An Informal Information-Seeking Environment

David G. Hendry and David J. Harper
School of Computer and Mathematical Sciences, The Robert Gordon University, Aberdeen AB1 1HG, United Kingdom. E-mail: {d.hendry,d.harper}@scms.rgu.ac.uk

When an opportunistic searcher encounters an over-determined information system, less than ideal search strategies often ensue. The mismatch can be addressed by reducing the determinacy of the system, thereby making it more amenable to informal problem-solving practices. This claim is investigated by designing an information-seeking environment, where search techniques are represented with a data-flow notation and where the searcher has control of layout; thus, to seek information, is to manage space. Search plans can be represented on the display, and perceptual cues about search progress are captured while searching. With elements of search activity visible, opportunistic problem-solving should be well supported. The interface is designed to be extensible so a wide range of search techniques can be represented, and emphasizes search material, such as queries, results, and notes, rather than system controls.

Introduction

A major characteristic of any problem solving situation is the “conversation” that the user has with his or her materials, whether computerized or not. Schön (1983) characterizes how pencil and paper support design this way:

The act of drawing can be rapid and spontaneous, but the residual traces are stable. . . . No move is irreversible. The designer can try, look, and by shifting to another sheet of paper, try again. As a consequence, he can perform learning sequences in which he corrects his errors and takes account of previously unanticipated results of his moves. (pp. 157–158)

Although not a designer in the conventional sense, the searcher is a problem solver, who sets goals, monitors progress, explains solutions, and optimizes for solution quality in the time available. But, like a designer, a searcher often produces an artifact, that is, a body of material bearing on the task at hand. If information about the provenance of that material—how it is “unfolding” or “evolving,” “contracting” and “expanding”—is available, then searching is potentially richer because the chances improve that the searcher will be prompted to reformulate and abandon plans, backtrack to points of task deferral or significant decision, and compare things side-by-side. When aspects of the search process are externalized, both prospective memory, plans for the future, retrospective memory, tasks completed, are supported. In programming environments, support for these basic cognitive tasks is considered essential because they allow people to work the way they want to—opportunistically (Visser, 1994). There is no a priori reason to think they are any less important in environments for information seeking; indeed, planning, backtracking, and comparison are at the heart of search tactics and strategies (Bates, 1979a, 1979b).

When assessing the match between users and information systems, reviewers often point to learning difficulties and working memory failures, technical incompatibilities, and large and complex feature sets. In a penetrating analysis, Bates (1990) argues that the goal should not be to replace the searcher with a knowledge-based system, but rather to design the interface to support the strategic, opportunistic behavior of searchers, and argues that there is a mismatch between the needs of searchers and what systems currently provide: “Good [search] strategy is usually achieved in spite of, or is superimposed on, information system designs” (p. 567). In another review, Shaw (1991) begins: “The less efficient searchers are more likely to become confused by the different possibilities, forget information retrieved, and repeat searches” and asks “how interface design can decrease stress or provide support?” (p. 166). Marchionini (1992) reviews the state of the art by setting systems for information seeking against a set of core tasks and assessing the fit. In his estimation, there is particular need for tools that help people define and discuss their problems. In a system-oriented review of over 50 different search-interfaces, Vickery and Vickery (1993) conclude that current interfaces may be “over-elaborated” or “over-engineered,” that is, the structure of the information displays...
and the ordering constraints for entering information are predetermined and unchangeable. In short, they suggest that people might benefit from simple systems that create a “flexible and revealing dialogue between the system and user” (p. 162). In motivating their own very innovative, long-standing research on information workspaces and visualization, Rao et al. (1995) say: “Though users may want to interleave access operations and track their progress, current systems are weak in their support for this process” (p. 30).

The contrast between these reviews of the match between searcher and system and the sketching described by Schön is stark. It suggests that the searcher might benefit from an “informal” workspace, which emphasizes ease of expression and gives prominence to the material of information seeking rather than system-oriented, retrieval controls. We shall see that informality, does not imply “less usefulness”; in fact, it is just the opposite: Informality opens the interface and gives searchers access to a wider range of information-seeking tools and strategies, and it allows searchers to be as structured and complete in their work as desired.

The goal of this research is to investigate how an informal computerized environment, or workspace, can support information seeking. In the next section, we outline the design space for information-seeking environments and justify our interest in one, underexplored, region of it. Next, we present a prototype, called SketchTrieve, that emphasizes this region. Then, we discuss how this environment might support information seeking in ways not available in other more over-determined systems. The prototype and the cognitive rational informing its design suggest how informal environments might better match the opportunistic quality of information seeking.

Information-Seeking Environments

Major Design Concerns

An information-seeking environment cannot be too specialized because that would impede access to a variety of different repositories. Nor can it be insular because that would impede the use of other applications during and after search (e.g., graphing, tabulating, and annotating tools; O’Day & Jeffries, 1993). Nor can it offer support for complex searching at the cost of making routine searching hard.

Marchionini (1992) calls for a “spreadsheet for information organization and retrieval” (p. 158); indeed, the spreadsheet, though targetted for a different problem-domain, confronts similar problems. Children use them to work math problems, scientists use them to solve differential equations. They are as effective for provisional scratch-pad computations as for definitive business presentations. To support different tasks, spreadsheets offer many niches of specialized function.

If the spreadsheet is a success model for end-user computing, then what does it suggest about environments for information seeking? Based on reviews of the spreadsheet (Hendry & Green, 1994; Nardi, 1993; Nardi & Miller, 1991), we derive four generalizations that seem to account for its success, and, by analogy, appear applicable to information-seeking environments. We claim that an information-seeking environment should:

1. Mediate communication. An information-seeking environment should allow the searcher to discuss searching with his or her peers, mentors, or clients. It should allow people to organize and explain a search episode, to inspect and learn from each other’s searches, and to ask for help. In ethnographic studies of spreadsheet use in situ, Nardi and Miller (1991) showed it to be an outstanding tool for computer-supported cooperative work. The reason? Computations can be laid out logically and documented with words and graphics; thus, formulae can be understood, copied, and reused.

2. Simplify IR technologies. Since the 1970s, it has been recognized that people encounter serious problems when dealing with the diversity and incompatibility of different systems; an environment for information searching should hide the “perplexing plethora of access protocols, system designations on different telecommunication networks, passwords, command languages . . .” (Williams, 1986, p. 205). The spreadsheet simplifies technology, in its case writing algorithms, with a scheme for referencing cells on a tabular display, which simultaneously acts as an input and output surface (Nardi & Miller, 1991).

3. Represent many IR technologies. An information-seeking environment should allow the searcher to interact with a variety of different “matching” technologies, visualizations, and interaction styles. It should be possible to carry out a Boolean search using a 1960s command language, delegate a request to an Internet search-agent and track its periodic reports, and apply the latest visualization of an information space. The spreadsheet provides a similar degree of variegation: New niches of function are incorporated, but such that the overall system image is not compromised.

4. Support incremental learning. An information-seeking environment should allow searchers to do “simple” searches quickly, without knowing very much. At the same time, the range of techniques for search and information extraction should be large; with experience, should come power. With spreadsheets, an engineer might first learn formula copying and use the scientific functions, then learn about the different styles for referencing cells, but never graph data or study macro programming. A financial analyst, on the other hand, might use the graphing tools almost exclusively.

In sum, it is these features, when combined, that make the spreadsheet ideal for the fluid expression of ideas and exploration of problems (Hendry & Green, 1994). Perhaps, then, a general lesson is to not take too narrow
a view of the IR interface, where, for example, particular query and result displays become preeminent. By broadening the view to that of an environment for human problem-domain communication (Fischer & Lemke, 1988), which supports the organization and exploration of search process, problems of over-determinacy can become more tractable. Two systems that take a particularly broad view, but for reasons unrelated to the spreadsheet, are InfoGrid and Labrador (Rao, Card, Jellinek, Mackinlay, & Robertson, 1992; Rao, Russell, & Mackinlay, 1993).

Design-Space Analysis

We use the Cognitive Dimensions Framework (Green, 1989, 1991) to map out the design space for information-seeking environments. These dimensions have been used to assess the strengths and weaknesses of a variety of cognitive artifacts, including visual programming environments (Green & Petre, 1996) and the spreadsheet (Hendry & Green, 1994). Green (1991) developed the cognitive dimensions to address what he considered to be three flaws of Human–Computer Interaction: Word poverty, focus on surface features, and over-technical analysis. Without a common vocabulary for characterizing the "structural" properties of user-interfaces, discussion is made difficult. For example, recognizing near identical problems in different contexts or arguing for particular design trade-offs requires detailed exposition in each case, rather than reference to an agreed concept. The dimensions are intended to improve discussion on issues of usability and usefulness, both in retrospect for existing artefacts, and proactively during design work. They can be used with more precision than design principles, such as: Affordances, constraints, feedback, natural mappings, and visibility (Norman, 1988). However, unlike other methods of analysis in HCI, such as scenario representations (Carroll, 1994), they tend not to focus on extreme detail. Thus, the framework complements other methods. Below, six of the dimensions are used to represent some of the major design variables for an information-seeking environment.

Closeness of Mapping. How closely do elements on the display relate to the searcher's problem domain? Some systems tend to emphasize the content of the repository, others emphasize the search engine. For example, RABBIT (Williams, 1984) operates on the basis of documents, where searchers flesh out an example document by selecting domain concepts from menus. RABBIT uses the example to generate a list of similar documents, which searchers can then "criticize" to obtain a new set. Other interfaces are oriented towards the system; for example, Envision's query window emphasizes controls for operating a best-match engine (Fox, et al., 1993).

Diffuseness/Terseness. Tufte (1990) complains that "Vacant, low-density displays . . . require viewers to rely on visual memory—a weak skill—to make a contrast, a comparison, a choice" (p. 50). How many bits on the display are directly related to a searcher's task at hand? If a query dialog, including its controls, must always lie beside a result display, then space that might otherwise be used to aid information extraction is unavailable. Displays claimed to be effective for disambiguating a large number of results often tend to have high information density. One example is the InfoCrystal (Spoerri, 1994).

Role Expressiveness. This dimension refers to the extent that relations between two things are visible. Term-document role expressiveness captures the notion that a retrieval system should inform the searcher as to why a document has been retrieved for a particular request. One technique is to highlight the words in the document that have contributed to the matching. Though simple, this perceptual cueing is highly effective (e.g., Landauer, et al., 1993). A more expressive means for highlighting the roles of terms in documents is TitleBars, which compactly shows the relative lengths of documents, term frequencies, and term distributions within a document (Hearst, 1995). With this visualization technique, people can determine if a document is, for example, equally about term X and Y, or if only one part of the document is about X. Structured search feedback is another method for disambiguating the roles of query terms (Landauer et al., 1993). Here, the idea is that the occurrences of terms are posted against a structural representation of the document or information space; for example, the table of contents. Term-document role expressiveness contrasts with the term-document type. How do two or more documents or neighborhoods in a large information space relate to each other? Showing the relations among documents is the goal of much of the visualization research in IR (e.g., Rao et al., 1995).

Secondary Notation. In many applications, people have control over how their work is arranged on the display. In a spreadsheet, for example, layout can be used such that an algorithm progresses from top-left to bottom-right, with each major step shown as a block. These visual cues do not affect the meaning of the computations; however, they can make a huge difference in comprehensibility (see Hendry & Green, 1993). Petre and Green (1992) introduced the term secondary notation to refer to typographic and layout cues that lie outside of a formal notation, such as found in circuit diagrams and programming languages. Systems for information seeking rarely, if ever, provide a secondary notation.

Side-by-side-ability. Hock (1983), a trainer in information seeking, insists: "A searcher must recognize that . . . comparing different approaches to a question can change a virtually worthless search into a good search" (p. 85). Comparison tasks are supported when it is possible to position queries, documents, or result sets in the
same locality. For example, if query X and query Y seem to produce different results, then it is much easier to make inferences about why, if X and Y can be viewed side-by-side. If they cannot be viewed together, working memory is unduly taxed or the searcher has to decouple from the system and rely on pencil and paper. Meyrowitz (1991) claims that many hypertext systems fail to offer good side-by-side-ability.

Viscosity. This dimension refers to how easily one object or action can be changed into another. If you have carried out three searches against one database, can you change the database and not have to reissue the same three queries? If you have to reissue them, then the viscosity of the information-seeking environment is higher than it might be. Other things being equal, environments with low viscosity are best because they allow people to work by trial and error.

Summary. On the one hand, research in information seeking generates very rich descriptions of user interaction, such as is summarized by Ingwersen (1992) and Marchionini (1995). On the other hand, the literature contains many expositions of particular interfaces, for example, the detailed functional features surveyed by Vickery and Vickery (1993). A serious problem, then, is creating a bridge between these bodies of knowledge so that consequences of particular design decisions, and the reasons for particular searcher behavior, can be discussed generally.

The cognitive dimensions appear to be a useful bridge. With them, it is possible to ask questions, such as: 1) Is there a tension between closeness of mapping and effective role expressiveness; 2) what is the repertoire of methods that can lower viscosity in interfaces for information seeking; 3) system X has very high information density, but why does it not offer much in the way of side-by-side-ability; and 4) what would an environment for information seeking be like if it provided a secondary notation. With a profile of how an interface measures up against the dimensions, one can then ask questions about users and tasks for which the interface is most suited. Although these and other cognitive dimensions, such as abstraction gradient, error-proneness, and progressive evaluation, have been used mostly in the evaluations of programming languages, the dimensions seem to capture important features about interfaces for information seeking.

Secondary Notation

We do not know of an environment for information seeking that emphasizes the dimension of secondary notation, where the searcher can employ space, layout, and locality to represent and to organize his or her activities. Such an environment might have similar advantages to the spreadsheet. The claim is that if searchers can organize their information seeking using spatial arrangement, the interface becomes flexible, rather than “over-elaborated.” Secondary notation can help the searcher in two important tasks: 1) Explanation and discussion of search process, that is, comprehension; and 2) search planning, monitoring, and task selection, that is, guiding the execution of search activity.

For improved comprehension, the searcher, for example, could put the most interesting results in the top-left of a workspace, show a series of related results to the right, and for completeness, show all the failed searches at the bottom. Layout policies, such as these, are common in spreadsheets. In electronic schematics, the locality of components and how wires do and do not cross provide the experienced reader with valuable clues about the function of the components (Petre, 1995). We shall see evidence that even novices, given the opportunity, seem to invent policies for laying out their search work.

The work of Petre (1995) and Petre and Green (1992) concentrates on how secondary notation can help in comprehension tasks. However, Kirsh (1995), in an article entitled The Intelligent Use of Space, presents a classification for how space can guide problem solving from moment to moment. One important idea is that people often “jig their environment” by putting cues into it; thereafter, the environment prompts them to take certain actions. For example, a searcher might type four queries onto the workspace and arrange them vertically, as a plan for what to do. Then, after running a query, the searcher could put it to one side. Here space is being used to record which searches have and have not been completed; because of the layout policy, progress on the task is resilient to interruptions, such as telephone calls. Such a display, using Schön’s terms, “talks-back” to people about their work. The prototype discussed below is intended to support searchers in just this way.

We have seen glimpses of search plans in “pseudo-searches,” notes made on paper for planning an online search, and Spink & Goodrum (in press) report that sketching is quite common in search intermediaries working notes. They argue that a workspace should allow searchers to create and manage working notes. Bellamy (1994) shows how “pseudo-code” helps programmers to communicate with colleagues and explore problems. Perhaps, “pseudo-searches” play a similar role. An informal environment for information seeking would help with such tasks.

The SketchTrieve Design

To investigate the effects of secondary notation in search activity we are implementing a prototype called SketchTrieve; the name is meant to connote ease-of-expression. Shown, in Figure 1, is a SketchTrieve canvas on the right and a two-pane document viewer on the left. The viewer offers minimal function. It is suitable for reading short documents, such as abstracts, and because
there are two panes, for comparing documents side-by-side.

System Image

SketchTrieve is a graphic editor, where search activity ensues by the manipulation of information retrieval services on a canvas. The dominant conceptual model is to select the services you need, connect them, press Run, and results will be displayed.

A service is represented by an icon, called the rubric, which indicates its purpose. There are two fundamental kinds of services. Input-services, accept queries from searchers, for example, free-text queries. Retrieval-services, in contrast, produce results, for example, a list of titles for matched documents from a repository. Searchers can distinguish these two types by glancing at a service’s rubric: An arrow pointing down indicates an input-service, whereas a question mark indicates a retrieval-service. Eight services are shown in Figure 1.

Like graphical elements in a drawing editor, searchers can select services, and move and resize them. Using buttons in SketchTrieve’s control panel, searchers can copy and remove services. To create a new instance of a service, the searcher selects from a cascaded pop-up menu. This menu, labeled “Z39 Servers” in Figure 1, consists of a list of service categories and sub-menus that contain particular kinds of services. (The category of the last created service is always shown in this pop-up.) Searchers can customize this menu, and canvases can be Saved, Opened, and so on.

Docking of Services

In Figure 1, the query “intelligent design tools,” has been docked with a single retrieval-service, indicated by the fuzzy bar between the two services. To dock an input-service to a retrieval-service, the searcher drags one to the other so that their bottom- and top-left corners are approximately aligned; then, SketchTrieve automatically snaps them together. To undock, the searcher simply moves one of the services away. When the Run button is pressed, for services that have been docked, or, as explained below, “wired,” the query flows from the input-service to the retrieval-service, and some results are produced. In Figure 1, the titles of matched documents are shown, and one document is displayed in the upper pane of the viewer. In this case, the retrieval-service represents an experimental, best-match Z39 Server we are implementing in other research, and the selected repository is a popular bibliography on HCI.

To the right, aligned vertically, are additional queries that have yet to be processed. To submit one of them, the searcher may: 1) Undock the query “intelligent design...
tools' and move it away; 2) dock the new query to the retrieval-service; and 3) press the Run button. Alternatively, the canvas could be scrolled to the right, and a new retrieval-service could be created; then, the query could be docked to it. Of course, nothing prevents the searcher from typing a new query into a docked input-service and pressing the Run button.

Finally, to the bottom-right is a diagonal line of collapsed retrieval-services that are being saved. All services can be collapsed and expanded by double clicking their rubrics. This is one method for controlling space consumption.

Wiring of Services

Wiring, another way to connect an input-service to a retrieval-service, allows a query to flow into several retrieval-services for improved side-by-side-ability. In Figure 2, for example, the same query flows into three different retrieval-services. The left-most retrieval-service is a general database of books and those to the right are two different result views of the HCI bibliography.

To wire two services together, the searcher presses a service's rubric or border, while holding down the Control key, and then moves the mouse to the targeted service and releases. Wires can be moved around the perimeters of services, and when services are moved, the wires are repositioned. To unwire, the ‘wiring operation’ is repeated in the same direction as the original wiring.

Why provide both docking and wiring? On the one hand, docking has a simpler action language. With docking, either the input- or retrieval-service can be moved to the other, and thus there are no ordering constraints; whereas with wiring, the searcher must begin with the input-service and use the Control key. On the other hand, docking is less expressive. With docking, only two services can be connected, flush vertically; whereas with wiring, many retrieval-services can be connected to a single input-service and the services can be positioned anywhere. We believe that the expressive potential is sufficiently different that searchers might use the two techniques to represent different things.

Drag-n-Drop

The third major interactive feature of SketchTrieve is drag-n-drop. In an early design study, we attempted an extreme stance where only drag-n-drop was provided.
FIG. 3. Drag-n-drop. Results produced by a retrieval-service can be dragged onto the display and used for other purposes.

(Hendry & Harper, 1995). However, comparative design mock-ups later convinced us that drag-n-drop alone perturbed the stability of the display to such an extent that the secondary notation would be unduly harmed. Nevertheless, we have not completely abandoned drag-n-drop. Searchers can drag a result from a results view onto the canvas; thereafter, it becomes an input-service, called a Document-alias.

In Figure 3, document titles have been dragged out of the retrieval-service on the left and positioned on the display. One such title is “Architectural drawing.” Notice that the document “Civil engineering drawing” is being used as a query to the HCI bibliography. Here, all words in this reference are being used in a best-match search.

The services in the top left are called Folders. Document-aliases can be dragged into and out of them. Folders are neither input- nor retrieval-services; rather, they are called filter-services. In general, a filter service accepts input from some service, and produces output that can be used by other services; for folders, drag-n-drop is a convenient mechanism, but filter-services can also be docked and wired. By convention, an double-arrow pointing up and down indicates a filter-service. In this example, the folder with a circular rubric is being used to hold more general references about drawing, whereas the container with the square rubric is being used to hold references specifically about drawing in design and engineering. Folders can be used for other things, for example, as a list of relevant documents in a system that accepts relevance feedback.

Customizing Services

An important feature of SketchTrieve is that searchers can customize the services, initially installed by an application programmer. Searchers can name services, provide descriptions for them, and choose a set of decorations: The surrounding shape of the rubric, its color, and thickness of line. Thus, services can be “typed.” This is a simple application of what Thomas Green calls the “description level,” where elements of a searcher’s problem domain are coded perceptually on the display (see Hendry & Green, 1993). The rubrics allow searchers to improve the closeness of mapping.

Services often have parameter settings, for example, communication settings, the databases to be searched, the number of documents to be retrieved, the type of view in which the matched documents are to be visualized, and so on. In SketchTrieve, a dialog of such service settings is accessible by pressing the Info button; this feature allows users to control the information density.

These settings can be set for all services of a given type. For example, a Z39.50 service, installed by the application programmer, might be customized by a user knowledgeable in its communication settings, and then customized in simpler ways by more discretionary searchers. The goal is to open SketchTrieve to varying degrees of customization, and thereby support incremental learning.

Indicating Connectable Services

Some services can be connected together, and some cannot. It makes no sense to connect, for example, two
retrieval-services; no data will flow between them. Nor is it sensible to connect a Free-text service to a Boolean-search service because the engine expects a Boolean expression, not a chunk of text.

The rules for how data flows between services must be made visible. When the mouse is clicked and held-down over a service, the display reacts in the following way. If the clicked service is an input-service then its bottom edge is made “fuzzy” and all retrieval-services that will take its input show “fuzzy” tops. Analogous perceptual cues are shown when a retrieval- or filter-service is pressed. SketchTrieve will not allow searchers to connect incompatible services.

Service States

Services switch among one of four states: “Working,” when results are being computed; “ready,” when new results have been computed but have not yet been inspected; “inconsistent,” when, for example, a user changes an input but has yet to press the Run button; and “okay,” when no state information is applicable. To signal these states, a stop-lights metaphor is used: Red for busy, working; green for ready, new-results-to-inspect; yellow, for caution, output-is-inconsistent-with-input; and when the stop light is absent, no state information is applicable. An oval containing one of these colors appears beneath the rubric to inform users about its state. (The retrieval-service at the left of Fig. 2 shows a stop-light.) High luminosity is used for these colors so that they stand out against the pastel colors used for the decorations and the light gray used for the wires and the borders of the services.

Summary

SketchTrieve is a data-flow notation for information seeking. Only compatible services can be connected by docking and wiring. Beyond this, SketchTrieve makes no assumptions about what people should do. In this sense, it is informal and under-elaborated. The prototype illustrates how docking, wiring, and drag-n-drop can be used to interact with search technology. It gives prominence to search materials by de-emphasizing retrieval controls. It allows services to be customized such that they can be reused by searchers with different levels of experience. For a technical description of SketchTrieve, see Hendry and Harper (1996).

Discussion

Cognitive Dimensions Review

To generalize, the design claims of SketchTrieve can be described by returning to the cognitive dimensions:

Closeness of Mapping. The rubrics allow searchers to encode aspects of their problem domains on the display, and system controls are normally hidden; thus, prominence is given to search materials rather than to system features.

Diffuseness/Terseness. SketchTrieve has high-information density: The rubrics and service-content windows have been designed to minimize the use of display space. Further improvements would be control over the visibility of the service’s controls and scroll bars, as well as control over the position of the rubric relative to the service’s view.

Role Expressiveness. SketchTrieve, as it stands now, provides no term–document or document–document role expressiveness. This not a consequence of the design per se, but of the research focus.

Secondary Notation. SketchTrieve, of course, emphasizes this dimension (see below). Additional provision for secondary notation, such as graphics and being able to type directly onto the display could be added, as in spreadsheets.

Side-by-side-ability. Because services can be expanded and collapsed, resized and moved, the searcher is able to make side-by-side comparisons. One direction for improvement is to provide forms of role expressiveness between two result views; for example, clicking on one result highlights related objects in the other.

Viscosity. Services in SketchTrieve are typed, indicated by their decorations. This is a very simple form of abstraction that can be used to lower the viscosity of the environment when a find/change dialog is provided. For example, suppose a searcher periodically runs a half-dozen queries against a database, but learns of a new database to search. To run the queries against the new database, it is necessary to change each of them. With a find/change dialog, however, the searcher could say “change all the source databases being accessed in the blue circles to X.”

Secondary Notation

Figures 1–3 exhibit a range of different uses of space. We claim that the secondary notation that emerges from arranging services on the canvas “talks-back” to the searcher in interesting ways about his or her problem solving. In Figure 1, a searcher’s idealized goal might be: “Submit a series of queries against a single repository, and save selected results sets.” Inspecting the canvas, one can discern two types of plans externalized by the searcher: Plans for what queries are to be issued, and plans for what results will be scrutinized later. Direct control allows searchers to use the display for both pro-
spective and retrospective memory about how the search is progressing.

In Figure 2, a searcher’s idealized goal might be: ‘‘Compare the general types of material in two different databases.’’ Hence, different views of databases have been positioned side-by-side so that comparisons can be made. Control over layout allows searchers to give emphasis to aspects of their work that are important to them.

In Figure 3, a searcher’s idealized goal might be: ‘‘Collect a set of documents on ‘drawing in engineering’.’’ Relevant documents are being saved in two folders; perhaps the folder containing general references was created in response to new needs after the search was begun. Documents still to be followed-up are positioned on the display as backtrack points, and notice the document ‘‘Let’s discover crayon’’ in the bottom-right: Because of its unique locality, it is given special status.

In all three figures, aspects of the searcher’s problem domain have been encoded in the rubrics’ decorations; for example, the HCI bibliography is represented by a light colored octagon, whereas the general reference database is represented by a dark octagon.

To seek information with SketchTrieve is to manage space. That is, searchers have direct control over what and how a record of their interaction is kept. No matter what the searcher does, some elements of his or her work—a group of input-services here, a series of results there—are captured on the display; thus, aspects of work become externalized on the display. Searchers can type in queries into input-services first, then set-up the retrieval-services, or they can do it the other way around. They can unwire services and move them away or delete an input-service; thus, searchers may lose information about what input generated what output. But, at the same time, they can also copy services and position them in ‘‘safe-areas.’’ Although we have much experience with space in everyday living, Petre (1995) shows that its proficient use in notations comes with practice; thus, people who have experience with SketchTrieve can be expected to use space more intelligently than novices. The experts will devise layout policies that are effective for them.

**SketchTrieve in Use**

We have performed simple formative evaluations of SketchTrieve, not to test predefined usability criteria, but rather to explore how people use its secondary notation. There were three main goals: 1) To estimate the efficiency of the action language; 2) to seek evidence that the display supports working memory during search; and 3) to seek evidence that the display can be used to plan for and organize longer term goals. We cannot report definitive results on these issues; however, observations suggest that, with a little practice, the action language is used fluidly, and the display does seem to support search activity in interesting ways.

Five people ranging in age from 10 to 58 were recruited. All had familiarity with the mouse and keyboard. After some basic instruction, subjects completed a series of five different sets of tasks. The procedure was informal; often, after a task, and sometimes during one, we would prompt people for their reactions and discuss the reasons underlying what they did. All tasks required subjects to search for known targets: Countries with names between five and seven letters. Evaluations took about 60 min.

To estimate basic efficiency of the action language, one set of tasks required subjects to carry out requests of this form: ‘‘Search for [Jamaica] in the [local/remote] database.’’ With practice, subjects can complete this task in about 22 s (range 8–36 s, five people timed on the fourth task), that is, they can create an input- and retrieval-service, wire or dock them, and type the country name. Of course, if the services are ready on the display, the task requires only the time to point and type (about 10 s, range 7–14 s, five people timed on the fourth task). For rough comparison, we have found that adults can find five and six letter words in the Webster’s Ninth New Collegiate Dictionary in about 16 s (range 5–28 s; five people timed on four words). This time, however, includes the time to scan the page and point to the word. As the experience with SuperBook has shown, the conventional book is a strong competitor (Landauer et al., 1993).

To see how the display might support working memory, subjects searched for multiple countries in multiple databases. We conjectured that something of the task structure would be discernible from the display. One task took this form: ‘‘1) Find [Algeria] in the [remote] database and [Vanuata] in the [local] . . . . 2) Now, switch the queries so they search different databases.’’ Inspecting Figure 4, one person carried out the Algeria-remote and Vanuata-local searches on the left and right, respectively. The two pairs of input- and retrieval-services were positioned close together, and aligned vertically. Then, he pressed Run. Next, he unwired the two retrieval-services and moved them down, making room for the new local and remote databases. He then wired these, but, as shown in Figure 4, he did not move the original input-services; rather, he crossed the wires. Later, he explained that he estimated this tactic would save time; however, an interesting consequence of the crossover is that it signals that some change in plans or usage has taken place. In sum, he used the vertical dimension to order his searches, stacked with the most recent on top, and the horizontal dimension was used to organize the two databases being searched.

To see how the display might help searchers organize longer term search goals, the experimenter instructed subjects thus: ‘‘I’m going to give you about 10 searches to carry out for several different people. I would like you to complete each search and keep track of your work.’’ These requests followed this template: ‘‘[Dan] is looking for something on [Panama] in the [local/remote] database.’’ Shipman, Marshall, and Moran (1995) analyzed
the spatial arrangements used on several two-dimensional surfaces, both computational and not, and report four common primitive structures: Lists, items aligned vertically or horizontally; stacks, items of "a single type that overlap significantly"; composites, items of "regular spatial arrangements of different types"; and heaps, "overlapping collections of different types." Would similar structures emerge in SketchTrieve?

Figure 5 is representative of how people have tackled this task: The horizontal dimension tracks the clients and the vertical dimension tracks the searches. As the subject encountered a new client in a search-task, he would create a Notes service and type in the client’s name. (A Notes service is a type of input-service, intended for annotating and planning search activity.) To carry out the searches, he created Free-text services, and positioned them just beneath the notes. He reused these Free-text services by unwiring the old retrieval-service, adding a new one, and wiring the new one. The long wire to the left of the figure is telling. Another interesting use of space can be seen in the middle where the note for Jill is wide. The reason seems to be that two of Jill’s searches occurred in succession just after a search for Dan. In sum, using the terms of Shipman, Marshall, and Moran (1995), the display can be characterized as a horizontal list of composites. Heaps or stacks have not been observed in the evaluations, but these structures are certainly expressible in SketchTrieve.

In contrast to the idealized scenarios in Figures 1–3, the displays produced by users of SketchTrieve are messy. Nevertheless, apparently useful information about the structure of the tasks completed is captured, while, at the same time, search proceeds with efficiency. This evidence is a first step towards testing stronger claims about the benefits, and costs, of secondary notation in information seeking. Speculating, it appears that secondary notation can support the moment-to-moment execution of search tasks, possibly, in particular ways, through the creation of "holes" on the display, crossed wires, and other irregular visual features. Larger chunks of activity can made comprehensible through structural arrangements of the sort reported by Shipman et al. (1995). Reports on the use of spreadsheets (e.g., Hendry & Green, 1994; Nardi & Miller, 1991), suggest that for more contextualized search
tasks, the use of secondary notation will be more pronounced.

Conclusion

The literature on information-seeking suggests that a general performance barrier is that IR interfaces are over-determined, that is, they prescribe the structure of the information displays and the ordering of dialogs too stringently. Marchionini (1995) summarizes the problem this way: “The information-seeking process has become much more integrated, fluid, and parallel because of electronic environments, but most of the tools and techniques that have been developed focus on query formulation and examination subprocesses” (p. 196).

Drawing lessons from the spreadsheet, we have addressed the problem broadly, aiming to design an information-seeking environment that: 1) Mediates communication; 2) represents a variety of IR technologies; 3) simplifies access to technologies; and 4) supports incremental learning. SketchTrieve is a response to these concerns, and we have tried to explicate how it addresses the problem of over-determinacy and rigorously justify its design.

The next step is to observe people searching for things they would really like to have and extracting information they need in their work. To this end, we intend to elaborate SketchTrieve such that Internet search engines are representable and such that it works alongside an Internet browser. Thus, returning to Figure 1, the left is location-centric for browsing strategies, whereas the right is retrieval-centric for more analytical strategies (Marchionini, 1995; Rao et al., 1992). Attempting to harmonize these world views will provide a transition from the largely analytic-based approach to design followed to date, to a more empirically driven one. Ultimately, assessing SketchTrieve against the above four concerns requires ongoing, rigorous usability study for iterative redesign (Landauer, 1995).

Another research direction is to formalize exactly how SketchTrieve’s display might influence information-seeking behavior. We conjecture that emphasizing secondary notation reduces the determinacy of an IR system, and consequently strategic, opportunistic searching is better supported. If this is true, then certain information on the display must be responsible. The scenarios in Figures 1–3 seem to particularize the effects of associating monitoring
information with content-based search information, for example, the sources identified by Michel (1994). Display-based cognitive modeling (e.g., Larkin, 1989) may clarify the problem and eventually lead to a better understanding of what prompts searchers to shift focus. In any case, reports on how searchers employ sketching and informal structures on displays are valuable (e.g., Shipman et al., 1995; Spink & Goodrum, in press).

We have been particularly concerned with contextualizing the design claims of SketchTrieve, and it is for this reason that the Cognitive Dimensions Framework has been used. Carroll (1994), while agreeing with Landauer on the necessity of empirical usability methods, also claims that ‘‘it is not at all clear that we can or should seek to ensure usability and usefulness only through purely empirical means’’ (p. 34). The framework, an analytic tool, has proven to be effective for conceptualizing new interfaces for information seeking, and for summarizing the design claims embodied in SketchTrieve. It is an alternative to scenario-based representations.

More generally, there is another problem to which the framework might be put. One the one hand, the field of information retrieval has a remarkable tradition of user-oriented studies (Ingwersen, 1996). However, the field has not been able to codify, in widely accessible form, how particular structural features of interfaces shape information-seeking behavior. Perhaps the framework, or one like it, could be used to generate a catalog of descriptions of exemplary IR applications on which empirical data has been collected. In this way, the behavior of searchers might be better accounted for. Petre and Green (1992) put it this way: ‘‘Understanding real tools for real users needs case law, a body of observations showing how generalizations about tools can be instantiated in specific contexts’’ (p. 67). Such a catalog would be a valuable design resource.

In conclusion, the data-flow notation of SketchTrieve is one form of secondary notation that results in an informal environment for information seeking. Other forms may prove to be ultimately more effective, and there are certainly costs associated with secondary notation, for example, the viscosity of the display and potential difficulties with role expressiveness. It would be surprising, however, if secondary notation did not have an important role to play for the searcher because such a conclusion would seem to contradict the general picture of end-user programming environments, including the spreadsheet. SketchTrieve begs questions about how interactive displays can enrich information seeking, and make it more lively, rather than merely permit it.

Acknowledgments

Comments from anonymous reviewers, and Andrew Dillon’s incisive criticism, significantly improved the presentation. Any remaining faults are, of course, our responsibility. Thanks go to the participants who tried SketchTrieve and taught us, and to Tore Bratvold, Ian Campbell, Matthew Chambers, Jan-Jaap Jdlens, and Joemon Jose whose comments improved its design. Our thanks to Hans-Peter Frei and Gabriele Sonnenberger for their collaboration. This work is generously supported by the Principal’s Research Fund at The Robert Gordon University and by UBILAB of the Union Bank of Switzerland.

References


