rapid execution and evaluation of alternative ways of processing, and expected that working with reconnaissance would encourage them to explore more options than without. Subject 1, in particular, noted how effectively the collated result presentation allows one to narrow in on the results of interest.

Coping with externally imposed changes

The tutorial, like the original study reported in section 7.4, drew the subjects' attention to the scenario of externally imposed changes (such as having to use a different page size, or remove a large section) for a document that has already been formatted once. All subjects

agreed that such situations were common and normally presented a heavy burden, and that reconnaissance could certainly be of assistance.

In particular, subjects 5 and 6 noted that having already been through the document once, defining an exploration range in terms of alternative acceptable processing options for various points, there was a chance that simply revisiting those same alternatives would be enough to find an outcome that suits the modified constraints. Going further, subject 5 suggested that in some cases the user may know in advance that a variety of formats will be needed, and could prepare for this need by building into a document some alternatives for aspects that were perceived to be good ways of making the document adaptable.

Subject 4 saw the main benefits of reconnaissance as applying to very large or complex documents—otherwise there was little to gain, she felt, because in a small document an experienced user will be able to solve most kinds of problem by inspection, seeing where adding or removing a few words of text will make the necessary difference.

7.6.2 Effort and accuracy

Although all the subjects felt that pursuing reconnaissance should lead to better results, most expressed some form of concern about the effort involved in doing so.

Subject 2, for example, expected that documents he would produce using InteracTEX would be better than normal because alternatives can be tried faster; he would try more alternatives, and he believed this would often bring to light a better outcome than might have been found if only a few alternatives were tried. However, he saw a danger that people would play too much—more than would be justifiable by the improvement in results. Although acknowledging that this was not entirely the 'fault' of the interface, he suggested the risk could be reduced by attention to the level of facilities in the option and measurement toolkit—e.g., providing 'canned' sets of typical changes that are known to work well together.

Many subjects noted that the addition of reconnaissance brought the need for a new kind of effort: deciding what aspects of the output to measure and what aspects of the specification to try changing. Subject 3 regarded this as a worthwhile development of a new skill. Subjects 2 and 4 noted the effort as being in addition to, rather than instead of, the typical challenge of figuring out how to provide hints that will coerce LaTeX into producing a better result. But subject 5 suggested that over the development and refinement of a whole document the additional effort might amount to no more than would have been expended in a traditional cycle of problem detection and fixing—and would bring the compensatory

benefit that, once defined, the adjustments and measurements are an investment that can decrease dramatically the effort needed to make a whole range of changes in the future.

Subject 4 was especially concerned at the cognitive effort of working with the highly abstract multivariate result presentation—having to keep in mind the correspondence between each scale on the interface and the variation in formatting that it represents. She made some suggestions towards a more integrated form of reconnaissance support, specifically for document formatting, that would involve active correlation between the manipulation of options and a WYSIWYG display of the affected region(s) of the output.

7.6.3 New opportunities and challenges

The facilities that were added to enable reconnaissance—being able to measure output formally and to run batches of trials—also enable various ways of working other than the originally intended form of reconnaissance.

Most subjects expected that they would add options and measurements along the way, if it were cheap (for example, implemented as a specialised LaTeX-like tag set). The fact that with reconnaissance these additions do not commit you to anything, but just allow you to leave open an option or to indicate a potentially valuable measurement, makes this practice more likely to be worthwhile than the actively discouraged habit of creating large numbers of hints to LaTeX.

Subject 6 expected that each user would develop a favourite set of options and measurements to be added automatically, for example to investigate the possible benefit of introducing explicit page breaks before any major sections. Many of these would take on the nature of features such as 'conditional page breaks', offered by some text formatting tools—but with the advantage that the tool would not seek to resolve each option individually but explore a wide range of combinations. These could then provide a safety net of available alterations, to assist in resolving further stages of options and measurements that the user would introduce after examining the particular features of a given document.

Subject 4 noted that if large numbers of reconnaissance options are generated as a matter of course, it becomes especially important for the interface to help the user understand what each choice or measurement relates to—for example, with facilities for managing just the critical working set of options and measurements.

7.7 Conclusions

The implementation and evaluation reported in this chapter provided various levels of feedback on the value and applicability of reconnaissance.

7.7.1 Adding reconnaissance to a trial-and-error activity

- **Discovering a useful set of components**

  With the help of existing LaTeX users, it turned out to be feasible to create a set of option and measurement facilities that would act as a toolkit covering a broad range of formatting issues.

  But because some facilities turned out to be particularly challenging to graft onto the existing LaTeX tools, there was only time to implement a relatively small proportion of the requirements that were identified. A designer with more opportunity to change the existing tools (for LaTeX this would simply require greater proficiency with the TeX language) should be able to produce a much more comprehensive set of reconnaissance facilities.

- **Applicability of the option space/measurement space characterisation**

  Because I wanted to explore the feasibility of presenting reconnaissance using the form of interactive parallel coordinates display suggested in chapter 6, I was only interested in supporting reconnaissance forays of a convenient form for such a display—i.e., having a range defined as an option space, and content defined as a measurement space.

  Using an option space imposes the restriction that every trial submitted for processing be described in terms of a value for each option scale. Although I found plenty of useful document-formatting scenarios that can be expressed naturally in this form, there are others for which it is inconvenient. For example, it is hard to express contingent options such as 'either include this single paragraph, or a list of the following items—in which case the list should take one of these three orderings'. InteracTeX's facilities do at least allow a user to work around this restriction, in that using option marking can be used to highlight meaningless trials at the 'pending' stage and then to select them as unprocessable.

  In short, future implementations of integrated reconnaissance facilities for domains such as document formatting will need a better way of overcoming this inconvenience, perhaps based on an entirely different way of requesting reconnaissance.

- **Providing a mechanism for expressing reconnaissance requests**

  In the case of LaTeX it was reasonable for reconnaissance requests to be expressed in textual form, alongside the users' document markup; existing users of LaTeX are accustomed to such abstractions, and to learning complex syntax rules and mechanisms. But although this simplified the implementation, it was not well integrated with the graphical interface aspects. Again, a developer of an existing graphically based text formatter would be in a better position to integrate reconnaissance controls with the existing interface.

- **Integrating reconnaissance with an ongoing development activity**

  InteracTeX has proved to be good for the task of revising the format of an existing document. As well as the testing, I have used it to refine some of my own work (and expect to use it for producing the fair copy of this thesis). This is the principal kind of task for which I designed the system, and is similar in nature to the isolated kinds of exploration task that are usually tackled with retrieval systems.

  However, one of the elements of feedback from the test subjects was a strong expectation that options and measurements could profitably be added throughout the development of a document. InteracTeX does already include many features that would facilitate ongoing development of a repertoire of options and measurements for a given document—for example, being able to introduce new options and measurements, and to hide those that represent issues that are (at least temporarily) resolved—but its current level of manual enabling and disabling of options would make it quite hard work for the user to revisit a large number of options. This is one of the motivators for introducing a further level of reconnaissance-control programmability.

7.7.2 How reconnaissance support was received

- **Recognition of its usefulness**

  The findings suggest that reconnaissance could indeed be a useful additional level of support in this opportunistic domain. The subjects were able to give examples of formatting variations they would like to explore, and measurements that would allow the effects to be summarised. Although I did not have the opportunity to implement all their suggestions in time for the tests, the subjects were able to identify various problems in their existing tasks that they felt would be alleviated by use of the reconnaissance technique.

- **Willingness to accept additional effort for improved quality**

  Although the subjects felt that reconnaissance could bring quality improvements that would be worth the additional effort, the tests that were carried out could not provide evidence showing a clear net gain. Part of the problem is the difficulty in predicting the cost of pursuing reconnaissance; clearly it saves repetitive effort in evaluating a large number of alternatives, but some subjects were concerned that the cognitive effort involved in deciding what alternatives to explore would weigh heavily against the potential advantages.

- **Other ideas sparked by the reconnaissance facilities**

  A finding that I regarded initially as merely obscuring the feedback on the reconnaissance implementation might, instead, be seen as the most significant result obtained from this work. That is the recurring suggestion that the facilities I had provided to support reconnaissance would have a more general role in the ongoing pursuit of a range of possibilities, and not just in exploring ahead at moments of indecision.

  Having analysed the burden imposed by many exploration techniques I developed the reconnaissance approach to assist users in finding which options, out of a wide choice, offer the more promising routes *forward* from the current point in an exploration. The generalisation of this approach takes into account the fact that what is often needed is a look *back*, to revisit those options that have not yet been positively discounted and that might have become more valuable in the light of progress since they were first examined.

  The principal enabling techniques of reconnaissance are: 1) the use of enumerated options to express a range of results worthy of consideration, 2) the use of measurements to summarise results that are generated, and 3) the collation of result reports to allow tradeoff analysis. Taken with the generalisation of continuously revisitable options, these are simply what you need to show a form of opportunistically generated 'map' of a complex result space. This concept is potentially as widely applicable as reconnaissance itself, and certainly represents an intriguing area for future work.

7.7.3 Further experiments

The difficulties encountered in grafting onto LaTeX the various controls and measurements that were wanted left relatively little research time available for evaluations and further iterative development. It is likely that the choice of a less complex domain would have allowed a larger number of tests and iterations, and perhaps even a longitudinal study of working practices being altered by the provision of reconnaissance.

However, the richness of the LaTeX environment, including the complexity of users' existing working practices and their perceptions of the exploration challenges, has allowed even this initial investigation to bring to light many issues that are likely to be useful in future efforts to implement reconnaissance in the same or other domains.

Directions for further development and investigation based on the InteracTeX example, which I feel will continue to be a valuable exemplar for the more general applicability of reconnaissance, include the following:

- obtaining quantitative measures of the relative levels of effort exerted, time spent, quality of document eventually accepted, and confidence about quality, with and without the use of reconnaissance;
- investigating whether users wait to find problems with formatting a particular document, or—as was suggested in the feedback so far—develop standard sets of options and measurements;
- investigating at what stage users start to have difficulty keeping a mental picture of what the reconnaissance display represents, and seeing whether they develop strategies for handling the problem as manageable sub-parts;
- seeing how users react if a sizeable reconnaissance foray reveals no especially suitable results—do they: put in checks that are more lenient? try other degrees of variation? just give up and accept a poor result?

Chapter 8

Suggestions for follow-on research

The InteracTEX implementation and evaluation reported in the previous chapter has served as a 'proof of concept' for the use of reconnaissance in one form of opportunistic exploration. In addition to further development and testing of InteracTEX, this thesis sets the scene for various directions of research relating to the more general exploitation of reconnaissance. This chapter describes some of those research directions.

Section 8.1 addresses the investigation of further domains in which reconnaissance may be useful. I outline four potential applications of the technique that are different from those considered so far, and that bring to light various potential roles for reconnaissance that may be found worthy of further investigation. The findings from such implementations will be important to the research proposed in section 8.2, which is to address the architecture and construction of generic facilities to assist designers in enabling reconnaissance for diverse application areas. Finally, section 8.3 outlines some of the challenges involved in quantifying the impact of reconnaissance, by observing users' reactions and evaluating its influence on the *effort–accuracy tradeoff* in their work.

8.1 Further implementations in diverse domains

As outlined in section 4.5, result-space reconnaissance can be expected to serve a useful role in a wide range of opportunistic explorations. A primary goal in further research is to ascertain which computer-based tasks can be characterised as involving explorations of a suitable kind. Within this thesis, chapter 4 illustrates how reconnaissance could be applied to a straightforward search within a flight database, while the InteracTEX implementation

demonstrates an alternative form of exploration that arises implicitly in choosing the formatting for a document. The following examples describe other potential ways of applying reconnaissance.

1. **Exploring the implications of weighting-based preferences**

   There are many computer-based tools that perform calculations based on 'weightings' specified directly or indirectly by the user. But these weightings, typically expressed as numerical values between zero and one, can be non-intuitive to a decision-maker who needs to make trade-offs between factors of fundamentally different type—such as price, engine power and luggage capacity of a new car.

   As one attempt to alleviate this problem, Vetschera (1994) describes MCView, a tool for exploring alternative rankings of multi-attribute results according to variation in the weights and aspiration levels assigned by the user to each attribute. A novel feature of MCView is that it allows the user to indicate specific 'holistic' preferences between results—assertions of the form that some result P is definitely preferred to some other result Q—and then to request that the system execute an 'estimation function' to determine a preference representation (an assignment of weights and aspiration levels to the attributes) that is compatible with as many as possible of the expressed individual preferences. Since there may be many representations that are equally compatible—e.g., in terms of how many of the holistic preferences they enable to be satisfied—the system is designed to take into account the existing representation and to select the closest of the available solutions. The approach therefore works best when the preferences are changed in a steady refinement from the user's initial estimates.

   A consequence of this is that MCView does not provide good support for a decision maker who wishes to explore several widely disparate preference representations—for example, being undecided as to whether to look for a car that is spacious even if a bit sluggish, or one that is powerful for its weight but is still cheap to run. It would be arduous for a user to make separate requests in order to force the estimation function to work from each of these different starting points.

   A way of using reconnaissance in such a task would therefore be to show, for each proposed addition to the holistic preference specifications, the alternative estimation-function results that would arise for each of the user's specified base-level preference representations. The kind of summary information that the user may wish to see for each alternative outcome might be a list of the holistic preferences that failed to be reconciled in each case.

2. **Progressive development of alternative artifact designs**

   A similar approach to the above could be used within the more concrete task domain of constructing interactive presentations. For example, a presentation designer may wish to let the viewer choose between various levels of presentation (overview or detail, each with or without spoken commentary), and may be undecided on how to organise the information in a way that will allow the viewer to find convenient paths through related topics. An example of alternative organisations would be a hierarchy that could be based on different primary and secondary information characteristics—e.g., course information arranged by faculty and then by degree level, or the other way round. For each organisation it is clear where the information for a particular faculty and degree will fit once it has been prepared, so the authoring system can allow a designer to provide the details just once and will then automatically slot them into their appropriate places in the candidate organisations. Summaries of the designs, as they are assembled, should be based on measurements that will allow the designer to notice undesirable features of the presentation as a whole, such as excessive delays in pre-fetching (e.g., from CD-ROM) all the materials a viewer might decide to select from a particular screen. Being alerted of designs that are failing on such a criterion may help the designer to choose between the alternative organisations, or may prompt a change in presentation content or format to relieve the detected problem.

3. **Measuring divergence of query refinements in relevance-feedback IR**

   When a user of the News Retrieval Tool (NRT, described in section 3.2.4) specifies just one of the retrieved articles as being particularly relevant to the search, the next retrieval will be biassed towards other documents that contain the same key terms. A potential hazard is that if the document uses a rather restricted terminology for the user's intended subject area, the bias will bring up only other documents that use the same terminology. But if the user selects a number of documents as being relevant, the system produces a single retrieval specification that takes into account the key terms of all the selections. If the documents all use a consistent terminology there is the same danger as before, but if they use radically different terminology their combined effect may be weakened. In the worst case the reformulation would fail to constitute a meaningful refinement of the query.

   Reconnaissance support could be applied to alert the user to the need to pursue the retrieval in independent directions rather than attempt to merge the relevance feedback into a single path. For this it may be sufficient to show the user whether, for a given set of documents that are relevant, the various retrievals that would result from feeding-back each document individually would contain a significantly different

set of highly-ranked documents from the retrieval that would result from the combined feedback. Any document that clearly stands out from the crowd may deserve to be handled as a separate query. (Whether the system makes it easy to pursue parallel queries, for example by supporting disjunction, is a separate issue.)

4. **Behaviour analysis for modelled systems**

   When working with a mathematical model, for example as embodied in a sales-strategy spreadsheet or a weather simulation, it may be important to know whether the model's outcomes are excessively sensitive to the starting values of some parameter(s), or of some coefficients within the model. Similarly, a rule-based model in an expert system might be excessively sensitive to the initial axiom assertions or the use of alternative reasoning strategies. If the user can specify those parameters or axioms whose values are open to some doubt, a reconnaissance foray that re-evaluates the model under all significant combinations of value perturbations can be summarised to provide information on the stability of the model. If some parameter turns out to be critically important the user may wish to pay close attention to its value.

The first two examples illustrate a form of reconnaissance that is subtly different from the domains considered in the earlier chapters. Previous examples, such as the flight search, emphasised the pursuit of progress by extension of the range of cases illuminated—e.g., trying alternative departure dates or cities. The examples above, by contrast, show that it can be useful to have a fixed range of illumination—namely, the various alternative designs being explored—but to keep re-evaluating the results within that range in the face of ongoing development of the information base from which the alternatives are derived. This is not to say that the different emphasis constitutes a clear-cut categorisation of reconnaissance types; in any given embodiment of reconnaissance a user may wish to pursue an exploration that includes elements of both. Indeed, the use of such a combination is implied by the InteracTEX test subjects' suggestion of maintaining a portfolio of alternative settings for re-evaluation, in case later updates alter the balance in previously explored trade-offs.

The latter examples also illustrate a potentially important distinction in the nature of reconnaissance: rather than reporting directly on the attributes of results produced by the underlying process (i.e., some matching documents or some predicted values), reconnaissance can be used to provide information about the state of the user's exploration as a whole.

But note that all the proposals so far satisfy the constraint, declared in section 2.1, that this thesis should only address applications in which exploration progress is directed explicitly

by the human user. There are various ways of going beyond this constraint that may turn out to be powerful and useful, including the following:

- **Reconnaissance under programmed control**

  Programming abstractions could be introduced to raise the available level of control of reconnaissance pursuit. Effectively this increases the complexity of the instructions to the 'scouts', so it is typically only relevant once the user has gained some details of the nature and layout of available results. Then the user may request, for example, that as soon as a result satisfying a given condition is found within some region of the option space the search within that locality should be suspended—or that the search should be diverted to alternative regions based on what has been discovered. Providing such additional programmability, yet retaining the user's feeling of control and comprehension of the search progress, will be a substantial challenge for designers of reconnaissance-based interfaces.

- **Reconnaissance under heuristic control**

  In some task domains users are prepared to accept computer assistance driven by heuristics, because the complexity of the domain makes an analytical approach impractical. Such complexity could arise in attempting to use reconnaissance in an especially large result space, so there may be cases in which the generation of the reconnaissance missions themselves can usefully be entrusted to a mechanism that the user does not fully understand. A 'genetic algorithm' approach, for example, would be able to work through a large number of reconnaissance missions, pruning the exploration by cutting short missions into regions of the option space for which the results are seen to have converged[1]. Again, the main challenge in designing such support will be in providing appropriate interaction between the heuristic and user-controlled aspects of reconnaissance pursuit.

Supporting a useful range of approaches to reconnaissance will be a crucial factor in designing generic components that can be used in many different domains. In addition there are issues, essentially orthogonal to the above, relating to the nature of the underlying exploration itself—for example, whether the exploration is monotonic or may require backtracking, is bounded or unbounded, has results that vary smoothly or stochastically over the input ranges of interest, and so on. But all these factors relate principally to the question posed by section 4.5: Can reconnaissance serve a useful role? Having established a role in a given domain there is still the challenge, as raised in section 4.6, of implementing software

[1]Recent work on genetic algorithms is reviewed by Srinivas and Patnaik (1994).

components to provide the necessary facilities. The following section explores some of the generalisations that will help in addressing this challenge.

8.2 Developing generic reconnaissance-support facilities

With the experience gained in developing reconnaissance for various domains, it should be feasible to design generic software components that 'factor out' common facilities required for reconnaissance support. Such components could then be provided within system construction environments, easing the task of implementing reconnaissance where it would be useful.

Section 4.6 introduced a broad distinction between reconnaissance provided by means of a 'reconnaissance shell' and that implemented in an 'integrated' form. These are just the extremes of a complex spectrum of potential integration of reconnaissance facilities with domain-specific software. Although the idea of a shell is attractive in that it would allow reconnaissance to be added to an existing application without further work by its designers, in many cases such facilities would offer severely restricted capabilities. The following subsections therefore explore some of the issues involved in encapsulating reconnaissance facilities within reusable components.

The first subsection addresses the extent to which shell-based facilities could obtain access to data and commands in systems that have no built-in reconnaissance support. Then I describe how an application developer might obtain additional support simply by using an application-construction toolkit whose components include abstractions that are relevant to reconnaissance. The final subsection, by considering a breakdown of reconnaissance into its constituent activities, illustrates some of the variation that can arise in the most general cases.

8.2.1 Supporting applications that were not designed for reconnaissance

As explained within chapter 7, a useful feature of LaTeX as an implementation domain was the ability to create InteracTeX as virtually a pure reconnaissance shell. Apart from some minor macro redefinitions the LaTeX engine required no alteration to play its part. And since LaTeX is directed by purely textual input, requesting alternative results simply corresponds to supplying alternative pieces of text. When a system produces purely textual output, as do typical flight-booking databases, it may be similarly straightforward for a reconnaissance shell to take measurements from the results.

It is not only text-based systems that can offer this level of accessibility. The Triggers programming system produced by Potter (1993a, 1993b) is a demonstration of the potential for providing control and measurement of systems that have an entirely *graphical* interface. Triggers provides capabilities for pattern matching among the pixels displayed on a computer screen, and for simulating use of the mouse and keyboard to invoke operations. Its capabilities appear promising for adaptation to the specification and analysis of reconnaissance forays.

There are inevitable limitations to surface-level access to a program's operation—i.e., access based on nothing more than what the user would normally be able to look at, or to specify. A limitation that was experienced in building InteracTEX is that the range of aspects accessible to the user may have gaps and inconsistencies. Another issue is the potential difficulty of reconstructing formalised information from a format that has been designed for human consumption. Finally, a practical limitation for a front-of-screen approach such as Triggers is the delay and distraction involved in displaying intermediate results on screen for analysis, rather than being able to process them all behind the scenes.

However, with regard to the difficulty of information reconstruction it should be noted that in some cases of reconnaissance it would be acceptable *not* to analyse the output; locating a piece of graphical output and displaying it 'as is' might be the best approach. For example, a user experimenting with values in a spreadsheet might be particularly interested in a derived result that is hard to 'see' in the figures but would show up clearly in a line graph. If the spreadsheet can plot this graph, and the result measurement can be defined in terms of locating and copying the relevant small region of the graph (e.g., enough to show the trend during one particular month), then the results from a reconnaissance foray could be assembled as a palette of these small regions for the user to view and compare.

8.2.2 Providing a reconnaissance-aware interface toolkit

A further degree of reconnaissance integration could be achieved, without significant burden on the designers of a domain-specific tool, if they can build its interface using a construction kit whose components automatically incorporate some reconnaissance-enabling features.

For example, a user presented with a menu listing various alternative parameter values may be able to indicate (e.g., using some form of 'shift-click') each of the entries that are potentially of interest, and have them accumulated in an option scale on a reconnaissance display similar to InteracTEX's parallel coordinates view. Likewise, 'shift-clicking' on a field within an information display could register that parameter as being a measurement of interest as part of the reconnaissance results, and lead to a measurement scale being

created. Similar facilities could be provided for registering interest in positions or regions of direct-manipulation controls such as sliders, data selections, or even the movement/resizing handles on graphical objects.

To support this level of interaction, the interface components need to incorporate application semantics of the level suggested by Johnson (1992) in his proposal for *selectors.* The characteristics that distinguish selectors from typical 'widget' components include the following:

- Selectors encapsulate application semantics, so their underlying values can serve both as application variables and as state variables for the interactive controls.

  This is what would enable a reconnaissance control system to interpret a user's selection as referring to a parameter choice or measurement within the application, rather than just a mouse-click on a particular part of the display.

- They embody independent details determining their presentation, so the same underlying value can be presented in multiple selectors having different roles.

  In particular, independent definition of the roles of the overall selector and its subcomponents (e.g., a one-from-many choice, and the individual options on offer) would enable the selector to be reconfigured to have an appropriate appearance and behaviour when transferred into a reconnaissance-control role.

Figure 8.1 shows the role played by selectors in the Application Construction Environment (ACE) as reported by Johnson, Nardi, Zarmer and Miller (1993). The middle level is the key to the ACE architecture, embodying components that are intended to allow application construction by people who are not professional programmers: *local developers* are domain-specific experts who can be expected to create their own applications if the necessary components can be 'plugged together' or controlled and coordinated using a straightforward 'formula language'. The other elements in this middle layer are the *application data types,* which encode the semantic and display behaviours of the general and domain-specific data types that are to be manipulated, and *visual formalisms.*

Nardi and Zarmer (1993) describe visual formalisms as objects that 'provide a specific orienting framework in which to cast an entire application. For example, a spreadsheet, a specialized table, allows users to develop applications in which the relations between numerical quantities are laid out and organized within the rows and columns of the spreadsheet table. The spreadsheet provides a structure into which a model is cast.' This capability of visual formalisms is due to their embodiment of 'strong representational, editing and browsing capabilities in reusable objects that can be specialized for specific applications.'

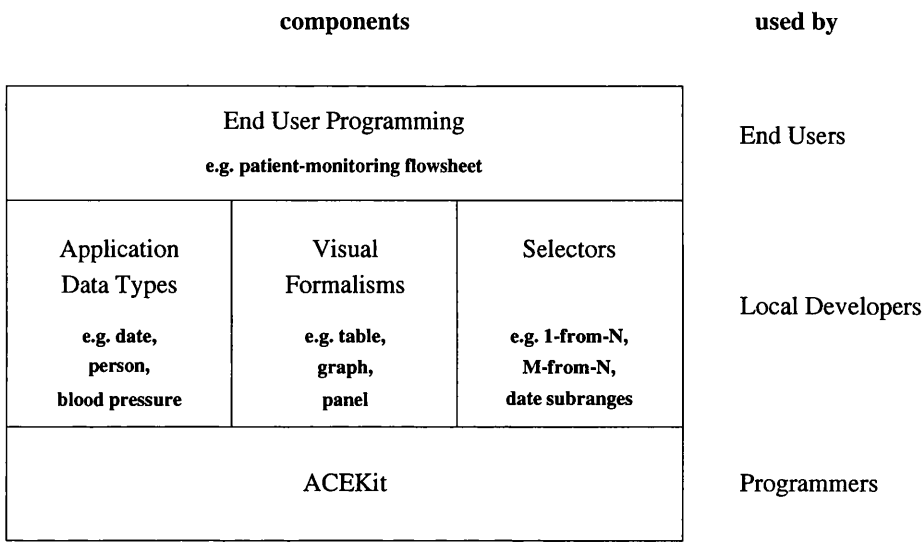


Figure 8.1 Component levels in the ACE architecture, with examples taken from an Intensive Care Unit patient-monitoring system (after Johnson *et al.*, 1993).

The ACE project is targeted at specialist local developers who will create additional systems from time to time as new needs are recognised. But the form of construction afforded by a kit of application data types, visual formalisms and selectors is exactly the level that is needed *each time* a user seeks to perform reconnaissance over some opportunistically chosen collection of selector options and application-variable values. A 'reconnaissance panel' visual formalism could be used as the framework for the overall coordination of a reconnaissance foray, while the selectors and their associated data-specific displays would take their places in some form of interactive display structure (which might be a parallel coordinates formalism, or some alternative) to allow the user to perform the necessary analyses.

Now that an appropriate level for the provision of reconnaissance-enabling components has been identified, the next section addresses in greater detail the roles that these components will have to fulfil.

8.2.3 The constituent activities of reconnaissance

Figure 8.2 shows the principal components required for supporting reconnaissance, using the structure of InteraCTEX as an example. The grey-scale colouring used for each code rectangle indicates whether it represents a facility that is a) fully generic—i.e., contains

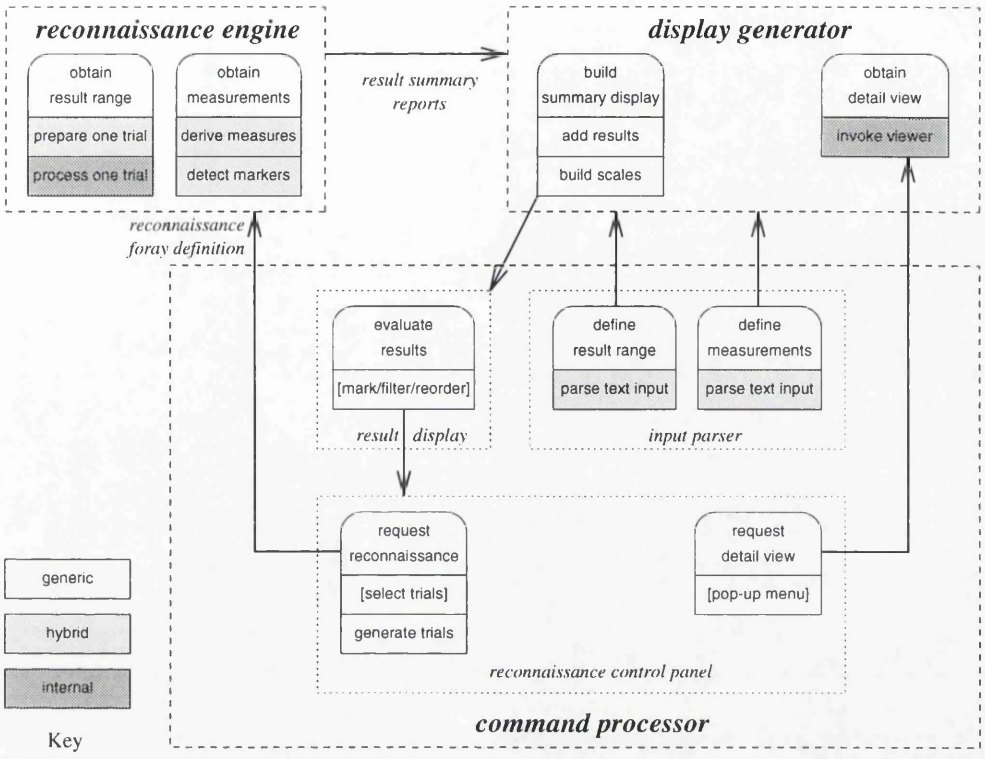


Figure 8.2 A breakdown of reconnaissance support within InteracTEX. Each round-topped box represents a software component that lets the user perform the activity named in its heading. The plain or shaded rectangles represent program components or (in square brackets) supported user actions; multiple rectangles represent distinct pieces of code that are all required.

no behaviour particular to the domain being controlled, or b) a hybrid built especially to combine the needs of reconnaissance and of the domain, or c) a capability that is 'internal' to the existing implementation but fulfils some reconnaissance role. Since InteracTEX is essentially a reconnaissance shell, the only parts that count as internal to the underlying system are those responsible for processing a single formatting 'trial' and for creating a full-detail view of a result. However, several of the parts are—at least in principle—generic reconnaissance-support facilities that could be used in other domains.

By way of contrast, figure 8.3 shows the reconnaissance support as envisaged for an implementation of the information retrieval (IR) divergence-detection feature proposed in section 8.1. In this domain the user is not required to make a special request for reconnaissance, nor to define the measurements that are to be taken. The result summary information is likewise integrated with the existing display. Thus almost all the facilities need to be of the 'internal', application-specific variety.

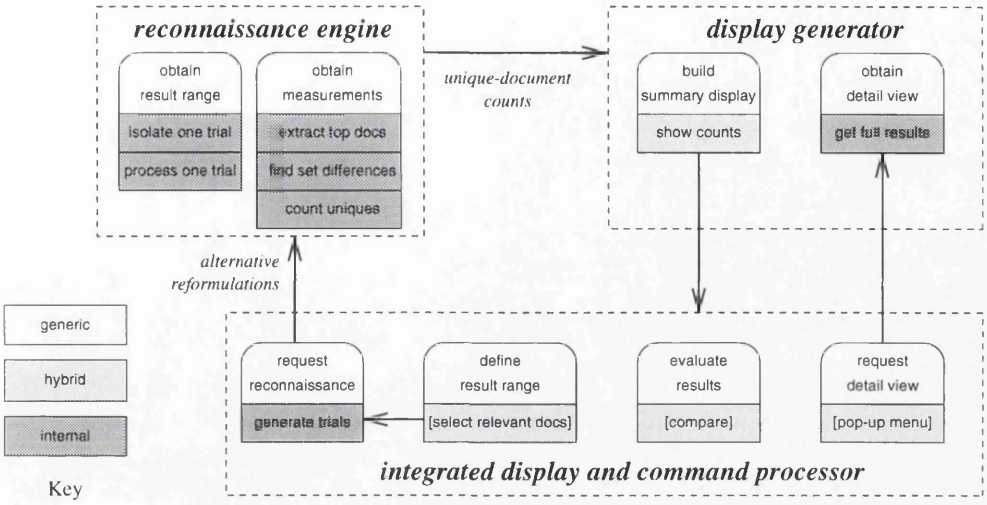


Figure 8.3 Support for relevance-feedback IR with divergence detection.

To provide further explanation of these figures, the following three sections describe the implementation of the major organising units, as defined in section 4.6 and shown here as dashed boxes. A fourth section illustrates some dimensions of potential variation in reconnaissance support, by sketching out a few ideas for alternative implementation approaches that may be useful in particular domains. Finally, there are some comments that indicate the challenge of trying to reuse existing program code at this early stage, before a range of applications has been implemented.

Command processor

The command processor is responsible for coordinating the provision of reconnaissance, by allowing the user to perform the following actions:

- **Define result range**

  This is the task of identifying which parameters the user is interested in varying, and which values to apply to them[2]. In InteracTEX the user embeds this information within the LaTeX source using a textual markup language that is interpreted by a specialised parser. In the proposed IR example the range is defined implicitly by the user's selection of several documents to be used in relevance feedback.

[2]It is referred to as 'result range' just for brevity. Of course the user specifies reconnaissance in terms of a range of input variation, without necessarily knowing what range of results this will generate.

- **Define measurements**

  This is the identification of measurements that are to be taken from each result to allow summarisation. In InteracTEX this, like the result definition, is achieved using the dedicated textual markup language. For the IR example the measurement is (unusually for reconnaissance) entirely pre-defined, so there is no need for explicit support.

- **Request reconnaissance**

  Since the pursuit of reconnaissance can consume a large amount of time and computing power, the user should be given a level of control that encourages an appropriate level of consideration before embarking on a foray. This idea corresponds to the principle of *commensurate effort* as discussed by Thimbleby (1990, p.359), the goal of which is to minimise *regret* by discouraging designs in which small actions by the user can have a large associated cost. In the case of InteracTEX, all the pending trials are accumulated in a list whose length the user can judge, and processing begins when the user selects some or all of the list entries and presses the 'process' button. But the danger of incurring excessive cost—for example, by having selected a thousand trials to be processed—is mitigated by support for halting the processing, adjusting the selections and restarting. All these controls appear on the InteracTEX *reconnaissance control panel,* that also contains and integrates the summary-result display.

  It is a significant feature of InteracTEX that, as can be seen in figure 8.2, the user's reconnaissance range and measurement specifications are not handled directly by the command processor but are passed first to the display generator. The summarised-result display is built before reconnaissance has taken place, which allows the same display elements to be used both to control the requested reconnaissance range and to view the results when they arrive. The rationale behind this 'inter-referential input/output' approach is described below, in section 9.3.2.

- **Evaluate results**

  Once the results have been summarised, the user needs facilities for evaluating and comparing them. For the IR example I would expect the results to be displayed in an integrated form alongside the documents in the normal relevance-feedback display, and compared by eye. InteracTEX's summary display is based on parallel coordinates, and incorporates complex facilities allowing the user to mark, filter and reorder the display scales; the effects of these actions are also reflected in the various widgets of the control panel.

- **Request detail view**

  The summary display allows the user to see which of the results are the most interesting with respect to the requested measurements. But those results will usually need to be viewed in full to discover the attributes that were not (perhaps cannot be) summarised. In InteracTEX the user requests detail views from a list of processed trials in the control panel.

Reconnaissance engine

Driven by requests containing range and measurement specifications for a reconnaissance foray, the reconnaissance engine is responsible for generating and measuring all the results in the requested range. One way of breaking down this process—but not the only one, as pointed out in the section below on specialised implementations—is to perform it in the two obvious steps:

- **Obtain result range**

  This stage creates the results that arise within the specified range. In InteracTEX the reconnaissance controller submits a single trial at a time, by explicitly generating separate versions of the document to be formatted and invoking the LaTeX engine once for each version. For the IR example the naive approach suggested in figure 8.3, of evaluating each potential retrieval as an entirely separate case, may in fact be unrealistic and unnecessary.

- **Obtain measurements**

  Once the results have been generated, the system must analyse them according to the measurements that have been requested. For this purpose InteracTEX runs a special 'spot detector' that analyses the formatted output file to discover the eventual positions of points that were marked in the input file. Calculations are then performed to derive the measurements that will be meaningful to the user.

  Figure 8.3 suggests one plausible combination of operations for analysing the alternative retrievals in the IR case: first isolate an interesting subset of the result ranking (e.g., the top 20), figure out which of these documents are not high in the combined-feedback retrieval, and count them. Every retrieval must be compared with the combined case, so that should be evaluated first.

Display generator

- **Build summary display**

  From a batch of results expressed in terms of their measurements, this component must construct the collated display that will allow the user to evaluate and compare all the results. Although the work on parallel coordinates presentations shows that a wide range of data types can be supported using such linear scales, there will be some forms of data that require more specialised displays—for example, attributes for which each result has not just a single value but a combination, such as the different combinations of people who would be available at various proposed meeting times. Similarly, our IR example might be best served by a display technique similar to TileBars, as demonstrated by Hearst (1995).

- **Obtain detail view**

  When the user requests a detail view, the display generator will typically need to call on the underlying application to furnish it. To provide these views quickly, InteracTEX includes the simple optimisation of retaining all the formatted results even after they have been measured; clearly in some domains this approach might be prohibitively expensive on resources.

Some suggestions for specialised implementations

- Based on the idea of *selectors* as described above, one approach to defining alternative sets of attribute values would be to accept multiple sets of values from a multi-variate dialog box. For example, the MCView decision-analysis tool prompts with a dialog box when the user asks to examine or change the attribute weights; a reconnaissance-aware dialog would have a special form of 'Accept' button that accepts the current settings but remains available for further interaction, thus allowing several independent sets of weightings to be specified. A benefit of integration at the dialog-box level is that the underlying system is determining a useful level of parameter grouping—i.e., the parameters that appear together within a single dialog. In a word processor, for example, this form of interface would allow the user to define different combinations of the parameter settings that together constitute a 'style sheet'.

- One example of an integrated technique for defining reconnaissance range in a direct-manipulation environment (such as a structured-drawing editor) could be derived from the use of *semantic snapping* (Hudson, 1990). An interface that supports semantic snapping aims to help a user specify valid targets during a direct-manipulation

dragging operation: as some graphical object is manœuvred into proximity with other objects the potential for useful connections between them is indicated graphically. If, at a given point in a manipulation, the system has detected several alternative valid connections, the user may be interested in trying each of them in turn rather than having to select just one. Providing some form of 'hotkey' to take a snapshot of the currently identified connections would allow the definition of a range of alternatives for exploration in a reconnaissance foray.

It may turn out that the only suitable form of summary display for such an interface would also have to be closely integrated with the existing system—for example, there might be a way of showing the results as alternative overlays on the working display.

- As was suggested above in the discussion of ACE, a promising approach for straightforward specification of *measurements* would be a facility that lets the user register interest simply by pointing to a displayed instance of the measure—for example, when it is on view within a requested information display such as a dialog box.

  Such connections could be implemented easily for values that are already held as application variables, but an intriguing extension to this approach would be a 'specification by demonstration' facility, so the user can simply perform a sequence of actions that culminates in copying and pasting some derived measurement into its place on a summary display. A system incorporating the technology demonstrated in Eager (Cypher, 1991) could keep a record of such an action sequence, and then execute appropriately modified sequences to obtain other values—e.g., for all the remaining values in some list. This in itself would constitute support for simple reconnaissance based on a single-attribute range; more general reconnaissance ranges would require facilities such as nested loops and non-contiguous iterations, that were not available in the 1991 version of Eager. The reader is referred to (Cypher *et al.*, 1993) for guidance on the facilities available in recent programming-by-demonstration systems.

- In some domains there will be great scope for optimisation of the result generation and measurement phases. A simple example of optimisation was proposed in the note in section 4.2.4, which suggests combining a large number of narrow-range database requests into a single request that will cover the appropriate range of alternatives. The enlarged query will probably provide a lot of additional results that the user does not want, but the system can easily filter them out.

  Another example of tight collaboration between result generation and measurement would be a lazy-evaluation system that can generate measurements without having to produce the entire results. One may find, for example, that a Computer-Aided

Design system can produce measurements relating to the physical properties of an artifact without having to perform any of the computationally intensive rendering of a graphical display.

Such optimisations clearly offer chances of radical improvement in reconnaissance performance, but the opportunistic nature of reconnaissance makes it unrealistic for a designer to hope to develop optimisations that will help on every occasion.

Challenges in reusing code

Within the InteracTEX support components there are notionally two distinct portions of code: 1) the code designed specifically to relate to the language and operations of the LaTeX processing engine, and 2) the code providing generic facilities that could be equally applicable in domains other than LaTeX—such as the parallel-coordinates view. However, although the InteracTEX code was designed to separate generic from clearly application-specific code (using 'abstract superclasses' to define the structure), any first-generation architecture such as this is likely to be insufficiently flexible for further implementations. Bearing in mind the variety suggested by the above discussion, we can see that issues such as the following are likely to arise:

- Facilities that were combined naturally within a single class in InteracTEX may need to be split into a reconnaissance layer and an application-specific layer, with appropriate communication between them. For example, in InteracTEX the code that parses the input markup language is also responsible for creating the objects that represent the sets of options available within the document.

- Some domains may require different divisions of responsibility, and hence different kinds of communication, between the various reconnaissance components. For example, in InteracTEX there is a rigidly defined relationship between the parallel coordinates display and the reconnaissance control panel: when the user enables a previously disabled input option this automatically causes the creation and display of further pending trials, to include that option alongside all available combinations of other options. In other domains the generation of alternative cases may be subject to different rules.

- InteracTEX's use of a monolithic command processor—that has supervisory control over all trial generation, processing, result summary display, tradeoff analysis and detail-view requests—will certainly be inappropriate for many domains.

8.3 Testing the impact and value of reconnaissance

In this section I describe various kinds of testing that are needed for evaluating the impact of the reconnaissance technique.

Result-space reconnaissance is an abstract concept. As is the case for other concepts, such as 'direct manipulation', it is not meaningful to attempt to evaluate the concept itself but only its embodiments in various domains. Therefore any tests suggested here should be considered as suggested directions of investigation for *each* domain for which reconnaissance is developed.

The proposed testing is divided into the following areas:

1. factors that influence a computer user's selection of reconnaissance as a progress strategy, alongside or instead of alternative available approaches;
2. the subjective and observed costs and benefits of pursuing reconnaissance in cases for which we expect it to offer clear advantages;
3. issues arising in explorations that are too large for reconnaissance to be pursued exhaustively, but in which well-informed use of reconnaissance may provide substantial benefits.

8.3.1 Factors affecting selection and pursuit of reconnaissance

Unless reconnaissance is the default (or only) exploration strategy available to a user embarking on some exploration, the first requirement for obtaining benefits from reconnaissance is that the user makes an appropriate decision to use it. Payne, Bettman and Johnson (1993) suggest (as reviewed in section 2.3.3) that a user will only elect to change to a new strategy under the following conditions:

- a belief that the present strategy gives less than the desired accuracy,
- being able to see that there is an alternative better strategy, and
- a belief that one is capable of executing the new strategy.

A set of experiments could be based around these conditions, using a task domain for which reconnaissance is not the only available strategy—for example, retrieval from a flight database. If the database used for the experiments is in fact a mock-up, the conditions

could be controlled by having each subject follow a set of steps (as if responding to requests of a travelling 'customer'), but providing different instruction orders for different subjects in order to influence the attractiveness of the results that are encountered near the start of the search. Subjects who find themselves working through unrelentingly unsatisfactory results may be more motivated to hand over control to a search engine that will not require them to perform so much repetitive query submission and result evaluation. This effect should be strengthened if the experiment is also adjusted to impose on some subjects a substantial result-retrieval delay (e.g., fifteen seconds per query, rather than five seconds or less).

Changing the order in which the subject is informed about additional search criteria could also be used to influence the subject's realisation that reconnaissance might serve as a 'better' strategy: if the early queries bring to light a great mixture of criteria, the subject might not readily come to the conclusion that evaluating a homogeneous region of the search space could be worthwhile.

Along with these tests, but placed either before or after them to balance learning effects, one could ask the user to complete tasks that are more free-style in nature and therefore allow them the option of pursuing the exploration either by reconnaissance or a normal stepwise approach. For example, they could be given one or more criteria and asked to find, as quickly as they can, any result that satisfies those criteria. Alternatively they could be asked to find the best result possible in (say) five minutes of searching. These experiments would provide the opportunity to observe user behaviour under truly 'opportunistic' conditions, rather than the contrived operating sequences proposed above.

8.3.2 Observations and subjective estimates of effort and accuracy

It is hoped that the testing of reconnaissance will lend support to the following claims:

- Relative to the time spent experimenting, reconnaissance will help subjects to find better solutions to the tasks they are set. Where an optimal solution exists, use of reconnaissance will therefore increase the likelihood that the subject will find that result.

  This can be tested simply by concealing an optimal result in the data set, and noting whether subjects a) come across it, and b) correctly assess it (whether immediately or later) as being the best result on offer.

- Relative to the time spent experimenting, subjects using reconnaissance will reach a more realistic assessment of their likely 'accuracy'—i.e., of the likelihood that their eventual solution is one of the best available.

The subjects can be asked to give a subjective evaluation of their level of accuracy, and their (presumably related) impression of the thoroughness of their exploration relative to the overall size of the result space. These estimates can be compared against the actual values available to the test organiser.

Subjects should be asked to express whether the use of reconnaissance has involved more or less effort than would have been expended otherwise, and whether they feel the overall effort in setting up and evaluating reconnaissance results has turned out to be worthwhile. In addition, it may be informative to obtain the subjects' opinions on the likelihood of reaching a better result within a constrained further period of exploration. It may be felt, for example, that there could well be better results available but that they represent 'needles in a haystack' that one could not realistically hope to find.

8.3.3 Coping with far-flung explorations

The use of reconnaissance becomes particularly challenging but potentially rewarding in explorations for which the best results are spread widely over a large result space. I claim that the availability of reconnaissance will be seen to encourage users to keep working on a number of promising threads of the exploration, rather than shutting down to a single direction on the basis of early feedback.

Chapter 9

Review of the thesis contributions

Chapter 8 proposes various research directions that continue from where the research in this thesis leaves off. In this concluding chapter I recapitulate the contributions already made by this research, and assess their significance.

The contributions can be grouped into the following levels, addressed in the next three sections of the chapter:

1. identification of the widespread problem of *under-informed outcome* in computer-based explorations, and analysis of how it arises;
2. the concept of *reconnaissance* as a way of delegating exploration activity to a computer;
3. novel ideas and experience arising from the practical realisation of reconnaissance support for a particular domain.

Finally, section 9.4 lists the main points in a compact form.

9.1 Identification and analysis of under-informed outcome

The factors contributing to the problem of *under-informed outcome* in computer-based explorations were investigated. A wide range of systems was reviewed, and although they support a great variety of exploration strategies it was illustrated, with the help of the *illumination zone model*, that all the strategies impose a substantial burden of effort on any user hoping to pursue a thorough exploration. The burden can be characterised as principally due to one of the two following problems:

- **Excessive formalisation**

  One kind of burden arises because the exploration is formalised in a way that forces the user to make consolidation commitments before being ready to do so. The various refinement-based strategies, for example, are geared towards exploration progress that moves monotonically towards some ideal result by narrowing the range of results being considered at each iterative step. Many strategies help the user to maximise the discriminatory effectiveness of each step, which has the apparently desirable outcome of minimising the number of steps needed to reach the supposed goal. But these highly selective choices reject large areas of the result space that, although not appearing to be useful at the time, might turn out to contain some of the best results.

  If a user is not satisfied with following an ever-narrowing search, but would rather explore a range of alternative avenues, these systems are quite obstructive; the user has to backtrack explicitly to an earlier point and then start out in another direction. And although it is reasonably common for a system to incorporate a search history sufficient to allow backtracking, few system designers tackle the efficiency and interface-model issues involved in supporting multiple-branch undo/redo. This acts as a further disincentive to users against embarking on exploration of alternative paths: unless it turns out that the best result is found in the last direction to be examined, the user will have to remember which path contained the preferred outcome and to re-trace the steps that led to it.

- **Insufficient formalisation**

  The opposite problem, of insufficiently expressive formalisation, also causes problems because too much of the burden is left to the user. This is the case in domains for which the designer is unable or unwilling to predict users' goals, and in order to avoid providing a system with an unhelpful, over-constrained set of facilities instead leaves the user free to guide the whole exploration by hand. This thesis identifies two sub-activities of exploration that are particularly laborious if the user is unable to delegate them to the computer: the specification of a large range of alternative result-space regions to illuminate, and the evaluation of trade-offs between large numbers of results. So even when a system supports rapid production and display of results in accordance with the user's requests, the user still has to do all the work of deciding what to ask for, evaluating what is produced, trying to figure out what change is needed, and applying the appropriate reformulation command to see if it helps. In short, a user's difficulty in pursuing a well-informed exploration in these systems is not due to lack of paths to pursue, but being spoilt for choice—and not being able to tell in advance whether taking one of the many available directions will lead to a more interesting result than the others.

One of the puzzling aspects of this work was the difficulty I had in finding any acknowledgement of the issue, other than in the general 'decision making' literature. But the studies of LaTeX usage suggested some factors that would help to explain the lack of concern:

- Users working with systems that engender under-informed explorations often pursue their own tactics, that can be surprisingly effective.
- Users are sometimes guilty of overconfidence in the results they obtain, feeling that there is nothing substantially better within worthwhile reach.
- Users are knowingly accepting results that could probably be improved upon—e.g., resigning themselves to satisficing because anything more would take too much effort.

In performing this analysis the *illumination zone model* has been most valuable, principally in enabling the diverse systems reviewed in chapter 3 to be analysed under a single set of terms and thus allowing a range of common problems and limitations to be described. Since the model was designed in tandem with the surveys and the early design experiments, it is itself a result of this thesis.

9.2 Reconnaissance: a 'middle ground' in formalisation

9.2.1 What has been shown

The main contribution of the thesis is the identification of reconnaissance as a useful metaphor for approaching a wide range of computer-based exploration tasks.

Given...

any computer-based exploration for which the user can formally describe summarisable result illumination in terms of a range, content, and display format

...then you can obtain...

collated summaries of results within requested illumination ranges

...giving...

the opportunity for the user to evaluate and compare a large number of results, without having to make a corresponding number of separate requests nor to view all the results in detail.

The key factors that boost the value of reconnaissance in relation to the effort required to use it are as follows:

1. **Only a low degree of formalisation is required.**

   Reconnaissance can be carried out even if the task as a whole is too opportunistic for the user's goals to be formalised in full. As long as the user can define formally some feature of a result that would make it count as interesting, the computer can be sent to illuminate the directions ahead. Where some direction appears to offer a substantial advantage over what has been obtained so far, the user can pursue it; when no such advantage seems to be available, and if the user cannot think of other directions to check, the illumination provides reassurance that the current result is the best available.

   The 'reconnaissance' analogy fits this form of illumination because it is like deploying scouts who are tireless and, although naive (i.e., can't understand the overall goals of the task), can be asked to report useful details of whichever regions they are sent to explore.

2. **Benefits are based on increased rather than exhaustive coverage.**

   Even if it is not practical for a user to request a fully exhaustive illumination, the reconnaissance will produce more information than the equivalent effort expended on obtaining individual results. For example, in a complex design task there can be literally hundreds of options and value adjustments, resulting in millions of plausible combinations. But even if a user requested permutations of the options on just four parameters, each having a few alternative values, this might result in hundreds of distinct trials—quick for the computer to evaluate, but far beyond what the user would have time or patience to pursue by interactive selection and examination of the results.

   A reconnaissance framework thus allows the explorer to keep many diverse options open, comparing a variety of outcomes to gain an understanding of available trade-offs. This engenders a feeling of being able to experiment, examining whichever parts of the result space are felt to be potentially valuable.

Reconnaissance is not a 'silver bullet' that can ease all tasks even within a single given domain; it just extends the scope for opportunistic delegation beyond the standard delegatable services. Referring back to figure 2.1 on page 12, the addition of reconnaissance may be represented as shown in figure 9.1. As in the original figure, the thickness of region that an arrow has to pass through symbolises the extent of the task that is handled at

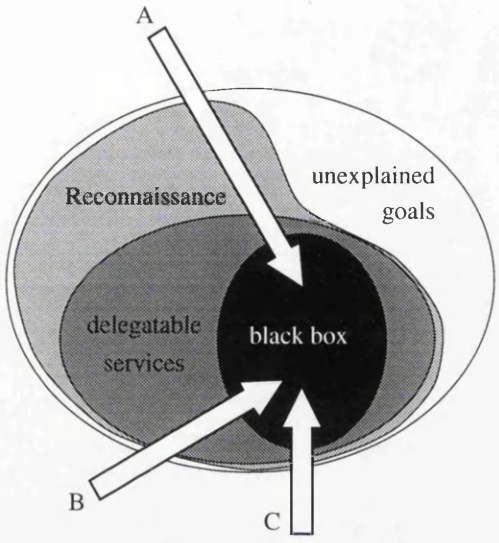


Figure 9.1 An updated version of figure 2.1, showing the effect of reconnaissance.

that level. For some tasks, such as those represented by B and C, reconnaissance may be unable to provide any more assistance than the black-box facilities; for task A, by contrast, reconnaissance may allow full delegation of exploration activities that the user previously had to carry out using unformalisable actions.

In general, the assistance offered by reconnaissance is that of helping a user understand what *results* are on offer, rather than just being aware of the available *processing options.* Wherever any computer system presents its user with a range of options, there may be a chance to apply reconnaissance to illuminate the results that could be obtained by selecting each available option.

Analysis in terms of cognitive dimensions

One way of explaining the benefits offered by reconnaissance is by reference to some of the *cognitive dimensions* first proposed by Thomas Green for describing the properties of artifacts as perceived by their users (e.g., see Green 1990 and 1991; Hendry & Green, 1994; Modugno, Green & Myers, 1994). The 'dimensions' are still somewhat experimental in nature; their names and definitions are being refined in an ongoing effort to pin down the crucial properties that it would be useful for them to capture. Three of the dimensions proposed so far that seem to be particularly relevant to the reconnaissance approach are those called *premature commitment, repetitious viscosity* and *side-by-side ability:*

- **Premature commitment**[1] is a measure of the user having to *commit* to choices—to the extent that it is laborious to change one's mind later—before being ready to do so. Premature commitment is undesirable. For example, an unusually constrained feature of the otherwise highly flexible and incremental Smalltalk development environment is that when creating a new class one must identify the superclass from which it is to inherit (and the superclass must already have been defined); this constitutes commitment to a class's position in the hierarchy, at a time when one may wish to leave this open to experimentation.

  In explorations, premature commitment is most evident in the practice of having to discard a large portion of the result space because only one direction of search focus can be pursued at a time. It is therefore at the root of the effort–accuracy tradeoff; a tool with high premature commitment makes it impossible to be accurate without expending enormous effort. Correspondingly, reduction or abolition of premature commitment is the main goal of reconnaissance support, and is tackled by allowing users to keep options open and to experiment.

- **Repetitious viscosity** is one of two identified forms of *viscosity* (the other being *knock-on viscosity*). Viscosity is a measure of the user effort required to impose changes in the system—the more 'viscous', the more effort is involved. Repetitious viscosity, in particular, refers to the effort in making what the user regards as a single change, but for which the system requires that the user carry out a large number of repetitive operations. In most cases, low viscosity is desirable. For example, if a user wishes to increase the amount of white space appearing above every paragraph in a document, a system that required each paragraph in turn to be adjusted would be characterised as having high repetitious viscosity for this task, compared with a system in which a single 'paragraph style' could be adjusted to affect all paragraphs in one go.

  In the case of reconnaissance, the task whose viscosity is under scrutiny is the illumination of a region of the parameter space that is of interest to the user. Many existing exploration systems exhibit high viscosity in that each point in the region must be requested explicitly by the user. Reconnaissance support allows the user to define the points implicitly by specifying the bounds of the region; as long as this specification requires less effort than explicit visits to all the points, the reconnaissance approach scores better on viscosity.

[1] In (Hendry & Green, 1994) an alternative but closely related dimension is proposed, called *imposed guess-ahead*.

(The reservation about low viscosity being desirable 'in most cases' deals with the standard desire to let a user perform operations with the least effort possible. But in special situations it is desirable for some operations to be hard to perform—for example, when they have dangerous or far-reaching consequences (such as firing nuclear weapons). In such cases high viscosity is a good way to force the user to consider the actions carefully.)

- **Side-by-side ability** is a measure of the ease of obtaining side-by-side views of results that a user wishes to compare. High side-by-side ability is therefore desirable. For example, many word processors maintain a single view of the working state of the current document; a user who wants to experiment with a change to the document must observe and remember the features that are important in the current state, then make the change and re-examine the document to see what effect the change has had—comparing it with the remembered pre-change state. For anything other than very simple or localised differences, such 'low side-by-side ability' imposes an enormous burden on the cognitive capabilities of the user. This burden multiplies dramatically when there are large numbers of results to be compared, and/or large numbers of independent individual effects to be traded-off within each result.

  The use of reconnaissance to bring together, onto a single display, a collection of the result properties that the user wants to see, for a whole set of results that are of interest, constitutes a major improvement in side-by-side ability.

9.2.2 Potential scope of applicability

As described in section 4.5, reconnaissance can be applied to any computer-based task for which the user can be provided with facilities for opportunistically requesting and accumulating illumination of potentially interesting parts of the task's result space. This requires a means of defining a reconnaissance range, and specifying a form of reconnaissance content that a) reveals information that will help the user judge the likely value of each result, and b) can be represented in summary form so that many results can be compared in a single illumination display.

This broad definition makes reconnaissance applicable under the following conditions that raise problems for other approaches:

1. **Results not instantaneously available**

   Reconnaissance can be applied to explorations in which the results of interest are not previously generated and available for retrieval, and cannot be generated instanta-

neously. These explorations are therefore not suited to techniques such as the Dynamic Query approach, in which the instantaneous feedback is crucial in letting the user get a feel for the distribution of results, or for the various forms of Information Visualizer tool that build a complete mapping of the result space for the user to browse and/or filter at will.

2. **Result space has no obviously useful inherent mapping**

   This is related to the previous point. Reconnaissance can assist in domains for which it is not practical to generate a single synthetic mapping of the result space, which could then be explored by the user.

   Some tools function by building a mapping that is known in advance to be useful for the given result domain—for example, the Information Visualizer tools. In other cases a result-space mapping is derived on the fly, such as in Boz (Casner, 1991), but these still rely on advance knowledge of how many results are to be presented, and how they are related to each other. Boz in particular relies on a formalisation of the user's overall goals in the exploration—which may not be practicable, as discussed in point 4 below.

   Since value can be derived from reconnaissance even if the results are mapped onto a generic, abstract presentation (such as a parallel-coordinates display), according to opportunistically selected measurable properties, reconnaissance can assist in any domain for which results can be summarised as a set of variables.

3. **Interesting results might not be clustered in the input parameter space**

   Many exploration tools provide facilities for the user to narrow the search to a range describable by a single abstraction (or single conjunction of abstractions) such as constraints on individual parameter values—for example, a database query or a particular location in a pre-mapped result space. Some include powerful facilities for a user to browse and compare results—either within the narrowed range, such as the lists of results provided in Helgon, or by steering the illumination range to cover alternative regions, such as the Dynamic Query tools' live variation of one range-defining parameter at a time. But even these tools still impose a large burden of effort in obtaining and comparing results that exist in disparate regions of the result space.

   Reconnaissance is not subject to this limitation; it can be used to bring together results with arbitrarily divergent descriptions. For example, whereas DQ tools enable exploration along a single dimension of the parameter space, a reconnaissance tool can support exploration of a region made up of multiple interacting dimensions.

The approach of Mutator is also suited to simultaneous variation over multiple parameters. Its randomised 'mutation'-based input variation does not provide the opportunity to be as systematic as an explicit reconnaissance foray, but has the advantage of being scalable to parameter spaces of arbitrarily large dimensionality—where explicit reconnaissance would become unmanageable. Indeed, in some domains a mutation-based definition of illumination range would be the only practical way to request reconnaissance.

4. **Exploration goals not known in advance, nor necessarily discovered until the end of the exploration**

   Reconnaissance is specifically intended for explorations in which the user might be unable to express, at any stage (let alone in advance), the goals of the exploration. This affects the ability of the designer of the computer system to provide facilities for assisting the exploration.

   Some systems do not support explicit expression of goals, but provide a single form of search support that is assumed to be suitable for all the kinds of exploration users may wish to perform. Some support explicit goal formalisation—notably Boz, which relies on the user being able to express the goals to a sufficient degree for the system to build a tailored result view suited to the achievement of those goals. Although users of Boz have some freedom to tailor the tool to the particular exploration in progress, it is only within the range of goals foreseen by the designer of the system. It is essentially a toolkit providing a high level of task abstraction, and therefore runs the risk of failing to be sufficiently flexible for some users' needs.

   Reconnaissance support is at a relatively low level of task abstraction, and therefore allows greater flexibility in the construction of a visualisation suited to some particular exploration. Scouts who are sent on reconnaissance missions can provide useful information whether or not the person who sends them knows in advance how their information will be used.

9.3 Option-space reconnaissance using parallel coordinates

9.3.1 What has been shown

The implementation reported in chapter 7 can be regarded as a 'proof of concept' for the practical ability to present reconnaissance to a user, and in particular the suitability of the parallel coordinates presentation technique as a basis for a powerful integration of control

and presentation components. Even though the implementation was only a prototype supporting a limited range of options and measurements, and requiring subjects to learn a specialised form of additional markup language, the potential usefulness of the approach was acknowledged by existing users of the underlying application.

Given...

a domain in which reconnaissance-based explorations are appropriate, and for which the reconnaissance range can be specified as an *option space,* and reconnaissance content as a *measurement space*

...then you can obtain...

a dynamically built multivariate display, based on the *parallel coordinates* presentation technique, that relates all the inputs and measurements of the reconnaissance results

...giving...

realisation of the benefits hypothesised for reconnaissance: increased willingness of users to investigate alternative results, including performing trade-off analyses among multiple results.

Bonus finding: a generalisation of reconnaissance

I had imagined that the implementation would show that in a rich exploratory environment users will enthusiastically change their working practices, to perform more thorough explorations, if they are given facilities that make it cheap to try multiple options and to revert to the original state if none of them seems to make an improvement.

It turned out not to be as straightforward as that: where existing systems offer particular facilities and particular constraints, people develop their own strategies. In the case of the existing tools this meant that although the users' explorations were not thorough, they could still be surprisingly effective. But in the case of the reconnaissance facilities, the test subjects spontaneously saw them as providing opportunities that I had not foreseen: a general ability to define a region of results that are potentially interesting, and measurements that summarise those results—not just for looking *forward* in an exploration, but also for re-visiting and re-evaluating diverse regions that have not yet been ruled out of the search.

9.3.2 Demonstration of enabling techniques

Implementing reconnaissance required various design decisions on an appropriate form of interface. The following are felt to be important enabling features:

Opportunistic construction and extension of the interface

The interface is built up by the user as the exploration proceeds, using a 'toolkit' of components for expressing the options and the measurements to be used. Working with low-level facilities brings flexibility and ease of understanding, but the vital ingredient to making the toolkit approach workable is the 'framework' that supports it, allowing the user to assemble higher-level abstractions from the components. It is these higher abstractions that enable an escape from the repetitious viscosity involved in always working at a low abstraction level.

In accordance with the illumination zone model, these facilities allow InteracTEX to incorporate explicit support for opportunistic delegation of each of the three identified exploration sub-tasks.

Equal Opportunity

Several researchers have identified the benefits in interfaces that suppress a needless distinction between what is an 'input' to an activity and what is an 'output'. Draper (1986) discusses the issue of *inter-referential input/output*, while Runciman and Thimbleby (1986) and Thimbleby (1990) refer to the goal of *equal opportunity.*

Opportunistic exploration can be regarded as a cooperative problem: some bits are to be done by the user, other bits by the computer. Because neither the designer nor the user of a system can know in advance which aspects of a given exploration are going to be formalised, there is a benefit to avoiding premature division of the activity into some aspects that are considered the domain of the computer and others that are considered the domain of the user. Instead all aspects of the interface should work together to define the results of what has been done so far, and what can be done next, such as by allowing constructive feedback.

Semiformal explorations often include situations in which information that first appears as a result can later be useful as an input—this is a principal feature of *retrieval by reformulation,* for example. Ahlberg and Shneiderman (1994) refer to this form of inter-referential I/O as *output-is-input*: the elements of the display are sensitive to user input, allowing efficient use of output as feedback into the process.

The limiting case of inter-referential I/O is *equal opportunity,* when all interaction is carried out through interface objects that can be controlled and acted upon either by the user or by the computer. In practice this involves ensuring that any input value the user may adjust should also be adjustable under control of the computer, and any result value the user can read off should also be usable by the computer in subsequent calculations.

The InteracTEX interface supports equal opportunity in the following ways:

- The user is able to filter the displayed results, and even to request further reconnaissance, using the components of the result display itself.

  This is one factor that contributes to the availability of *opportunistic delegation*: the user is able to move at will between informal evaluation of displayed result properties, and the use of the same display for formal actions such as filtering and highlighting results or requesting illumination of additional regions of the result space.

- Scales of the same form are used for showing the input options and the measurements.

  The ability to filter results on the basis of measurement features allows a form of declarative interface, even though the underlying implementation for illuminating results may be strongly procedural. Clearly it is declarative only in a limited sense, since the results will only be found if the user has selected a sufficient range of input options to explore, but designers of software with reconnaissance support can help by structuring their code to perform the minimum processing needed to evaluate the requested measures.

9.4 Conclusions

The main contributions of this thesis are summarised below.

9.4.1 Problem identification and analysis

1. The identification of the general problem of *under-informed outcome* in computer-based explorations, and its characterisation in terms of subjectively rational human decision-making behaviour.

2. A breakdown of explorations into the activities of *illumination, evaluation,* and *consolidation,* allowing the construction of the *illumination zone model* of computer-supported exploration to help identify causes of excessive effort in illumination and evaluation, and the unhelpful limits of consolidation support.

3. A categorisation of a wide range of existing computer-supported exploration tasks, showing the various ways in which each imposes an unacceptable burden on a user wishing to engage in a thorough exploration.

9.4.2 The Reconnaissance concept

1. A proposal for a high level of exploration support by a computer system which, by being based on the *reconnaissance* analogy, makes effective use of the ability to delegate only simple 'scouting' tasks.

2. The development of a check-list of features that render an exploration activity suitable for reconnaissance support, and hence the identification of various kinds of task that are not explicitly based on exploration but, because they may be regarded as leading to choices from the domain of a notional *result space,* can be supported by use of reconnaissance.

3. The identification of a broader issue, related to the facilities that are used to support reconnaissance, involving the general instrumentation of output from a complex processing task.

4. Various proposed directions for further research into the applicability of reconnaissance, and into the design of mechanisms for making reconnaissance easier to implement.

9.4.3 Option-space reconnaissance using parallel coordinates

1. The identification of a sub-class of task whose result space can be explored to a useful level of thoroughness by enumerating the points in an *option space.*

2. A proposal and implementation of an interactive computer-generated display, based on the *parallel coordinates* multivariate presentation technique, to serve as an integrated means for requesting and collating results in a reconnaissance-based exploration.

3. A demonstration of how reconnaissance can be applied productively within an existing computer-based activity in which the user's implicit search for a satisfactory outcome is not usually supported as an exploration.

Appendix A

Usage of the term 'formal'

There has been something of a divergence of interpretations of the term 'formal' in computing literature.

The relevant part of the definition in the Concise Oxford Dictionary (Eighth edition) is the 7th of the listed adjectival senses, namely:

> **formal**
>
> - in accordance with recognized forms or rules.

One may trace the divergence in usage by observing the implication of the kind of 'rules' that are to be followed:

- Potter, Sinclair and Till (1991), in their book 'An Introduction to Formal Specification and Z', give the interpretation that is understood in the field of computing science broadly referred to as 'formal methods'. They state:

  > 'What is needed [as a program specification language] is a vehicle which offers the precision of a programming language, but without the prescriptive aspect whereby every detail of data structure and algorithm must be provided. We also want to be able to conduct rigorous arguments to establish desirable properties of our specification. This combination of precision and the ability to argue rigorously is what we mean by formality.'
  > (p.7)

These practitioners wish to *reason* about program specifications, so they require that the specifications be presented in a language with a well defined set of manipulation rules. When the mathematical specification of a program can be proved, by these rules, to conform to the mathematical specification of its purpose, a proper implementation of that specification is guaranteed to be correct in a corresponding sense.

- In his paper 'On Visual Formalisms' Harel (1988) introduces the *higraph,* a graphical specification notation derived from a marriage between connected graphs and Euler circles, as a means of representing the topologies of systems such as computer programs and data stores. In the conclusions to this paper he states that:

  > '... the intricate nature of a variety of computer-related systems and situations can, and in our opinion should, be represented by *visual formalisms:* visual, because they are to be generated, comprehended, and communicated by humans, and formal, because they are to be manipulated, maintained, and analyzed by computers.'
  > (original emphasis)

  Harel stresses the contrast between visual formalisms (such as the higraph) and other diagrammatic methods that are 'described in a manner that is devoid of semantics, and can therefore be used at best as informal aids'. His view of formality can be seen to admit a weaker interpretation than the previous one: he does not require the notation to have the power to prove properties of the system, but merely that it should only support manipulations that have a meaningful correspondence with the modelled world. It is the correspondence, rather than the manipulation itself, that needs to follow clearly defined rules.

- In the same year as Harel's paper, Lai, Malone and Yu (1988) gave the following definition of a *semi*formal system:

  > 'We define a semiformal system as a computer system that has the following three properties: (1) it represents and automatically processes certain information in formally specified ways; (2) it represents and makes it easy for humans to process the same or other information in ways that are not formally specified; and (3) it allows the boundary between formal processing by computers and informal processing by people to be easily changed.'

  This level of usage (which has been taken up by many people in the field of human–computer interaction) is superficially the same as Harel's, but it admits an even less constrained interpretation. When information is stored in a computer, any operation

on that information that can be carried out by the computer by following a set of rules (for which one may read 'instructions') can be regarded as a 'formalised' operation. It is just *assumed* that the rules, and hence the operations, have a valid impact on the information as far as the user is concerned.

Thus one arrives at the definition of 'formal' that is intended to be understood in this thesis, where it is applied to information or to tasks:

formal

- represented on a computer in a way that allows operations that are meaningful to the user to be performed computationally.

Appendix B

Questions for measuring LaTeX experience and approach

The following is a reproduction of the prompting sheet I used during 'Phase 1' testing of each subject, to establish their level of use of LaTeX and their approach to ensuring high-quality outcomes.

```
[ Phase 1A start time:        ]

   A1 what kinds of document they produce, whether these involve LaTeX,
      and what kind of size. Also: would they say this list is in
      increasing order of their concern for how the document looks?
      A. informal notes for own consumption
      B. docs for discussion/reference within local peer group
      C. frequently-updated or replaced docs for dissemination among
         larger groups, e.g. newsletter, release notes, collaboration
         discussion documents
      D. long-lived official documentation, for wide distribution
      E. thesis
      F. conference papers
      G. books
      H. other (e.g. personal letters, shopping lists,...)

   A1a whether they find themselves using LaTeX to handle documents from
       elsewhere
```

A2 how much and why they use TeX/LaTeX
- A. frequency: documents per week
- B. is it used readily, or only if it is the only available option?

A3 what level of interaction they use. Would they be happy to:
- A. run LaTeX on existing docs, with/without any changes
- B. use basic set of LaTeX commands and styles - perhaps modifying templates
- B'. pull in, and use on trust, interesting macros or styles
- C. use a wide range of LaTeX commands. For a start:
  "For each of the following commands, do you know what it does? Would you be happy using it?"
  \newpage \marginpar \sloppy \protect \nopagebreak[3]
- D. sometimes create/redefine simple LaTeX commands
- E. use pieces of pure TeX
- F. regularly create/redefine complex LaTeX, or even TeX, commands or macros
- G. write or update entire LaTeX/TeX document styles
- H. other

Phase 1B - sample document formatting: [Start time:]

B1 Get a document that is already formatted. Something like 10 pages, with some 'interesting' things like lists, figures, examples. Get the subjects to comment on:
- A. how seriously they took the formatting
- B. how well they think they have done
- C. what they would prefer to be different
- D. what's stopping them from making the change

B2 Change its style, to throw the formatting. Print out the changed version. Before taking it away, get the subjects to look at it - ask and record:
- A. how happy they would be to submit it as is
- B. which things they now regard as unsatisfactory
- C. what they would bother trying to change

B3 Get them to work on improving the doc, using their own normal setup, until they feel it has reached a stage that they would be happy to present as 'not official, but still worth caring about'.
- A. How long did they feel was worthwhile working on it?
- B. What kinds of change did they make?

B4 Print out the document. Before taking it away, ask:
   A. how happy they are with the end result
   B. what they would still like to change about it. Get some idea of which are the most serious problems, and which they regard as nice-to-haves.

B5 Suggest that they might be asked to come up with flawless formatting, e.g. for a book. Ask:
   A. How long they would expect to have to work on the doc to get it looking just right.
   B. What they feel they would have to fix
   C. How they would go about it.

B6 in general, what kinds of action would they feel they had to be prepared to take to fix up a document?
   A. changing the text to fit a space
   A'. changing local formatting, e.g. adding specific page breaks
   B. gross change to the formatting e.g. line separation, indention
   C. pulling in additional macros, written by themselves or others
   D. defining new macros
   E. tweaking the penalties

B7 How much help do they find the diagnostic info printed out by TeX/LaTeX?
   A. (When) do they look at it?
   B. which errors/warnings would they take seriously?

B8 (if needed) "To what extent do you agree or disagree that the following statements apply to your use of LaTeX?"
   A. TeX/LaTeX provides, at some level, everything necessary for good typesetting of the documents I produce
   B. I like to use TeX/LaTeX because of the feeling of flexibility being available if needed
   C. I don't need to worry about typesetting, because some expert will have produced a style file that takes care of it.
   D. The style files I use seem to do a good job
   E. I often make manual changes to improve the format of my documents

[Phase 1B end time:]

Points to note regarding subject's normal processing setup:

1. what tools they use (how many edit windows, what kind(s) of viewer)

2. whether they are happy to work with online viewers, or need to see the document printed out

3. how the task is attacked:
   - working through in page order, or trying whole thing at a time
   - only processing part of the document
   - rapid re-saving (small changes) and formatting, or fixing lots of things and trying the whole thing again
   - relying on diagnostics (or lack of them), or always looking at a viewer?

4. what they find especially good/bad about the TeX/LaTeX setup they have

Phase 1C: Demonstrate the same re-formatting task being carried out with the help of reconnaissance, then request feedback as follows:

C1. would reconnaissance facilities like these be useful, or essentially a waste of time?

C2. would use of reconnaissance lead to better documents?

C3. would the subject explore a larger range of alternatives with the reconnaissance facilities than without?

C4. (a hunch): were there any occasions where reconnaissance showed that NONE of the alternatives being tried seemed to offer a better result, thereby enabling the subject to cut his/her losses and try a different approach?

Glossary

Terms that have specialised meanings within the thesis are gathered below. Because many of the terms form related groups, it has been decided to show them in their groups to assist the reader in combining and contrasting their individual contributions. The term definitions also include illustrative examples. In line with the broad applicability of the thesis the examples are drawn from rather different kinds of task: (1) a traveller's queries for an available seat on a long-haul flight, and (2) an architect's exploration into the creative possibilities for the layout of an apartment.

under-informed outcome (p.3)

The end-state of an exploration that the explorer has terminated even though the domain could contain results significantly preferable to what has been found. *e.g., Accepting the compromises embodied in the first apparently reasonable flight, or floor plan, without having investigated other queries based on less painful compromises.*

opportunistic exploration (p.3)

An exploration in which the explorer can predict neither the course nor the end goals, because they are both contingent on what is found in the course of the activity. *e.g., Looking for a 'good' available flight, or a 'good' floor plan for the apartment.*

Opportunistic exploration is made up of the following principal activities:

illumination

The explorer requesting and obtaining a view of some part of the exploration domain. *e.g., A query to a flight database, or a set of constraints on the apartment such as rooms it must include.*

I identify the following characteristics of an explorer's request for illumination:

range specification

The portion of the domain to be illuminated. *e.g., All flights on 3rd November from Glasgow to Los Angeles; 50 possible floor plans with two bedrooms.*

content specification

The features that are to be reported. *e.g., Airline, departure and arrival times, lowest available fare on the flight; outline plans marked with proposed room function.*

display format

How illumination is to be presented. *e.g., As a list of flights in reducing fare order; a grid layout of the alternative floor plans.*

evaluation

The explorer extracting from the **illumination** some information regarding the current exploration. *e.g., Examining the details of flights or apartment plans, to see what is good or bad about the ones that have been presented.*

consolidation

Use of accumulated information by the explorer to decide whether the exploration is finished and, if not, where to explore next. *e.g., Deciding to look for a cheaper or quicker flight by requesting a different departure date; deciding to try plans with fewer but larger rooms.*

result space (p.3)

An exploration domain consisting of the (real or notional) enumeration of all the possible results that can be generated by some computer-based system. *e.g., The notional collection of all possible result displays (typically ranked lists containing a few flight details), corresponding to all permissible queries that could be submitted to the flight retrieval system; the notional collection of all floor plans corresponding to the given fixed constraints such as overall shape and minimum room size.*

effort–accuracy tradeoff

effort

A measure of the amount of work performed by the explorer over the course of an entire exploration. *e.g., The time and cognitive effort involved in extracting details from the results, to allow comparison between all the examples that are seen.*

accuracy

A measure (often qualitative) of the closeness of a chosen outcome to the 'ideal' outcome for the given combination of domain and user goal. *e.g., The explorer's satisfaction with the price and convenience of the selected flight, relative to the flight that—had it been found—would have given the highest level of satisfaction.*

illumination zone model (p.16)

A schematic representation of the processing components and information stores of a system for exploring some **result space**. The model includes the following elements:

fixed base data

Data that do not change during the course of the exploration, and have a crucial role in defining the **result space** being explored. *e.g., The flight-booking database; a set of planning rules, such as some kinds of room requiring an outside wall.*

illumination specification

The user-chosen instructions that direct the part of the **result space** being examined at the current stage of an exploration. *e.g., The current query, or a fixed feature in a floor plan around which other elements are to be arranged.* Depending on the domain, part of the illumination specification might constitute the *current viewpoint.*

result-space illuminator

A processing device that generates **illumination** of the **result space**, by processing the **fixed base data** in accordance with the **illumination specification**. *e.g., The database retrieval engine; the layout generator.*

illumination zone

All illumination information currently viewable by the user. *e.g., The currently retrieved set of flights, or suggested set of designs.*

display specification

The user-chosen instructions that direct the mapping of information currently held in the **illumination zone** into a user-viewable form. *e.g., Paging or scrolling position for flight results; perhaps zooming and view-detail specification for floor plans.*

display generator

A processing device that generates viewable (and possibly interactive) output from the **illumination zone** in accordance with the **display specification**. *e.g., The database 'front end'; a graphical floor-plan browser.*

exploration context

The collective information that represents the state of an exploration currently in progress. This information comprises the contents of the **illumination zone**, the two specifications that determine its content and display characteristics, and any maintained search history information.

command processor

A processing device that interprets commands from the user, to adjust the **exploration context** and hence to cause a change to the information presented in the output.

(result-space) reconnaissance (p.76)

Reconnaissance in a result space is just a particularly useful form of **illumination**, characterised by:

- flexible, opportunistic **range specification**
- flexible, opportunistic **content specification** that produces summaries of results
- a **display format** in which all the summaries are collated into a single display.

option space (p.124)

A particular kind of **range specification** based on enumeration of combinations of discrete values from a set of dimensions. *e.g., All combinations of flight date and destination, picked from four alternative dates and three alternative destinations.*

References

Adams, J. L. (1987) *Conceptual Blockbusting.* Penguin Books.

Ahlberg, C. and Shneiderman, B. (1994) 'Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays'. In *Proceedings of ACM CHI '94,* 313–317.

Ahlberg, C. and Truvé, S. (1994) 'Designing, Implementing, and Evaluating a Tightly Coupled Dynamic Queries System'. SSKKII Technical Report SSKII94.08, SSKKII, University of Göteborg, Sweden.

Ahlberg, C., Williamson, C. and Shneiderman, B. (1992) 'Dynamic Queries for Information Exploration: An Implementation and Evaluation'. In *Proceedings of ACM CHI '92,* 619–626.

Ahlberg, C. and Wistrand, E. (1995) 'IVEE: An Environment for Automatic Creation of Dynamic Queries Applications'. Submitted to ACM CHI '95.

Berry, D. C., and Broadbent, D. E. (1986) 'Expert systems and the man-machine interface: part 2: the user interface'. *Expert Systems: The International Journal of Knowledge Engineering* **4**.

Bertin, J. (1981) *Graphics and Graphic Information Processing.* de Gruyter, Berlin.

Buzan, Tony (1982) *Use Your Head.* BBC Books, London.

Card, S. K., Moran, T. P., and Newell, A. (1983) *The Psychology of Human Computer Interaction.* Lawrence Erlbaum, Hillsdale.

Card, S. K., Robertson, G. G. and Mackinlay, J. D. (1991) 'The Information Visualizer, An Information Workspace'. In *Proceedings of ACM CHI '91,* 181-188.

Casner, S. M. (1991) 'A Task-Analytic Approach to the Automated Design of Graphic Presentations'. *ACM Transactions on Graphics* **10**(2), April 1991, 111-151.

Chatterjee, A. (1995) *Visualizing Multi-Dimensional Polytopes and Topologies for Tolerances.* Ph.D. Thesis, University of Southern California.

Chernoff, H. (1973) 'Using Faces to Represent Points in k-Dimensional Space'. *Journal of the American Statistical Association* **68**, 361–368.

Chimera, R. (1992) 'Value Bars: An Information Visualization and Navigation Tool for Multi-attribute Listings'. In *Proceedings of ACM CHI '92,* 293–294.

Chomut T. (1987) *Exploratory Data Analysis Using Parallel Coordinates.* M.Sc. Thesis, UCLA Computer Science Dept. Also IBM LA Scientific Center Report 1987-2811.

Cleveland, W. S. and McGill, M. E. (eds.) (1988) *Dynamic Graphics for Statistics.* Wadsworth and Brooks/Cole, Statistics/Probability Series.

Cleveland, W. S. and McGill, R. (1984) 'The Many Faces of the Scatterplot'. *Journal of the American Statistical Association* **79**, 807–822.

Curtis, R. and Scarfone, S. (1992) 'XFace, an X tool for presenting multivariate data, and its use with software metrics'. In *Proceedings of the Eleventh Annual International Phoenix Conference on Computers and Communications,* 525–530.

Cypher, A. (1991) 'Eager: Programming Repetitive Tasks by Example'. In *Proceedings of ACM CHI '91,* 33–39.

Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., and Turransky, A. (1993) *Watch What I Do: Programming by Demonstration.* MIT Press, Cambridge, MA.

Desai, A. and Walters, L. C. (1991), 'Graphical Presentations of Data Envelopment Analyses: Management Implications from Parallel Axes Representations'. *Decision Sciences Journal* **22**(2), 335–353.

De Soete, G. (1986) 'A Perceptual Study of the Flury-Riedwyl Faces for Graphically Displaying Multivariate Data'. *International Journal of Man-Machine Studies* **25**(5), 549–555.

Draper, S. (1986) 'Display Managers as the Basis for User-Machine Communication'. In Norman, D.A. and Draper, S.W. (eds.), *User Centered System Design.* Lawrence Erlbaum Associates, Hillsdale, NJ, 339–352.

du Boulay, B., O'Shea, T., and Monk, J. (1981) 'The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices'. *International Journal of Man-Machine Studies* **14**.

Eick, S. (1994) 'Data Visualization Sliders'. In *Proceedings of ACM UIST '94,* 119–120.

Eisenberg, M. and Fischer, G. (1994) 'Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance'. In *Proceedings of ACM CHI '94,* 431–437.

Fischer, G., Lemke, A. C., Mastaglio, T. and Morch, A. I. (1991) 'The Role of Critiquing in Cooperative Problem Solving'. *ACM Transactions on Information Systems* **9**(3), April 1991, 123–151.

Fischer, G. and Nieper-Lemke, H. (1989) 'HELGON: Extending the Retrieval by Reformulation Paradigm'. In *Proceedings of ACM CHI '89,* 357-362.

Flemming, U. (1989) 'More on the representation and generation of loosely packed arrangements of rectangles'. *Planning and Design* **16**, 327–359.

Flemming, U., Coyne, R.F., Glavin, T., Hsi, H. and Rychener, M.D. (1989) 'A generative expert system for the design of building layouts.' 1989 Report Series, Engineering Design Research Center, Carnegie Mellon University.

Flury, B. and Riedwyl, H. (1981) 'Graphical representation of multivariate data by means of asymmetrical faces'. *Journal of the American Statistical Association* **76**, 757–765.

Frawley, W. J., Piatetsky-Shapiro, G. and Matheus, C. J. (1992) 'Knowledge Discovery in Databases: An Overview'. *AI Magazine* **13**(3), Fall 1992, 57–70. An overview of Piatetsky-Shapiro, G. and Frawley, W.J. (eds.) *Knowledge Discovery in Databases.* AAAI Press, Menlo Park, CA., 1991.

Godin, R., Gecsei, J. and Pichet, C. (1989) 'Design of a Browsing Interface for Information Retrieval'. In *Proceedings of ACM SIGIR '89,* 32–39.

Godin, R., Missaoui, R. and April, A. (1993) 'Experimental Comparison of Navigation in a Galois Lattice with Conventional Information Retrieval Methods'. *International Journal of Man-Machine Studies* **38**, 747–767.

Green, T.R.G. (1990) 'The Cognitive Dimension of Viscosity: a sticky problem for HCI'. In *Proceedings of HCI INTERACT '90,* 79–86.

Green, T.R.G. (1991) 'Describing Information Artifacts with Cognitive Dimensions and Structure Maps'. In *Proceedings of the HCI'91 Conference on People and Computers VI,* 297–315.

Grinstein, G., Pickett, R. and Williams, M.G. (1989) 'Exvis: An exploratory visualization environment'. In *Proceedings of Graphics Interface '89,* London, Ontario, 254-261.

Harel, D. (1988) 'On Visual Formalisms'. *Communications of the ACM* **31**(5), May 1988, 514–530.

Hearst, M. A. (1995) 'TileBars: Visualization of Term Distribution Information in Full Text Information Access'. In *Proceedings of ACM CHI '95,* 59–66.

Hendry, D.G. and Green, T.R.G. (1994) 'Creating, Comprehending and Explaining Spreadsheets—A Cognitive Interpretation of What Discretionary Users Think of the Spreadsheet Model'. *International Journal of Human-Computer Studies* **40**(6), 1033–1065.

Hudson, S.E. (1990) 'Adaptive Semantic Snapping: A Technique for Semantic Feedback at the Lexical Level'. In *Proceedings of ACM CHI '90,* 65–70.

Höök, K., Karlgren, J., and Wærn, A. (1995) 'A Glass Box Intelligent Help Interface'. In *Pre-proceedings of IMMI-1: the First International Workshop on Intelligence and Multimodality in Multimedia Interfaces: Research and Applications,* Edinburgh, July.

Inselberg, A. (1981) 'N-dimensional graphics part I: Lines and hyperplanes' IBM LA Science Center Report G320-2711.

Inselberg, A. (1985a) 'The Plane with Parallel Coordinates'. *The Visual Computer* **1**, 69–91.

Inselberg, A. (1985b) 'Intelligent instrumentation and process control' in Weisbin C.R.(ed.), *Artificial Intelligence Applications—The Engineering of Knowledge-Based Systems.* Proceedings of the 2nd conference on AI (CAIA-85). IEEE Computer Soc Press, 302–307.

Inselberg, A. and Dimsdale, B. (1991) 'Parallel Coordinates: a Tool for Visualizing Multivariate Relations'. In Klinger, A. (ed.), *Human-Machine Interactive Systems,* Plenum Press, New York, 199–233.

Inselberg, A. and Dimsdale, B. (1994a) 'Multidimensional Lines I: Representation'. *SIAM Journal on Applied Mathematics* **54**(2), 559–577.

Inselberg, A. and Dimsdale, B. (1994b) 'Multidimensional Lines II: Proximity and Applications'. *SIAM Journal on Applied Mathematics* **54**(2), 578–596.

Inselberg, A., Dimsdale, B., Chatterjee, A., and Chao-Kuei Hung (1993) 'Parallel coordinates: survey of recent results'. In *Proceedings of Human Vision, Visual Processing, and Digital Display IV,* February 1993, San Jose, CA. SPIE Proceedings Series, Volume 1913, 582–599.

Inselberg, A., Reif, M. and Chomut, T. (1987) 'Convexity algorithms in parallel coordinates'. *Journal of the ACM* **34**, 765–801.

Johnson, J. (1992) 'Going Beyond User-Interface Widgets'. In *Proceedings of ACM CHI '92,* 273–279.

Johnson, J.A., Nardi, B.A., Zarmer, C.L., and Miller, J.R. (1993) 'ACE: Building Interactive Graphical Applications'. *Communications of the ACM* **36**(4), 41–55.

Kaplan, S.J., Kapor, M.D., Belove, E.J., Landsman, R.A. and Drake, T.R. (1990) 'Agenda: A Personal Information Manager.' *Communications of the ACM* **33**(7), July 1990, 105–116.

Karlgren, J., Höök, K., Lantz, A., Palme, J., and Pargman, D. (1994) 'The Glass Box User Model for Filtering'. In *Proceedings of 4th International Conference on User Modeling,* Hyannis, ACM Press.

Keen, P. G. W. and Scott-Morton, M. S. (1978) *Decision Support Systems: An Organizational Perspective.* Addison-Wesley, Reading, MA.

Kleinmuntz, D.N. and Schkade, D.A. (1993) 'Information Displays and Decision Processes'. *Psychological Science* 4(4), 221–227.

Kochhar, S. (1994) 'CCAD - A Paradigm for Human-Computer Cooperation in Design'. *IEEE Computer Graphics and Applications* **14**(3), 54–65.

Kochhar, S. and Friedell, M. (1990) 'User Control in Cooperative Computer-Aided Design'. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST '90),* Snowbird, Utah, 143–151.

Kochhar, S., Marks, J. and Friedell, M. (1991) 'Interaction Paradigms for Human-Computer Cooperation in Graphical-Object Modeling'. In *Proceedings of Graphics Interface '91,* 180–191.

Koestler, A. (1964) *The act of creation.* Macmillan, New York.

Lai, K. Y., Malone, T. W., and Yu, K. C. (1988) 'Object Lens: A "Spreadsheet" for Cooperative Work'. *ACM Transactions on Office Information Systems* **6**(4), 332–353.

Lamport, L. (1985) *LaTeX: A Document Preparation System.* Addison-Wesley Publishing Company.

Hing-Yan Lee, Hwee-Leng Ong and Karanbir Singh Sodhi (1995a) 'Visual Data Exploration'. To appear in *Proceedings of the 3rd Intl. Applied Statistics in Industry Conference,* Dallas, Texas, June 1995.

Hing-Yan Lee, Hwee-Leng Ong and Karanbir Singh Sodhi (1995b) 'A KDD Experience in Pedestrian Accident Analysis'. To appear in *Proceedings of the MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases,* Heraklion, Greece, April 1995.

Hing-Yan Lee, Hwee-Leng Ong, Eng-Whatt Toh and Sieu-Kong Chan (1995) 'A Multi-Dimensional Data Visualization Tool for Knowledge Discovery in Databases'. Submitted for publication.

Lunzer, A. E. (1994) 'Reconnaissance Support for Juggling Multiple Processing Options.' Technique Note. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '94).* Marina del Rey, California, November 1994, 27–28.

Mackinlay, J.D., Robertson, G.G., and DeLine, R. (1994) 'Developing Calendar Visualizers for the Information Visualizer'. In *Proceedings of ACM UIST '94,* 109–118.

Magleby, E.J., Burton, R.P. and Scott, D.T. (1991) 'Parallel Axes Graphics Techniques for the Analysis of Multivariate Data'. *Journal of Imaging Science* **35**(6), 394–397.

Mander, R., Salomon, G., and Wong, Y. Y. (1992) 'A 'Pile' Metaphor for Supporting Casual Organization of Information'. In *Proceedings of ACM CHI '92,* 627–634.

Marsh, S. (1992) 'The interactive matrix chart'. *SIGCHI Bulletin (USA)* **24**(4), 32–38.

Maulsby, D.L. & Witten, I.H. (1989) 'Teaching a Mouse How to Draw.' In *Proceedings of Graphics Interface '89,* 130–137.

Meyer, B. (1994) 'Adaptive Performance Support: User Acceptance of a Self-Adapting System'. In *Proceedings of 4th International Conference on User Modeling,* Hyannis, ACM Press.

Modugno, F., Green, T.R.G. and Myers, B.A. (1994) 'Visual Programming in a Visual Domain: A Case Study of Cognitive Dimensions'. In *People and Computers IX, Proceedings of BCS HCI '94,* Glasgow, August 1994, 91–108.

Myers, B. A., Smith, D. C. and Horn, B. (1992) 'Report of the "End-User Programming" Working Group'. In Brad A. Myers (ed.), *Languages for Developing User Interfaces.* Jones and Bartlett Publishers, Boston, 343–366.

Nardi, B. (1993) *A Small Matter of Programming.* MIT Press, Cambridge, MA.

Nardi, B.A. and Johnson, J.A. (1994) 'User Preferences for Task-specific vs. Generic Application Software'. In *Proceedings of ACM CHI '94,* 392–398.

Nardi, B.A. and Zarmer, C.L. (1993) 'Beyond Models and Metaphors: Visual Formalisms in User Interface Design'. *Journal of Visual Languages and Computing* **4**, 5–33.

Ng, W.Y. (1991) 'An Interactive Descriptive Graphical Approach to Data Analysis for Trade-Off Decisions in Multiobjective Programming'. *Information and Decision Technologies* **17**(2), 133–149.

Norman, D. A. (1986) 'Cognitive engineering'. In Norman, D.A. and Draper, S.W. (eds.) *User Centered System Design,* Lawrence Erlbaum Associates, Hillsdale, NJ, 31–61.

Norman, D. A. (1988) *The Psychology of Everyday Things.* Basic Books, New York.

Norman, D. A. (1991) 'Cognitive artifacts.' In Carroll, J. M. (ed.) *Designing Interaction: Psychology at the Human–Computer Interface.* Cambridge University Press, 17–38.

Norman, D. A. (1993) *Things that make us smart: defending human attributes in the age of the machine.* Addison-Wesley Publishing Company, Reading, MA.

Norman, D. A. and Bobrow, D. G. (1979), 'Descriptions: an intermediate stage in memory retrieval'. *Cognitive Psychology* **11**, 107-123.

Paese, P. W. and Sniezek, J. A. (1991) 'Influences on the appropriateness of confidence in judgment: Practice, effort, information, and decision-making'. *Organizational Behavior and Human Performance* **16**, 366–387.

Pavlidis, T. and Van Wyck, C. (1985) 'An automatic beautifier for drawings and illustrations'. In *Proceedings of ACM SIGGRAPH '85. Computer Graphics* **19**(3), 225–234.

Payne, J.W, Bettman, J.R., and Johnson, E.J. (1993) *The adaptive decision maker.* Cambridge University Press.

Pollitt, A.S. (1986) 'Query-by-Menu: A novel DBMS query language, a description and comparison with QBE'. In *Proceedings of the 8th Research Colloquium of the BCS Information Retrieval Specialist Group,* University of Strathclyde, 1986.

Pollitt, S.A., Ellis, G.P., Smith, M.P., and Li, C.S. (1994) 'HIBROWSE: adding the power of relational databases to the traditional IR architecture—the future for Graphic User Interfaces'. In *Proceedings of the 15th Research Colloquium of the BCS Information Retrieval Specialist Group,* Glasgow, 1993, 108–118.

Potter, B., Sinclair, J. and Till, D. (1991) *An Introduction to Formal Specification and Z.* Prentice Hall International (UK) Ltd.

Potter, R. (1993b) 'Guiding Automation with Pixels: a Technique for Programming in the User Interface'. Summary of demonstration video. In *Proceedings of ACM INTERCHI '93,* 530.

Potter, R. (1993b) 'Triggers: Guiding Automation with Pixels to Achieve Data Access.' In Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A. and Turransky, A., *Watch What I Do: Programming by Demonstration,* MIT Press, Cambridge, MA, chap. 17.

Quinlan, R. J. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers.

Rao, R. and Card, S.K. (1994) 'The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information'. In *Proceedings of ACM CHI '94,* 318–322.

Rieman, J., Davies, S. and Roberts, J. (1992) 'A visit to a very small database: Lessons from managing the review of papers submitted for CHI '91.' In *Proceedings of ACM CHI '92,* 471–478.

Robertson, G.G., Card, S.K., and Mackinlay, J.D. (1993) 'Information Visualization Using 3D Interactive Animation'. *Communications of the ACM* **36**(4), 57–71.

Runciman, C. and Thimbleby, H. (1986) 'Equal Opportunity Interactive Systems'. *International Journal of Man-Machine Studies* **25**(4), 439–451.

Russo, J. E. & Dosher, B. A. (1983) 'Strategies for multiattribute binary choice'. *Journal of Experimental Psychology: Learning, Memory, and Cognition* **9**, 676–696.

Sanderson, M. and van Rijsbergen, C.J. (1991) 'NRT: news retrieval tool'. *Electronic Publishing, EP-odd* **4**(4), 205–217.

Sarkar, M. and Brown, M. H. (1992) 'Graphical Fisheye Views of Graphs.' In *Proceedings of ACM CHI '92,* 83–91.

Shipman, F. and McCall, R. (1994) 'Supporting Knowledge-Base Evolution with Incremental Formalization'. In *Proceedings of ACM CHI '94,* 285–291.

Shneiderman, B. (1993) 'Dynamic Queries for Visual Information Seeking' University of Maryland technical report CS-TR-3022, Sept 1993 (revised Jan 1994).

Smith, D. C. and Susser, J. (1992) 'A Component Architecture for Personal Computer Software'. In Brad A. Myers (ed.), *Languages for Developing User Interfaces.* Jones and Bartlett Publishers, Boston, 31–56.

Smith, R. B., Ungar, D. and Chang, B.-W. (1992) 'The Use-Mention Perspective on Programming for the Interface'. In Brad A. Myers (ed.), *Languages for Developing User Interfaces.* Jones and Bartlett Publishers, Boston, 79–89.

Sørgaard, P. (1988) *A Discussion of Computer Supported Cooperative Work.* Ph.D. thesis, Aarhus University, Denmark.

Srinivas, M. and Patnaik, L.M. (1994) 'Genetic Algorithms: A Survey'. *IEEE Computer* **27**(6), 17–26.

Suchman, L. A., (1987) *Plans and situated actions: the problem of human-machine communication.* Cambridge University Press, Cambridge.

Thimbleby, H. (1990) *User Interface Design.* ACM Press, New York, NY.

Todd, S. and Latham, W. (1992) *Evolutionary Art and Computers.* Academic Press Ltd.

Tufte, E. R. (1990) *Envisioning Information.* Graphic Press, Cheshire, CT.

Tukey, J.W. (1977) *Exploratory Data Analysis.* Addison-Wesley, Reading, MA.

Tversky, A. and Kahneman, D. (1981) 'The framing of decisions and the psychology of choice'. Science **211**, 453–458.

Tversky, A. and Kahneman, D. (1990) 'Cumulative prospect theory: An analysis of decision under uncertainty'. Unpublished working paper, Stanford University.

Tweedie, L., Spence, R., Dawkes, H. and Su, H. (1995) 'The Influence Explorer'. Short paper. Presented at ACM CHI '95, Denver, Colorado.

Tweedie, L., Spence, R., Williams, D. and Bhogal, R. (1994) 'The Attribute Explorer'. In Video Proceedings of ACM CHI '94. Also described in Conference Companion, 435–436.

Vetschera, R. (1994) 'MCView: An Integrated Graphical System to Support Multiattribute Decisions.' *Decision Support Systems* **11**(4), 363–371.

Waite, K. W. (1991) 'Iconographer: an Icon-based Visualisation Toolkit'. In Waite, K. W. (ed.) *Current Human-Computer Interaction Research: An Anthology of Recent Papers—Volume II.* Report GIST-91-1, Computing Science Department, University of Glasgow, 1–14.

Wegman, E.J. (1990) 'Hyperdimensional Data Analysis Using Parallel Coordinates'. *Journal of the American Statistical Association* **85**(411), Sept 1990, 664–675.

Weiland, W.J. and Shneiderman, B. (1993) 'A Graphical Query Interface Based on Aggregation Generalization Hierarchies'. *Information Systems* **18**(4), 215–232.

Weisberg, R. W. (1986) *Creativity: genius and other myths.* W. H. Freeman and Company, New York.

Weitzman, T. (1986) 'Designer: A knowledge-based graphic design assistant' ICS Report 8609, University of California, San Diego.

Williams, M. D. (1984) 'What makes RABBIT run?'. *International Journal of Man-Machine Studies* **21**, 333–352.

Williamson, C. and Shneiderman, B. (1992) 'The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system'. In *Proceedings of ACM SIGIR '92,* 338–346.

Winograd, T. and Flores, F. (1986) *Understanding Computers and Cognition—A New Foundation for Design.* Addison-Wesley.

Woodbury, R.F. (1991) 'Searching for Designs: Paradigm and Practice'. *Building and Environment* **26**(1), 61–73.

Yen, J., Neches, R., Debellis, M., Szekely, P. and Aberg, P. (1991) 'Backbord: An Implementation of Specification by Reformulation'. In Sullivan, J.W. and Tyler, S.W. (eds.), *Intelligent User Interfaces.* ACM Press, New York, NY, 421–444.