Toward a Human Centered Scientific Problem Solving Environment

Version # 2
4/1/98

Thomas T. Hewett
and
Jennifer L. DePaul
Department of Psychology/Sociology/Anthropology
Drexel University
Philadelphia, PA 19104 USA
+1 215-895-2461
hewett@drexel.edu
Abstract

The goal of this paper is to review some basic human computer interaction considerations and to describe some cognitive science research which can be used in developing designs for scientific problem solving environments. First we argue for the importance of taking account of the user’s goals and task-oriented needs and draw a distinction between interface and interaction design, arguing that both levels of design must be addressed in human centered design. We then discuss the critical role of analogical thinking in the development of scientific knowledge and consider some of the implications of the need to allow for analogical thinking in designing problem solving environments. Finally, we conclude with some generic recommendations for features which should be given consideration in making choices among design alternatives. Along the way we provide examples or pointers to current PSE research projects which illustrate a number of the points being discussed.

Keywords: human centered design, problem solving environments, interaction design, interface design, scientists, scientific computing, knowledge workers

Introduction

Probably the single most hoary admonition in human factors, in ergonomics, and in human computer interaction (HCI) is, "Know the user." However, this admonition might better be formulated, "You have a model of the user, what is it?" Anyone contemplating design of a computational tool or environment has a conception of the intended user group, the tasks those users want to perform, and how they perform their tasks. Problems often arise simply because the designer does not recognize discrepancies between these models and reality.

A problem solving environment (PSE) is an integrated collection of software tools that facilitates problem solving in some domain. While a particular scientific or engineering problem domain will dictate the nature and range of symbolic and numeric computational tools needed in a PSE, it is helpful to step back from a focus on computational tools and consider the task-oriented needs and goals of the users. Although some of the points about PSEs made here are described or are implicit in the work of others,\(^1\)\(^8\) the focus and motivation of this paper are quite different. For example, the major point to be argued here is that overall interactive system design criteria and performance metrics should be weighted to include the costs of human time and cognitive load as well as demands on machine cycles. (In a great many cases, human cycles cost more than machine cycles.) In the past, work on interfaces to PSEs
has been thought of as an important supporting area of work on PSEs. Effectively, this paper argues that the time has come to treat the human interaction with a PSE as being a component of the PSE which is as central and as important as Symbolic Computing, Numerical Analysis, or Visualization.

**Interfaces and Interactions**

In thinking about HCI aspects of what does or does not make it possible for a user to make effective use of a PSE, there are two different levels of design criteria that are important. One level is the interface or “tactical” level of design. The second level is the interaction or “strategic” level. While there are a few reports specifically discussing user interface issues for PSEs, they typically do not distinguish between design considerations operating at the interface and interaction levels.

At the interface level, design criteria address issues such as how interface components should be built and how they should be shaped. For example, aircraft cockpit instruments need to show certain necessary information clearly. Similarly, in a wide range of circumstances, knobs, switches and dials need to be easily discriminable by sight and/or by touch. Interaction level design criteria address the overall goals of the user and take into account the way in which the user thinks about the task. A concern with interface features is important since badly designed components can make the user’s task more difficult, lead to unnecessary errors, and even prevent successful task accomplishment. However, optimizing interface features without paying close attention to the overall flow of interaction can also lead to unworkable or unusable systems.

As an example of the difference between interface and interaction level considerations and an example of the importance of attending to both, consider the following “hypothetical” situation. Imagine a high-end scientific visualization program running under a Graphical User Interface (GUI). Within the visualization window a 3D object is represented, e.g., a complex surface in grid mode with portions of the surface wrapping around, intersecting with, and concealing other portions of the surface.

For manual control of rotation the software provides two rotator wheels. In the lower right hand side of the window “frame” appears the first rotator wheel. The user can “grasp” the wheel with the pointer and move either upwards or downwards to signal the computer to rotate the visual object around its horizontal axis. In the window “frame” across the bottom of the window is another wheel controlling rotation about the vertical axis. Each wheel is carefully designed to recreate the appearance of physical rotator wheels, even to creating the visual effect of having the top and bottom
(or right and left) edges of the wheel appear to recede behind the plane of the scroll bar. The objects are clearly rotator wheels and their intended function is obvious even to an inexperienced user.

Now consider the following “hypothetical” interaction scenario. Assume that the researcher wants to be able to visually discriminate certain internal aspects of the structure which are partially concealed by outer portions of the surface. Further assume that the 3D representation is computationally intensive and challenges the power of the hardware. Autorotate will be unsatisfactory in this case since the rotation of the figure will be slow, or worse, slow and jerky. However, even if rotation is smooth, the view of interest to the user may be visible for only 5-10 degrees of rotation. Then it is necessary for the user to wait. Meanwhile the incompletely grasped mental representation inside the user’s head begins to deteriorate. Other cognitive events occur, further disrupting the user’s ability to mentally integrate information across time and cognitively construct a “mental picture” of the details. Compounding the difficult is that the less one understands to begin with, the less one is able to “see” in the figure. even when there is a temporary open view of the internal structure.

In addition, there are contexts in which manual control of rotation will be equally unsatisfactory, but for a very different reason. With a computationally intensive figure, there will be a time lag between movement of the rotator wheel and the appearance of the redrawn figure on the screen. Now, assume there is no visible cue by which the computer signals the user, “I’m processing.” Seeing no visible result of having moved the rotator wheel, and being uncertain of the degree of rotation effected by this first attempt to rotate the 3D object, the user “grasps” the wheel and tries again. Naturally, this action aborts the earlier processing and tells the machine to compute the new position of the object. Still having had no feedback after a second and third attempt to rotate the figure, the user next tests the horizontal rotator wheel. Again there appears to be no direct result. (Meanwhile the computer has been busily and patiently recomputing a new position of the 3D object each time it has been instructed to do so.) Finally, the user pauses in manipulation of rotator wheels to try and determine what is being done wrong. Suddenly, with no apparent transition, the 3D object appears in its most recently computed position.

Although the user “knows” this is the same object from a few moments ago, it is extremely difficult to see any relationship between the new view of the object and the original one. Having no idea how to move the rotator wheels in such a way as to return the 3D object to its initial position, the user closes the file and restarts from scratch. Eventually a novice user’s frustration level may reach the point at which the
computational tool is abandoned as being unwieldy and unsatisfactory. A user who has a need which can only be fulfilled by this particular tool will persevere and eventually learn though hard experience to calibrate movement of the mouse to an approximate rotation on the screen. But, why should the experience be hard?

A bigger, faster machine won’t really solve the real problem here. While it is true that a really nice interface on an under powered machine won’t serve the user’s needs, the user’s frustration is not determined solely by the fact that the machine takes time to compute each new view of the 3D object. Within limits, time is a minor irritant compared to the fact that the user is unable to control viewing of critical information in the 3D representation. The control afforded by the rotator wheel is not appropriate to the user’s needs. There is no simple and direct way for the user to calibrate physical control movements to results. Too many cognitive operations intervene between views of the critical information.

What is needed here is some interface level mechanism which makes possible and facilitates the interaction the user wants or needs to have. What is needed here is a mechanism where the user is not required to look away from the 3D object to control its rotation and is therefore able to concentrate attention on the object rather than the interaction with an interface component. This concentration of attention on the task at hand facilitates the ability of the user’s visual system to integrate across time in developing an uninterrupted mental representation of the object being rotated on the screen. Our ability to design and provide such an interface mechanism requires recognizing that it is needed in the first place (i.e., that the user has this goal) and then developing various pieces of the interface to make it possible.

**Interaction design.** As illustrated above, even with locally well designed interface components a tool can fail to support the user’s interaction needs unless care is also taken to understand and support the user’s task goals. Focusing on the problem of interaction design, Norman has offered a series of ideas about how to design artifacts that are both usable and useful. The model he has developed involves what he calls the "seven stages of action." To complete an action successfully, the user must first formulate one or more goals. Once a goal has been established there are then six additional steps which must be taken for the user to determine whether or not the goal has been accomplished. The user must: 1) formulate an intention to act; 2) specify the action(s) to make; 3) execute the actions successfully; 4) perceive the state of the world; 5) interpret the state of the world, and 6) evaluate the outcome.

The first three of these steps (1, 2, & 3) Norman calls, "The Gulf of Execution," and the second set of three (4, 5, & 6) he calls, "The Gulf of Evaluation." If any step doesn’t
work properly either the user won't make progress towards the goal, or else the user won't be able to determine the outcome of actions taken. The human centered PSE designer should facilitate the user's tasks by helping to make it clear how to bridge the Gulf of Execution and by providing appropriate feedback to make it possible to bridge the Gulf of Evaluation. As noted above, one prerequisite to designing a PSE which facilitates tasks is understanding what those tasks are and how the user thinks about them.

At a general level, some guidance in how a PSE designer might help the user bridge the Gulfs of Execution and Evaluation can be found in a set of design principles proposed by Norman. The first design principle is that the designer should enable the user to make use of both knowledge in the head and knowledge in the world. Knowledge in the head is that knowledge which is stored in human memory. However, we often can and do store only partial knowledge in memory and make use of the presence of knowledge in the world to assist in guiding our actions. For example, most people have little trouble dialing a phone number such as 555-PUCK to get hockey tickets. However, with no telephone dial or set of push buttons visible, many of those same people have to reconstruct rather than remember the letters associated with each number. (They also have difficulty identifying the two letters of the alphabet which do not appear.) Similarly, the typical GUI functions in part as an external memory storage device we have created for the user. Since the GUI stores many commands on the screen where they can be easily located or found with minimal search time, the interface components provide memory cues which reduce the user’s internal memory burden and processing load.

A second design principle identified by Norman is that the designer should seek to simplify the structure of the tasks which the user needs to perform. This simplification can involve several different options. One is to keep the task the same and provide one or more mental aids (e.g., the calendar for people who have to do scheduling; e.g., a symbolic computing engine which can send numeric output to other tools). Another option is to improve feedback and the user’s ability to maintain control by using technology to make things visible that would otherwise not be visible (e.g., instruments in the automobile; e.g., processing flow in a parallel program). A third option is to automate portions of the task without radically altering the actual flow of the task (e.g., provide turn signals for automobiles rather than require arm and hand signals; e.g., provide a consistency checker which determines if there are missing parentheses in a complex equation; e.g., develop a notation translator which facilitates conversion from one notation to another). The fourth option described by Norman for simplifying the
structure of the user’s tasks involves actually changing the nature of the task to make it more easily performed (e.g., learning to tell time is much easier with digital watches than with analog; e.g., some navigational instruments require positioning of a slide and reading of results rather than calculation of numbers).

Norman’s third design principle is to make things visible so that the user can more easily bridge the Guls of Execution and Evaluation (e.g., the Captain of an ocean going ferry boat should have an indicator light on the bridge which clearly signals that the front deck doors which prevent large waves from washing over the automobile deck have been closed). A fourth design principle is to get the mappings right. That is, the results of users’ actions should match their expectations. For example, the motion of a throttle should require a forward push, as should the motion of other controls associated with an increase in speed. Controls to rotate 3D visualizations should move or point in the direction of intended motion and there should be a direct, obvious tie between action and consequence. One of the more important of Norman’s principles is that the designer should design for error. The designer should assume human error will occur and allow for error by making it possible for the user to be able to identify when an error has occurred and to be able to cope with or even avoid the undesirable consequences of the error.

The user, the task, and successful performance. A clear implication of Norman’s work is that one central goal for human centered PSE design should be an understanding of the researcher’s tasks and how the target users proceed in solving their research problems. Adopting this goal immediately commits one to developing explicit answers to several questions. Exactly who are the target users, what knowledge are they assumed to have, and what are their skills? What are the specific tasks such users will want to be able to complete and are there generic types of tasks or task characteristics? How are the criteria established by which successful user performance will be measured? Is it possible to determine metrics for successful use of a scientific PSE?

As a number of sources on Task Analysis have pointed out, the user’s task can have a significant impact on how tools are used. For example, suppose we intend to design a PSE for engineers. Our PSE design decisions need to take account of the fact that there appear two broad classes of engineering design problems. One type of problem involves solving a novel instance of a fairly well understood class of problems. The other type of engineering design problem involves solving a novel instance of a novel class of problems which is not yet well understood. The Civil Engineer designing a bridge to carry a secondary road over a new expressway typically has a somewhat
different set of task demands than those that exist for a Civil Engineer designing an entirely new type of bridge. The tools needed in a PSE may well differ radically in these two cases, and even if the computational tools are the same, the ways in which those tools will be applied will be very different.

In fact, a PSE designed for users working with novel instances of a well understood class of problems might well be unusable by users developing a solution to one or more instances of a novel class of problems. For example, consider the use of computer aided design (CAD) systems in architecture. It is possible to divide architectural design into two broad categories of activity. One category is that set of activities associated with preparation of a model or design representation to illustrate the structure or concept to a customer. Another category of activity is that set of activities associated with sketching out and developing a novel design concept in the first place. Even casual observation of usage patterns of architectural CAD systems suggests there are important task differences. While it is not unusual to find someone trained in the use of an architectural CAD system putting it to use in polishing and preparing drawings for presentation, it is quite a challenge to find anyone using such a CAD system while creating an original design concept.

A PSE user can be expected to do an approximate cost benefit analysis on the software functionalities, ignoring or not using those which do not return good value for the effort or time invested. In addition, it is important to understand approximately how steep the learning curve is for various features so as to be able to estimate how much training time may be required and to understand why someone who tries out a PSE may abandon its use. In other words, it is important to understand both the nature of the work required by the user's task and the relative amount of work required to make use of various PSE functionalities.

**Analogical thinking: cognitive task aspects of scientific problem solving.** One place to begin identifying task critical functionalities and evaluation criteria for a human centered PSE is with a realistic understanding of the current procedures, practices and activities which the PSE is meant to augment or replace. While Norman’s work addresses the general design problem of taking account of the users’ goals and tasks, it is possible to be more specific about some of the characteristics of the tasks and goals of users likely to be working with the tools in a scientific PSE. One way to enhance our understanding is to take a selective look at some of the research on scientific problem solving.

There are a variety of reasons for believing analogical thinking is a key process involved in creative advances in science. Of particular relevance to thinking about the
design of PSEs is work on the use of analogy in individual scientific problem solving. Another important area of research involves study of the use of analogy in generating new insights during the social process of science and in facilitating communication about research.

Clement demonstrated that the steps used by scientists in developing a problem solution are not necessarily derivational or formal. Clement gave a mechanics problem to 10 participants who were sophisticated in Physics but had no extensive experience with solving problems from the class of mechanics problems used. Participants in the study ranged in expertise level from graduate students to a Nobel laureate in Physics. The participants were asked to “think aloud” while solving the problem. Careful examination of video tapes from these sessions revealed that only one participant used anything resembling an abstract principle to derive a solution to the problem. The typical solution procedures involved memory search for an analogous situation with similar characteristics. The search was conducted through association or by examining dimensions of similarity. The typical solution then involved the development of a possible solution on an analog which had been judged to be appropriate. Once the solution had been worked out for the analog, the individual then proceeded to test the solution on the original problem.

Dunbar reported on a year long study conducted in four molecular biology laboratories where he followed all aspects of several projects. The processes studied included, “planning the research, executing the experiments, evaluating the experimental results, attending laboratory staff meetings and public talks, planning further experiments, and writing journal articles (p. 365).” In the conduct of this study Dunbar was given free access to the laboratories and was allowed to interview anyone in the laboratory. He attended meetings, read and kept copies of grant proposals, attended talks and lectures, and read drafts of papers. In addition he was frequently invited to attend impromptu meetings and discussions or to witness events occurring in the lab.

Dunbar reports that, “analogies were an important source of knowledge and conceptual change (p. 381).” There were three different classes of analogy used by the researchers studied. **Local analogies** were those from within the same domain as the research being conducted and were drawn from a previous experiment to a current one. This type of analogical reasoning typically occurred when a scientist would attempt to map a currently unsuccessful experiment onto a similar experiment which had been successful. These local analogies seemed to be one of the main mechanisms for driving the research program forward. **Regional analogies** involved mapping a system of
relationships from a similar domain onto the problem domain being studied. This type of analogical reasoning was typically, “employed when the scientists were working on elaborating their theory and planning a new set of experiments (p. 383).” These regional analogies seemed to be one of the main mechanisms for allowing the scientists to fill in gaps in their own knowledge and to suggest new questions to ask about the entities being studied. **Long-distance analogies** involved mapping a concept from a very different domain onto the problem domain being studied. Dunbar did not find this type of analogy being used in solving experimental problems or in doing theory or model building. Typically, “long-distance analogies were used to highlight features of the research that were salient,” and were used to, “bring home a point or to educate new staff members of a laboratory (p. 383).”

**Mathematical problem solving: one type of PSE user, his goals and his needs.** To provide an example of some of the points made above, we will describe some of our work with a researcher doing work with symbolic mathematics. As a part of a larger project aimed at integrating symbolic computing into a scientific PSE, we are currently working with a research mathematician to try and understand how he does his work and what types of computational tools are needed to support that work. The mathematician wants to produce geometric representations of a particular class of equations known as soliton equations. About twenty five years ago, certain equations which are highly complicated were found to be exactly solvable. These types of equations, complex equations with simple behavior, are referred to as soliton equations. There are a number of such equations known. Two examples of equations which behave in this manner are the equations for fiber optics and the equations for smoke rings (or tornadoes). The mathematician being studied has found that these different equations share subtle correspondences. Furthermore, he believes that the soliton equations are so similar that there must be a generalized geometric analog, a curve moving in some space (within or beyond the third dimension), which is common to all soliton equations. Currently, to solve the representation problem, the mathematician takes the information provided in the solutions of the soliton equations, the bend and the amount it leaves a plane, and converts that information to create a geometric representation.

Through a series of interviews, conducted as preliminary information gathering sessions before directly observing problem solving, some interesting things were learned about the problem solving behavior of the mathematician. Although the information provided here was self-reported by the mathematician and was reported post-hoc, there is reason to believe that the issues discussed in the interviews are
representative. The behaviors described by this mathematician parallel those found in experts in other fields.\textsuperscript{13-16}

In the past year, the mathematician has put together a paper in which he argued that higher dimensions play a role in forming the representations of some soliton equations. With this knowledge he then found a possible representation for the particular soliton equation he was working on solving, a representation which exists in some n-dimensional space. Since this type of solution is very difficult to work with and since most people have trouble comprehending higher dimensions, the mathematician’s next step is to find another soliton equation which could occur in 3D space, or to find another space in which a simplified version of the current representation could occur. To begin this next step of the problem, the mathematician is looking at the possibility of representing these equations in the three sphere, hyperbolic space, or one of several known space models where light does not travel in a straight line, but nevertheless, travels in a controlled or understood manner. His reason for turning to these representation spaces is that his knowledge of the behavior of the structure of these spaces has led him to believe the curves or surfaces of the soliton equations could potentially be characterized by the same type of behavior.

It is interesting to note that in the information we have gathered from the mathematician there are several clear instances of each of the three types of analogical thinking described by Dunbar.\textsuperscript{14} In one brief description of where he intends going next with his research, the mathematician actually utilized all three types of analogy. First, he employed a local analogy alluding to the possibility of extending his work to a different fiber optics equation. Second, he utilized a regional analogy when considering looking for a new representation in a space where light does not travel in a straight line. This is not a piece of knowledge specific to the mathematician’s area of expertise but it does fall within the same “superordinate category.” Finally, in explaining his work to one of us, the mathematician made use of a long-distance analogy. He explained that the geometric representations can be considered analogous to, “...a wire moving in space. It can twist and bend, but not stretch or shrink.”

In terms of his 3D visualization needs, the mathematician reports that he needs to be able to discretize the equations in such a way that he is able to directly perceive some of their structure. The starting object in the visualization is a curve in the plane. He reports that he utilizes rotation regularly to help gain a sense of structure and often finds that a switch in visualization modes (from solid to grid or from grid to solid) is helpful in “seeing” what happens. He reports being frustrated by the fact that the intersections of the surfaces are important and are often particularly hard to see. He
also reports that it is important to him both to see the full result surface and to see how the curve evolves frame by frame. Consequently he expresses a desire to have tools which he can use with a 3D visualization which will allow him to create an intersection focus, an edge enhancement, and a local isolation of pieces of the picture for manipulation.

**Designing for Scientists and Other Knowledge Workers**

There are a variety of reasons for thinking analogical thinking is important to the development of new insights in science. Is there reason to believe that one can support this type of thinking in the design of scientific PSEs? Some guidance can be found in the work of Candy and Edmonds who have studied a variety of knowledge workers (e.g., a racing bicycle designer, a speech scientist, etc.). In their results they have also found analogical thinking to be important when these kinds of people are engaged in various creative and problem solving activities. For example, Candy and Edmonds worked closely with the designer of the Lotus Sport racing bicycle. (The introduction of this bicycle design into Olympic competition represented a successful major reconceptualization of how racing bicycles should be designed.) Of particular importance to the work of this designer was being able to move design ideas from one analogous domain to another.

Candy and Edmonds have used their experiences studying knowledge workers as a basis for proposing software design criteria which extend quite naturally to the design of scientific PSEs. Perhaps the most critical finding which emerges from their work is that it is not possible to define in advance what the knowledge worker will do and so it is necessary to allow for a variety of task appropriate needs. Some other consistencies emerging from the work of Candy and Edmonds are that the user should, at any time, be able to: 1) take a holistic view, i.e., be able to “step back” and look at the whole picture; 2) temporarily suspend judgment on any matter; 3) make unplanned deviations; 4) return to an old idea and goal; 5) formulate problems as well as solve problems; and 6) reformulate the problem space as the conception of the problem evolves over time.

Candy and Edmonds have also proposed a process model for understanding how knowledge workers (including scientists, designers, etc.) interact with their subject matter in the development of new knowledge. Resulting from work developing Knowledge Support Systems and observations conducted of experts at work, this process model contains three major modes of human activity--Exploration and Evaluation; Generation and Invention; and consideration of Constraints and
Requirements. In Exploration and Evaluation the critical human activities consist of examining data, applying existing rules for analysis of data, along with analyzing, evaluating and refining rules for data analysis. In Generation and Invention the critical human activities consist of examination of data and creation of rules which may lead to new insights and which can be applied to other aspects of the process, e.g., developing new rules for data analysis. Consideration of the problem constraints and requirements involves identifying, clarifying, and revising constraints and requirements which limit or shape the end product.

Clearly, any computing system which is designed to make all of these activities possible in a reasonably natural fashion is going to have to pay considerable attention to how users think about the problems they want to solve. Consequently, PSE designers should seek early, on-going involvement with representative members of their target class of users as part of the design team as a way of avoiding being trapped into erroneous assumptions about the user, about the problem solving processes being supported, or about the desirable characteristics of either the interface or the interaction flow between the computer and the user.

Some Implications for Designing Human Centered PSEs

Since the design arena of interest is scientific PSEs, we would expect that a human centered PSE will incorporate a range of tools for symbolic and numeric computation appropriate for work in a particular scientific problem solving domain. Because of the nature of most computational tools we would also expect the symbolic and numeric tools in a PSE to also be useful in closely related scientific problem domains. Similarly, given the importance of visualizations, we would expect a PSE to offer powerful and flexible visualization tools. Certainly, we would also expect the tools, symbolic, numeric, and graphical, to work together easily and efficiently.

In addition, given the discussions of interaction design, analogical thinking, and some ideas about designing for scientists and knowledge workers, it is possible to suggest some design criteria to be used in development of a human centered PSE. It is also possible to suggest some ideas for the types of components or tools which are desirable in a human centered PSE. Also, the earlier discussions suggest some ideas for the architecture of how the components of a PSE ought to work or interact with each other. What follows is a “wish list” for PSEs. This list is intended to be suggestive of directions and issues rather than exhaustive. Following the discussion of this “wish list” we will provide some pointers to some existing PSEs which provide at least some degree of instantiation of one or more of the characteristics on the list.
A library of analogs. Since local and regional analogs are important in guiding scientific thinking we would expect that an ideal PSE would contain a “help” library of instances of such analogs. Certainly, part of the help library would be the standard context sensitive hyperlinked help which one expects to be available to assist users in working with individual tools. Similarly, another part of the library would be the context sensitive help which users might need for navigation or for transfer of data, etc. between tools. But most importantly, this help library would contain applied examples of tool use which might, with minor modification, serve as templates to be used solving analogous problems.

In order to support and facilitate analogical thinking these examples would be drawn from more than one problem domain and this portion of the help library would be user modifiable so that additional examples might be added. In addition the library ought to provide access to more abstract structures such as the parallel archetypes described by Chandy and his associates. However, to help ensure cross pollination from one problem domain to another, each abstract structure, whether it be a template or an archetype, etc., should have multiple instantiations associated with it so that the user is able to see something of the range of applicability of the abstract structure.

In addition, the examples and abstract structures should be user modifiable for use in solving problems from conceptually related problem domains. In other words the ideal PSE should make it possible to do with scientific problem solving across multiple domains what tools like a GUI builder enable. One is able to find a GUI one likes, borrow it, and modify it to fit. Many people use a similar process for building web pages. One finds a page which one likes, downloads it and modifies it. Usually this is done after looking at two or three other web pages for ideas before building one’s own.

Multiple representations. Since it is hard to know in advance which of several alternative problem or data representations will be most useful to a researcher, an ideal PSE should allow the researcher to model and explore an idea using multiple representations of the data or information of interest. These representations should be chosen or modifiable by the researcher so as to support and be informed by individual judgment. As in the case of the mathematician described above, each representation he uses--mathematical, 2D and 3D--offers different insights or different capabilities for working with the problem domain.

Simultaneous representations. Since it is hard to predict which data sets will be useful in helping a researcher create new associations or recognize the existence of relationships, an ideal PSE should allow different sets of data or different data representations to be viewed at the same time. In the case of the mathematician
described above, the ability to work with or manipulate two different versions of a 2D representation in a common visualization, or in overlapping visualizations, will certainly allow him to decide how to adjust the values of equation parameters, and might well allow him to discover a useful or hidden characteristic of the visualization. Similarly, suppose an epidemiologist has a database on new viruses and another on unexplained deaths. The ability to work with or manipulate both data sets in a common or in an overlapping representation or visualization might well allow the researcher to identify a possible relationship between a new virus and the unexplained deaths.

**Flexible and tailorable usage.** Since it is impossible to predict in advance which components or functions of the PSE will be needed at a given time in the development of the researcher's thinking, the ideal PSE must be tailorable enough to allow a person to pick and choose which pieces of functionality to use in different configurations. Since it is impossible to predict in advance when or how the researcher will choose or need to work or to perform a particular task, the ideal PSE must be flexible enough to allow a person to use the system or its components at any and all times with minimal interruption. One of the effective implications of this is that a variety of activities may need to be performed iteratively and with the ability to halt a process, back up, modify the input or the process, and restart. Another implication is that activities such as printing or various kinds of system housekeeping checks, etc. should be performed in the background. In addition, the researcher should be able to decide which of a variety of processes to run in the background so as to be able to go off and do other things in the foreground while waiting for processing results to emerge.

**Multiple configurability.** Since it is quite likely that a researcher will have more than one problem (or class of problems) which need attention, and may be working on problems which require multiple work sessions, the ideal PSE should make it possible for the researcher to create and to store one or more environmental configurations for later use. This ability to re-create the context of work, including intermediate results, from the last session will allow the researcher to resume interrupted tasks with a minimum of disruption and to work on different problems which are in different stages of their solution. Another implication of a capability to store particular configurations of a PSE is that certain configurations may be particularly conducive to solving one class of problems more efficiently than another. The ability to store prototype configurations will save time and thought. In addition, prototype configurations will help to reduce the possibility of errors and oversights while re-assembling a particularly useful combination of component tools.
Multiple store and find operations. Since it is often the case that new ideas emerge out of the interplay of different perspectives or points of view on a problem, the ideal PSE will allow a user to keep track of many ideas and a variety of different ideas, and to organize them in a variety of meaningful ways. Effectively the PSE should provide the equivalent of “Post-it” notes which can be easily created from anywhere within the PSE without greatly disrupting the researcher’s current activity. In addition the researcher must be able to store idea “sketches” or drafts of ideas in a way which allows easy location and referral to a particular idea. Otherwise, the ideas can and will be forgotten or lost within the system.

For example, an architect may come up with several innovative concepts while designing a building. Some of the images or concepts will not fit, however, with the final result. It should still be possible to store the images or sketches so as to ensure that details are not forgotten over time. Additionally, the researcher should be able to locate these “idea sketches” using a variety of methods, including searches on such things as the presence of a name, a word or object, a file name, a date or a title. Even though the name given to a particular idea or file may be forgotten, users can often remember such things as the date on which the idea was created, the project being worked on, or a particular person with whom an interaction took place.

Multiple database access. Since the individual researcher must be a domain expert to be able to make a contribution to the expansion of knowledge in a domain, an ideal PSE should provide a way for the person to quickly store, find and browse information about the domain, particularly new information about recent advances. At a minimum the PSE should allow the researcher to store articles, or their content, in a manner which facilitates easy retrieval of data, conclusions, experiment descriptions, etc.

Multiple communication channels. Since making a contribution to a change in the state of knowledge in the researcher’s chosen problem domain requires the ability to present and to communicate information and new results to others working on the problem domain, an ideal PSE should allow the person to easily communicate results, observations, difficulties and conclusions with other people in the field and to easily prepare effective presentation materials. However, as in the case of email, the PSE should provide ways for the person to communicate with others at times which are convenient. Interruptions during work periods should be limited to those specifically allowed by the researcher.

Logging of process and results. An important aspect of providing a problem solver with the necessary scaffolding to support problem solving in a particular domain is the creation of mechanisms which provide the memory aids or cues helpful in
reinstating their thinking processes and conclusions. Consequently, an ideal PSE should provide optional access to transaction logging and intermediate result logging so that the researcher can, if desired, use the logs both as a place to store intermediate solution steps and as a mechanism for recreating the thinking which went on in getting to a particular state of affairs or to the solution of a particular subproblem. For example, in its current instantiation, Matlab provides two different commands which allow one to find out the variables in the current workspace. After having created 10 or 15 variables, if you want to remind yourself of which variables you have created you can use one command to produce a list or another command which also gives you the size and type. In neither case do you get any context about when or what you were doing when you created the variables. For relatively simple cases with relatively few intervening cognitive events you may be able to successfully rely on your own memory. However, for more complex cases a fair amount of re-thinking may be required to recreate the mental context that existed when the variables were first created. It would be helpful to have more context.

**Ability to restructure the problem domain.** An ideal PSE may well also provide the researcher with a way of creating and maintaining a modifiable knowledge base or statement of “rules” or conditions or parameters which govern the problem domain and which are based on the researcher’s current understanding of the domain. Since problem domains change with time the researcher needs to be able to modify this knowledge base either by modifying or deleting existing rules or by adding new rules to the knowledge base. In addition it should be possible to make such changes at any time during on-going tasks. For example, suppose that a researcher has come up with the hypothesis that a particular type of virus is not the sole cause of a particular type of disorder but the knowledge base contains a rule which keeps people without that virus from being included in data sets containing samples of patients. The scientist may need to modify or eliminate this rule in order to get a sample of patients which fit the criteria for which he is looking.

**Some pointers to existing PSEs.**

Recognizing that even a partial “wish list” such as the one described above might well be seen as either ignoring the successful work of active PSE developers or as too much “pie in the sky,” we want to turn now to providing at least some pointers to human centered features of existing PSEs. Not surprisingly there is no existing example of a single PSE which makes possible all of the features described in the “wish list” above. (In fact such a PSE may not even be possible, let alone practical.) Indeed, for
many computationlly complex problems the impressive accomplishment lies not in how well the software takes account of the problems of the human computer interaction, but in the fact that the system works well in solving an important problem in the first place.

There are, however, a variety of successful PSE projects to which to which we can point that provide illustrations of some of the ideas described in the “wish list.” In some cases these features represent partial instantiations of ideas described above. In other cases they represent mature examples which the PSE developers thought of first out of a concern for their users. For the convenience of the reader, all pointers will be to PSEs or projects described in a single issue of *IEEE Computational Science and Engineering*. A partial list of some human centered features to be found in these projects includes: allowing full user control of the starting point of automated processes; providing templates of abstract structures and of interaction summaries, incorporating features to provide memory context (e.g., hyperlinking, outlining), providing the ability to inspect intermediate results and the ability to add new algorithms to a database of algorithms, offering a high level specification language, and allowing choices among options; allowing specification of format and destination of files, and providing for email and web type flexibility; providing for cross-disciplinary problem solving and offering canned configurations for novices, allowing on demand result logging and flexible configuration of models, both within and between models.

**Conclusion**

This paper has begun exploration of some of the considerations associated with designing for human centered interaction with a scientific PSE. The basic goal has been to make the case that Human Computer Interaction is no longer just a supporting area of concern for PSE development. A concern with HCI is as central to successful PSE development as are such important components as Symbolic Computing, Numeric Computing, and Visualization. As the work on PSEs matures from a necessary focus on efficient accurate solution of a particular class of problems to taking on the creation of a true computer based working environment for scientists or engineers, an understanding of interaction flow issues and of the cognitive science aspects of how target users work will become even more important.

It is important to create mental aids or assists for the researcher. These aids should support thought processes and reduce or eliminate cognitive load during the flow of interaction. Effectively the working environment needs to make possible the kind of mental work in which problem domain experts seem to engage. To this end it is important that there be clear, timely feedback about the consequences of actions so as to
facilitate error detection and correction. When working with tools, the researchers should be able to change values, to change relationships, to change terms in equations, or to change code, but should not have to do so unless motivated to do so. In addition, researchers should be able to browse, to play, to do “what if?” exploration, etc., with a variety of different problem representations which can interact with each other seamlessly.

There should be a heavy reliance on the ability to create multiple alternative representations (e.g., graphics, diagrams, or 3D representations) so as to represent information about space and time and to assist in visualization of abstract relationships. The researcher’s working environment should provide multiple paths through the information structure or problem domain space. The PSE should make it possible for the researcher to establish effective memory retrieval cues which aid in being able to remember the structure and contents of the working environment and problem space.

When navigating around in the PSE, or when accessing information and/or tools, the researcher must be in control rather than the computer. That is, any computer based tools used by the researcher should provide an easy way to move from one place to another, to background or foreground an activity, to suspend an activity, to resume an activity, and to quit when done. Similarly, the researcher’s working environment should also make it possible to go outside the current problem domain to explore analogous problem domains for useful ideas and insights which can occur in an environment with a different set of constraints.

The ideas described above imply that a high priority should be placed upon minimizing or eliminating barriers to successful problem solving. Since it is not possible to predict the exact path of problem solution it may be that the best that can be done is to get out of the way and stay out of the way. As a consequence, the environment needs to support and allow for the way researchers think about a particular problem domain. It also needs to support and allow for the ways that thinking will change as the researcher interacts with the problem domain space.
Acknowledgments

Work on this paper was supported in part by Grant CCR-9527130, "Integration of Symbolic Computing with Frameworks of Classes and Problem-Solving Archetypes." The comments of several anonymous reviewers helped us to clarify both our thinking and our language, and the guidance and patience of the editors was invaluable.

References


