Application Use Strategies

Suresh K. Bhavnani

Strategies for using complex computer applications such as word processors, and computer-aided drafting (CAD) systems, are *general* and *goal-directed* methods to perform tasks. These strategies are important to identify and learn because they (1) can make users more efficient and effective in completing their tasks, and (2) are often difficult to acquire just by knowing commands on an interface.

To understand strategies to use computer applications, consider the task of drawing three identical arched windows in a CAD system. As shown in Figure 1A, one way to perform the task is to draw all the arcs across the windows, followed by drawing all the vertical lines, followed by drawing all the horizontal lines. Another way to perform the same task (Figure 1B) is to draw all the elements of the first window, group the elements, and then make three copies of the grouped elements.

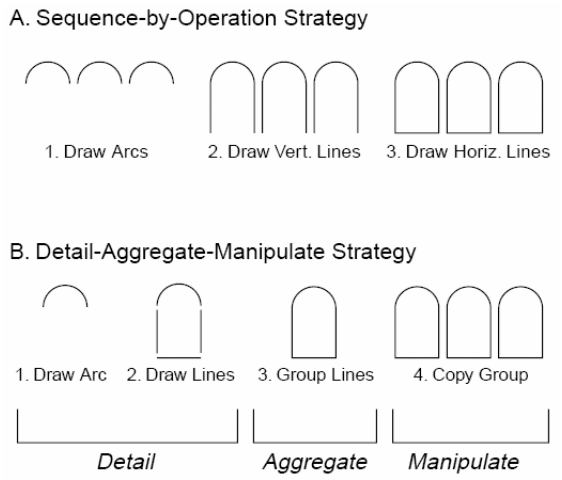

Figure 1. Two strategies to perform the 3-window drawing task. (Source: Bhavnani, S.K., and John B.E. (1996). Exploring the unrealised potential of computer-aided drafting. Proceedings of CHI'96, 337. Copyright 1996, ACM, Inc. Reprinted by permission.)

The first method is called *Sequence-by-Operation* because it organizes the drawing task by performing one set of identical operations (like *draw arc*), followed by performing the next set of similar operations (like *draw line*). The second method is called *Detail-Aggregate-Manipulate* because it organizes the task by first drawing all the elements of the first window (Detail), grouping that set (Aggregate), and then making copies of that set (Manipulate). Both the above methods are *strategies* because they are general and goal-directed. For example, the Detail-Aggregate-Manipulate strategy is general because it can be used to create multiple copies of sets of objects in a wide range of authoring applications, such as creating many identical paragraphs to create address labels in a word processor (e.g. Microsoft Word). The Detail-Aggregate-Manipulate strategy is also goal-directed because it can be used to complete the task of drawing three arched windows.

The definition of a strategy described in the above paragraphs subsumes more limited strategy definitions used in different fields ranging from business management to cognitive psychology. These definitions include various concepts such as time (e.g. a long-term plan for achieving a goal), the

existence of alternate methods (e.g. any method that is non-obligatory), and performance outcomes (e.g. methods resulting in a competitive advantage). However, exclusion of these concepts (time, existence of alternate methods, and performance outcomes) from the definition of a strategy enables us to describe strategies in a more encompassing way irrespective of whether they are short-term or long-term, unique or one-of-many, and efficient or inefficient.

The costs and benefits of using strategies

Although the two strategies shown in Figure 1 achieve the same goal, they have different costs and benefits associated with their use. By drawing all the arcs before the lines, the Sequence-by-Operation strategy reduces the cost of switching between the *draw arc*, and the *draw line* commands. Furthermore, the strategy uses simple commands that are useful to perform a large set of tasks. Therefore, the short-term *learning* cost of using this strategy is small. However, because the user is constructing every element in the drawing, the *performance* cost (measured in terms of time and effort) can become large when drawing repeated elements across many tasks especially in the long term. In contrast, the Detail-Aggregate-Manipulate strategy requires the user to draw the elements of only one window, and makes the computer construct the rest of the windows using the *group*, and *copy* commands. For a novice CAD user, the short-term learning cost for the Detail-Aggregate-Manipulate strategy involves learning the *group* and *copy* commands, and how to sequence them. However, as is common in the use of any new tool, this short-term learning cost is amortized over the long term because of the efficiency gained over many invocations of the strategy. This amortization therefore lowers the overall performance cost.

Research has shown that strategies like Detail-Aggregate-Manipulate can save users between 40%-70% of the time to perform typical drawing tasks, in addition to reducing errors. Furthermore, with properly designed strategy-based training, such strategies can be taught to novice computer users in a short amount of time. For users who care about saving time and producing accurate drawings, learning such strategies can therefore make them more efficient (save time) and more effective (reduce errors) with relatively short training.

A framework to organize strategies for complex computer applications

Given the important role that strategies can play in improving overall productivity, a central research goal has been to identify and organize strategies to use computer applications such as authoring and information retrieval applications. Frameworks to organize strategies suggest the design of: (1) training that teaches the strategies in a systematic way, (2) new systems that provide effective and efficient strategies to users with little experience, and (3) evaluation methods to ensure that designers consistently offer the commands for using efficient and effective strategies.

One useful way to organize strategies is based on the *powers* or general capabilities (that can be realized through appropriate commands) of computer applications that the strategies exploit. For example, the Detail-Aggregate-Manipulate strategy described in Figure 1 exploits the *iterative* power of computers because the computer (instead of the user) performs the repetitious task of copying the elements multiple times. Strategies have also been identified that exploit other powers of computers such as the powers of *propagation*, *organization*, and *visualization*.

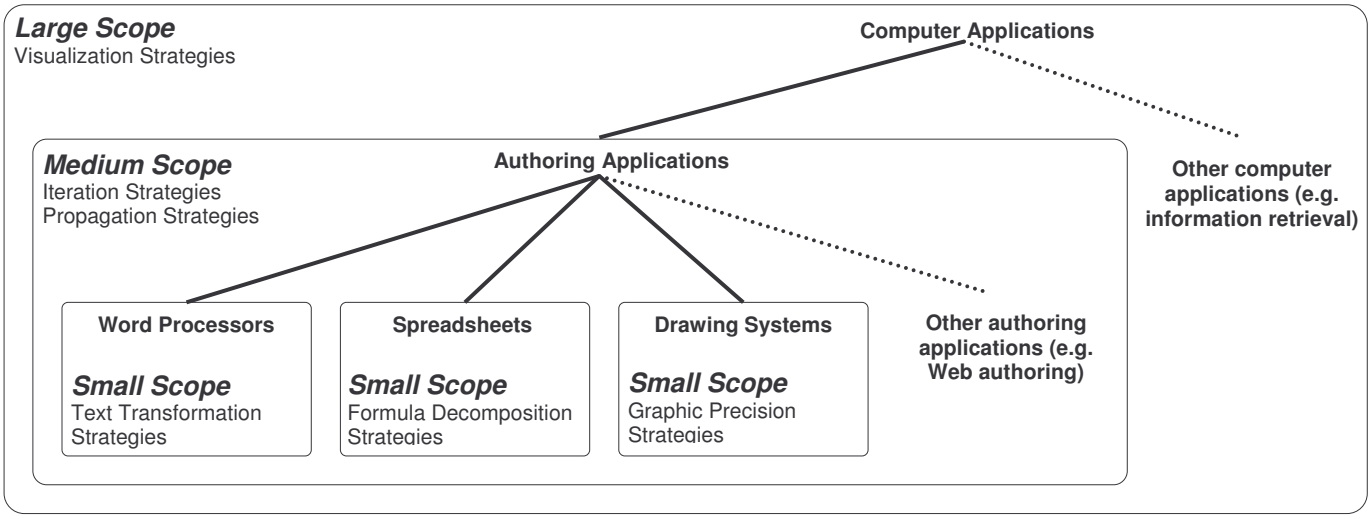

Figure 2. Strategies have been identified to exploit different powers of computers at different scopes levels. Large scope strategies are useful to many classes of computer applications, such as authoring and information retrieval applications. Medium scope strategies apply to a single class of computer applications, such as authoring applications. Small scope strategies apply to a single sub-class of applications, such as only to word processors. The dotted lines represent how future strategies can be included in the framework.

Another way to organize strategies is by the *scope* of their use. For example, some strategies are broad in scope because the powers they exploit are offered by a large range of computer applications such as authoring and information retrieval applications. Other strategies are narrower in scope and applicable to a smaller range of computer applications such as only to word processors.

Figure 2 shows how strategies have been organized in terms of computer powers *and* the scope of their use: (1) *Large scope* strategies are useful across more than one class of computer applications, for example strategies that exploit the power of visualization useful in all computer applications with a graphical user interface. (2) *Medium scope* strategies are useful within one class of computer applications such as strategies that exploit the powers of iteration and propagation in authoring applications. (3) *Small scope* strategies exploit capabilities that are useful within one sub-class of computer applications such as strategies to exploit the power of graphic precision in drawing applications. Each of these sets of strategies is discussed below.

Large scope strategies useful across more than one class of computer applications

Given the wide permeation of Graphical User Interfaces (GUI) across computer applications, most useful computer applications require some interaction with a visual interface. Such computer applications offer the power of *visualization*, which essentially provides ways to selectively view information on the screen. For example, a common word processing task is to compare information, which exists in one part of a document, to information in another part of the document. When these two parts of the document cannot fit simultaneously on the screen, the user can perform the comparison task in several ways. One way is to scroll back and forth between the relevant parts of the document. This method is time-consuming and error-prone because it requires the user to remember the information that is not visible. Another way to perform the same comparison task is to first bring together on the computer screen the two relevant parts of the document, before comparing them. The information can be brought together on the screen by different commands, such as by opening two windows of the same

document scrolled to the relevant parts of the document, or by using the *split window* command in Microsoft Word to view two parts of the document simultaneously.

In addition to being useful for word processing tasks, this visualization strategy is also useful when one is drawing a complex building in a CAD system, or when one is comparing information from two different webpages when retrieving information on the Web. Hence strategies that exploit the power of visualization have wide scope spanning many different classes of computer applications (represented as large scope strategies in Figure 2).

Medium scope strategies useful within one class of computer applications

While visualization strategies have the widest use across classes of computer applications, there are three sets of strategies that are limited in scope to only one class of computer applications:

1. Strategies that exploit the *iterative* power of computers, such as the Detail-Aggregate-Manipulate strategy (shown in Figure 1), are useful mainly for authoring applications such as drawing systems and word processors. For example, besides being useful to draw repeated graphic elements in a drawing system, the Detail-Aggregate-Manipulate strategy is useful in a word processor to create multiple copies of address labels, or copies of a table in a spreadsheet.

2. Strategies that exploit the power of *propagation* provided by authoring applications enable users to set up dependencies between objects, which enable modifications to automatically ripple through the dependent objects. For example, often users have to change the font and size of headings in a document to conform to different publication requirements. One way to perform this task is to manually change the font and style of each heading in the document to conform to the publication requirement. This is time-consuming especially when the document is long, and error-prone because certain headings may be missed, or incorrectly modified. A more efficient and effective method to perform the same task is to first make the headings in a document dependent on a *style definition* in Microsoft Word. When this style definition is modified, all dependent headings are automatically changed. This strategy is useful across different authoring applications such as spreadsheets (to generate different results by altering a variable such as an interest rate), and CAD systems (to generate variations of a repeated window design in a building façade).

3. Strategies that exploit the power of *organization* provided by authoring applications enable users to explicitly structure information in representations (such as in a table). These explicit representations enable users to make rapid changes to the content (such as adding more text to a table cell), without having to manually update the structure of the representation. For example, one way to represent tabular information in a word processor is by using tabs between the words or numbers. However, because tabs do not convey to the computer an explicit tabular representation consisting of rows and columns, the tabular structure may not be maintained when changes are made to the content. A more efficient and effective way to perform this task is to first make the table explicit to the computer by using the command *insert table*, and then to add content to the table. Because the computer has an internal data structure that represents the structure of a table, the tabular representation will be maintained during modifications (such as adding more content to a cell in the table). Organization strategies are also useful in other authoring applications. For example, information can be stored using a set-subset representation in a spreadsheet (such as using different *sheets* to organize sets of numbers), and in a CAD system (such as using different *layers* to organize different types of graphical information).

As discussed above, strategies that exploit the powers of iteration, propagation, and organization are useful mainly for authoring applications (represented as medium scope strategies in Figure 2). However, it is important to note that the powers of iteration, propagation, and organization can also be offered by other classes of computer applications such as by information retrieval applications. For example, many web browsers offer users ways to organize the addresses of different retrieved webpages (such as the organizing features provided by the *favorites* command in Internet Explorer). However, while powers provided by authoring applications can be provided in other classes of computer applications, the strategies that they exploit them will tend to be the same.

Small scope strategies useful within one sub-class of computer applications

Strategies at the most specific level (small scope) in the framework exploit powers provided by particular sub-classes of applications. For example, the power of graphic precision is offered mainly by drawing systems (such as CAD systems used by architects and engineers). Strategies that exploit graphic precision enable users to create and manipulate precise graphic objects. For example, a common precision drawing task is to create a line that is precisely tangent and touching the end of an arc (as shown in the arched windows in Figure 1). One way to perform this task is to visually locate, and then click the end of the arc when drawing the line. This is error-prone because the user relies on visual feedback to detect the precise location for the end of the arc. Another way is to use the *snap-to-object* command, which enables the user to click a point that is only approximately at the end of the arc. The computer responds by automatically locating the precise end of the arc, and therefore enables the user to draw a line that is precisely tangent to, and starting at the end of the arc.

Similar small scope strategies (useful within a single sub-class of computer applications) have been identified for word processors (such as those that assist in transforming text to generate summaries, translations etc.), and for spreadsheets (such as decomposing formulas into sub formulas to enable quick debugging).

Future extensions of the strategy framework

The strategy framework in Figure 2 has focused on authoring applications. However, the framework can also be extended to organize the large number of search strategies that have been identified to use information retrieval applications such as general-purpose search engines (e.g. Google). In contrast to computer powers that have been useful in organizing strategies to use authoring applications, strategies to use information retrieval systems appear to be driven by attributes of how information sources are *structured*. For example, a large portion of the Web contains densely connected Web pages referred to as the *core* of the Web. The densely connected structure of information sources in the core suggests the importance of using a variety of browsing strategies (that rely on using hyperlinks to move from one page to another) to locate relevant sources. In contrast to the core, there is a large portion of the Web that consists of new pages that do not have many pages that link to them. Strategies to find these pages therefore require the combination of different query-based search engines to locate them, given that no single search engines indexes all web pages.

While there has been much research on strategies to find relevant sources of information, a new set of strategies are being identified to determine how to select and order relevant sources of information because of the way information is *distributed* across sources. For example, facts about healthcare information are typically scattered across different healthcare portals. This situation requires strategies to visit specific kinds of portals in a particular order to enable a comprehensive information

accumulation of the relevant information. Such strategies become critical in domains (such as healthcare) where incomplete information can lead to dangerous consequences.

Another important difference between strategies to use authoring applications and strategies to use information retrieval systems is that search strategies are fundamentally *heuristic* (rules of thumb that do not guarantee successful task completion). This is in part because of the users' shifting evaluation criteria of what is relevant based on what is being learned during the search process.

How the identification of strategies can be applied to improve designs

The identification and analysis of strategies to use computer applications suggests the following three practical uses:

1. Design of strategy-based instruction. Strategies to use authoring applications have led to the design of strategy-based instruction. Strategy-based instruction teaches commands in combination with the authoring strategies shown in Figure 2. Research has shown that students who took the strategy-based training acquired more efficient and effective strategies, and demonstrated a greater ability to transfer that knowledge across applications compared to students who were taught only commands.

2. Design of new search systems with search procedures. The identification of search strategies to deal with the scatter of information across the Web has led to the design of a new kind of domain portal called a Strategy Hub. Such a domain portal instantiates heuristic search strategies to visit sources of information in a particular order. Recent studies show that such a system is more effective to enable users find comprehensive information for specific topics, when compared to the information retrieved by users of other search systems.

3. Design of an analysis method to ensure consistency in powers across applications. To enable the widest use of strategies across computer applications, designers must provide a consistent set of commands. Therefore, a method called Designs Conducive to the Use of Efficient Strategies (Design-CUES) has been developed that enables designers to systematically check if their designs provide the commands necessary for users to implement efficient and effective strategies.

Summary

Many years of research has shown that learning commands for using complex computer applications is not enough to complete tasks efficiently and effectively. The effective and efficient use of computer applications often requires the use of strategies in addition to commands. An important research goal has therefore been to identify strategies to use a wide range of computer applications, which has resulted in the identification of strategies that exploit different powers of computers, at different scope levels. These strategies have been used to design strategy-based instruction, new forms of search systems, and new design methods, with the ultimate goal of making users more effective and efficient in the use of complex computer applications.

Further reading

Singley, M., & Anderson, J. (1989). *The Transfer of Cognitive Skill*. Cambridge, MA: Harvard University Press.

Bates, M. (1979). Information search tactics. *Journal of the American Society for Information Science 30*, 4, 205–214.

Bates, M. J. (1998). Indexing and access for digital libraries and the Internet: human, database, and domain factors. *Journal of the American Society for Information Science, 49,* 13, 1185-1205.

Belkin, N., Cool, C., Stein, A., & Thiel, U. (1995). Cases, scripts, and information-seeking strategies: on the design of interactive information retrieval systems. *Expert Systems with Applications 9*, 3, 379-395.

Bhavnani, S. K. (2002). Domain-specific search strategies for the effective retrieval of healthcare and shopping information. *Proceedings of CHI'02*, 610-611.

Bhavnani, S. K. (in press). Why is it difficult to find comprehensive information? Implications of information scatter for search and design. *Journal of the American Society for Information Science*.

Bhavnani, S.K., & John, B.E. (2000). The strategic use of complex computer systems. *Human-Computer Interaction, 15*, 107-137.

Bhavnani, S.K., Reif, F. & John, B.E. (2001). Beyond command knowledge: Identifying and teaching strategic knowledge for using complex computer applications. *Proceedings of CHI' 01*, 229-236.

Bhavnani, S.K., Bichakjian, C.K., Johnson, T.M., Little, R.J., Peck, F.A., Schwartz, J.L., & Strecher, V.J. (2003). Strategy Hubs: Next-generation domain portals with search procedures. *Proceedings of CHI'03*, 393-400.

Drabenstott, K. (2000). Web search strategies. In W.J. Wheeler (ed), *Saving the User's time through subject access innovation; Papers in honor of Pauline Atherton Cochrane*, 114–161. Champaign, Ill: University of Illinois.

Mayer, R. E. (1988). From novice to expert. In M. Helander (ed.), *Handbook of Human-Computer Interaction*, 781-796. Amsterdam: Elsevier Science.

O'Day, V., & Jeffries, R. (1993). Orienteering in an information landscape: how information seekers get from here to there. *Proceedings of CHI '93*, 438-445.

Shute, S., & Smith, P. (1993). Knowledge-based search tactics. *Information Processing & Management, 29*, 1, 29-45.

Siegler, R.S., & Jenkins, E. (1989). *How Children Discover New Strategies*. New Jersey: Lawrence Erlbaum Associates.

Acknowledgements

This research was supported by the National Science Foundation, Award# EIA-9812607. The views and conclusions contained in this document should not be interpreted as representing the official policies, either expressed or implied, of NSF or the U. S. Government. The author thanks M. Bates, B. John, S. Mathan, F. Peck, F. Reif, and G. Vallabha for their contributions.