User-Centered Graphical User Interface Design for GIS

April 1991

David P. Lanter

NCGIA
Department of Geography
University of California at Santa Barbara
Santa Barbara, CA 93106

Rupert Essinger

Applied Interactive Technology
16 Bell Street
Henley on Thames, Oxforshire RG9 2BG
England

National Center for Geographic Information and Analysis

Report 91-6

User-Centered Graphical User Interface Design for GIS

David P. Lanter
Department of Geography
University of California
Santa Barbara, California 93106
USA

Rupert Essinger
Applied Interactive Technology
16 Bell Street
Henley on Thames, Oxforshire RG9 2BG
England

The quality of the user interface has a great bearing on the utility of a geographic information system. The user interface, however, has not been a strong point of GIS (Cowen and Love 1988; Egenhofer and Frank 1988). To increase the efficiency of GIS the user interface must provide a simple conceptual model of what is happening to the database (Collins et al. 1983). It must be easy to learn, appear natural, and independent of implementation complexities such as data structures and algorithms (Eggenhoffer and Frank 1988). In order to do this, the user interface of the GIS should show itself to its user as a system, and not as various collections of data (Driver and Liles 1983).

This paper discusses how traditional user interface design focuses on how to best represent the software functionality rather than on how to meet the expectations of the user. User-Centered Design is offered as an alternative that focuses on the two-way mapping between system functionality and the user's conceptual model of the system. Graphical user interface techniques are discussed as ways of creating the necessary two way mapping and facilitating the usability of GIS systems.

User-Centered Design

The problem of making GIS useful to people is a user interface design problem rather than an engineering problem. Engineering typically begins by eliminating the subjective factors, but it is exactly the subjective factors that are critical to the usability of information systems. To create a successful user interface the designer must understand how people think and work. The designer must realize that users do not actually use algorithms, data structures, networks, functions or subroutines, even though as technical professionals this is typically the domain in which they work. Instead, system users push buttons, choose options, type things in, make selections from menus, give commands and manipulate controls. In other words, user interfaces are **illusions** that hide the underlying architecture of the technology prominent in the programmer's view and repackage it as something understandable and usable by analysts and decision makers. Some of the most successful user interfaces are complete illusions outwardly bearing no resemblance to the data processing happening inside the machine. Of course, these illusions require their own program code. It is not unusual for more than 60% of the code in a complex software system to be dedicated purely to the user interface. This stands in sharp contrast to the 35% dedicated to the user interface in early GISs (Nicholson 1983).

Why do users need these illusions? One of the biggest problems for end-users is that the things computers let users do are **abstract**. Even the terminology they employ: files, directories, records, databases, logging on and off, function keys, scroll-bars, control-alt key combinations, etc. is often unfamiliar and grounded in abstraction. In everyday controls such as light switches, door handles and shower fittings, there is usually a fairly obvious correspondence between what a control looks and feels like physically and what it actually does. If one is not sure which light switch on a panel controls which light, one can generally try them until the right one if found. John Carroll (1984) at IBM calls this **learning through exploration** and the ability to perform this significantly increases the learnability of a system. In computer software this element of physicality is lost. Almost everything relating to the internal workings of the computer is hidden and largely divorced from anything the user understands.

To make systems truly usable software illusions are built on top of the underlying functionality. These illusions make abstract things appear concrete and give users the impression that they are controlling real objects. For example, the three-dimensional buttons and animated pulldown menus of Graphical User Interfaces (GUIs) have utility far extends far beyond mere cosmetic drapery. They serve to restore an element of physicality and concreteness that promotes understanding and a feeling of being in control of computer software.

User Models

Users have **mental models** about the tasks they accomplish with a system, and the way the system lets them accomplish those tasks. These models are defined by the user's prior experience, existing knowledge, and preconceptions about tasks. For both a

task and the way to accomplish the task to make sense, they must correspond to existing knowledge the user already has. In addition, for new material to be understood, interpreted and integrated with this existing knowledge it must be introduced in a clear fashion (Norman 1988). When new material is introduced poorly, users attempting to integrate it with their existing knowledge invent their own explanations. For example, my students can often be found blaming themselves when the GIS packages they are working with crash as the result of software bugs. On other occasions, they can be found blaming the GIS for problems they inadvertently created. This behavior is typical of many computer users. They tend to make up their minds about the function of systems based on concocted explanations for problems they do not understand. One of the jobs of the user interface designer is to make sure the conclusions they reach are the correct ones.

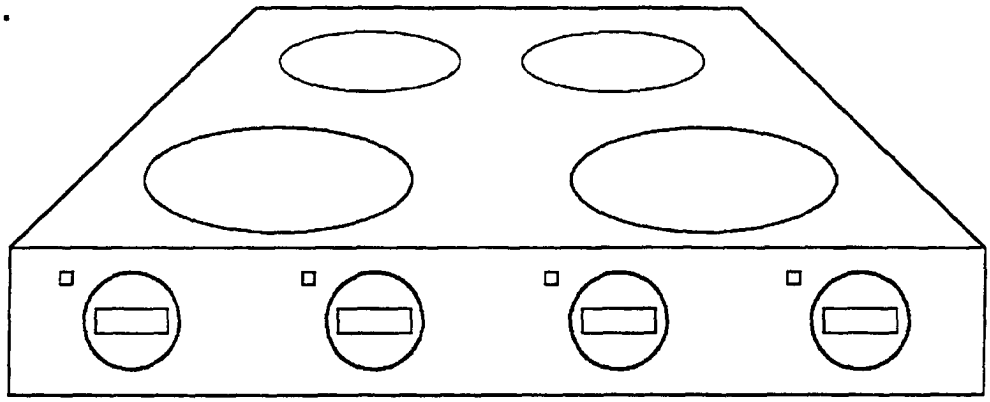
Each user possesses a conceptual model of the software system he or she interacts with. This model is shaped and influenced by both internal and external factors. The internal factors relate to the user's goals, expectations, intentions, experiences, pre-conceptions, past knowledge and explanations. The external factors relate to the system's interface as it initially presents itself to the user and as it reveals other dimensions of itself as it is used. System interfaces designed to match the user's existing conceptual models often require very little external documentation. Those designed without attention to the user typically require volumes of documentation, and days of training to change the users conceptual models to fit the system. Support is a euphemism for helping users adapt to difficult aspects of a system. All of these, thickness of user manuals (in inches), training time (in days), and support (in numbers of telephone calls) are indications of a failure of the system interface to map onto the user's conceptual model.

Natural or intuitive user interface **mappings** are obvious. For example, to move something up, you move the control up. The arrangement of controls matches the way the things being controlled are arranged. These natural mappings usually involve appropriate and immediate confirming feedback. For example, when driving a car the car goes in the direction the steering wheel turns to. This mapping, however, is reversed in steering a small boat with a filler or outboard motor. Hence, controlling a small boat is often difficult for the novice. The same is true for the turn indictor signal control of a car. Moving the turn indicator up or down to signal a right or a left turn is initially an unnatural mapping until it is conceptualized as a natural extension of the directional turn of the left hand side of the steering wheel. With a computer mouse, the feedback is more instantaneous and easy to understand. New users typically do not notice the partly unnatural mapping involved. That is, the mouse is moved forward on the desk to move the cursor up on the screen. This somewhat unnatural mapping is relatively easy for users to overcome, especially as left and right movement work as expected (move the mouse left to move the cursor left).

Water faucets are another classic problem of mapping. The user wants to set the volume of water and also its temperature. Water, however, normally comes in two pipes. Even mixer taps normally have two faucets - one for hot and one for cold. Most mixer taps merely do what is going to happen anyway (mix hot and cold water). The problem is that the two things that faucets make easiest to control - the volume of hot water and the volume of cold water - are not the two things users want to control. What the user would prefer is to turn on water at the desired temperature and pressure. They often have to set, adjust, and make subtle resettings and re-adjustings of their bath or shower water temperature.

In the stove top illustrated in Figure la, many people have great difficulty knowing which knob controls which burner ring. There is a real problem of mapping here. The user's conceptual model includes four rings, yet while the user can see four controls the mapping from them to the burners is unexpressed. Notice how on a stove, like in most computer systems, these usability problems are solved by adding documentation or training. In the case of the stove, the documentation is often a little picture or 'icon' added next to each control. The little icons do not solve the usability problem because they must be read everytime the stove is used. Usability here has been bolted onto the system. In the redesign illustrated in Figure 1b there is a natural mapping between the positioning of the burners and the positioning of the knobs that control them such that there can be no confusion as to which knob controls which burner. The user interface reflects the user model and so the need for documentation and training disappear. Notice how this arrangement does not require little icons beside each knob to indicate which ring it controls. Usability here has been **built into** the system.

1a.



1b.

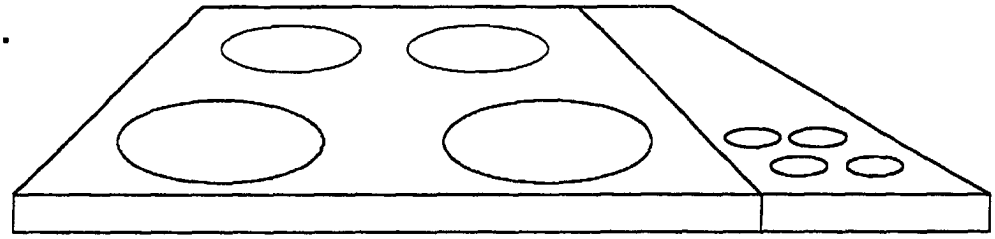


Figure 1. Natural mappings in interface design. 1a) It is difficult to know which knob controls which burner ring in this design. 1b) It is easy to map between the position of the knobs and the burners in this design.

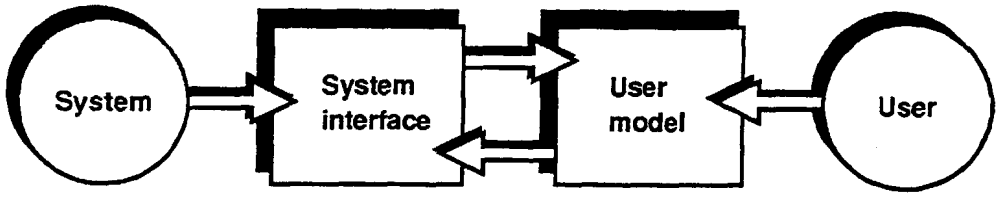
User-Centered System Design

In computing. unnatural and arbitrary mappings that retard usability are legion. It is a great challenge to build natural, meaningful mappings, especially in complex information systems. What lies between a system and its user is shown conceptually in the diagram of Figure 2a. There is a two way mapping between the system interface and the user model. The system interface designer, therefore, aims to match the user model. This kind of design has a particularly strong impact on the ease of learning a system. When users find their expectations and assumptions validated in the way a system works it follows that there is less need for users to expend mental resources on learning entirely new concepts and ways of doing things. When a system does not match the user model, additional learning is required by the user. Ibis places an increased memory load on the user as they must remember new and unfamiliar objects, processes, and relationships between the two.

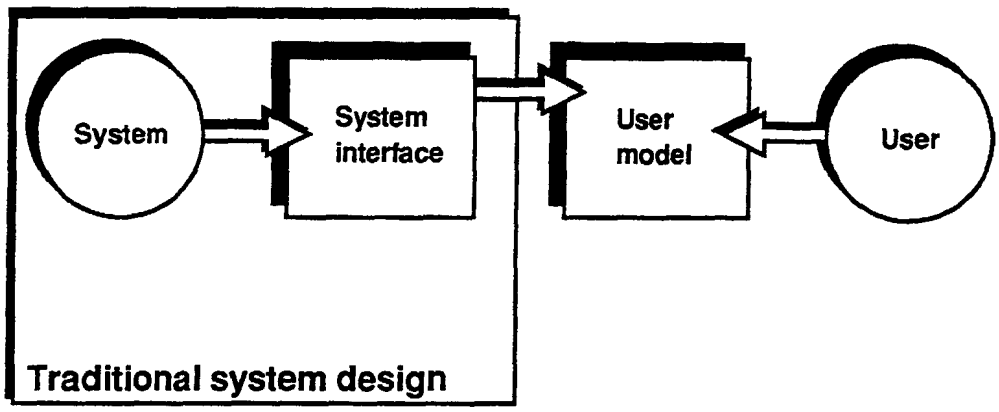
Traditional user interface design takes place in the left side of Figure 2b. That is, most user interfaces are currently designed to solve the problem of how to best represent the system function in the system interface. In the stove controls, for example, the first part of the interface was designed correctly in that it was decided that four controls was the best way to control four burners. (This portion of the design could have been botched by use of one big knob that is controlled with an additional SHIFT' key that is held down to reach the first ring, a CONTROL key to reach the second ring, and so on.) But another aspect of mapping the interface on to the user model was missed. That is, the relative positioning of the burners and controls so that the user could easily map which control related to which burner.

This does not mean that the user interface must be a slave to the user's model. The designer should aim at moulding and directing the users model in order that new, more informative and efficient ways of doing things can be introduced in understandable ways.

2a.



2b.



2c.

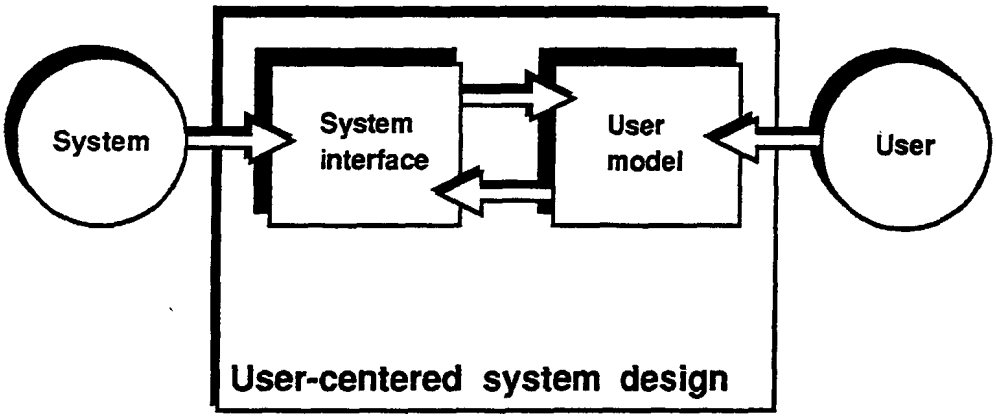


Figure 2. Focus of user interface design. 2a). The two-way mapping between system interface and the user model. 2b) Traditional design focuses on representing system functionality in the user interface. 2c) User centered design focuses on how to map the system interface to the user's existing model and how to shape the user's model.

For example, the analog clock or wrist watch is an unnatural mapping of time. Its two, or three, hands of different lengths pass around the same circular face. Children often have difficulty learning to read the analog clock. A task that almost always has to be accomplished by rote memorization. Once mastered, however, the watch provides additional information about the relative passage of time. Digital clocks, on the other hand, lack this additional information. In addition, the analog watch display provide the standard metaphors of clockwise and counter-clockwise used often in directions for turning all sorts of familiar circular controls. Responding to the user's model in a software design is therefore a two way process. The focus of user-centered interface design should concentrate on:

1) how to map the system interface to the user's existing model, and
2) how to shape and influence the user's model while they interact with the system.

Information, tasks and methods provided by a system interface will be more likely to be understood and learned if they fall into existing conceptual frameworks that the user has. If these are not readily available, conceptual frameworks will have to be invented that clearly communicate the functioning of the system to the user. The domain of user interface design centers on the creation of specifications that express system capabilities throughout the system interface in ways that match, adapt, or create conceptual models in the user model (Figure 2c).

Graphical User Interfaces

Of the many subjective factors that come to play in user-computer interaction a critical one is the burden the system places on the consciousness of the user. Users interacting with graphics systems are best served when the system virtually disappears from their consciousness leaving only their work and its ramifications to claim their attention (Foley J. D., Wallace V. L. and Chan P. 1984). If the user model can be expressed and dynamically maintained for the user on-screen, the user obtains psychological closure. That is, the user no longer has to store this mental image in short term, active memory (because the screen holds it for him) and can instead use that short term mental resource for more immediate tasks. These issues are addressed by research into how to deliver a conceptual model of the system to the user on-screen in a way that maximizes usability of the system and productivity of the user.

One way of doing this is with graphic metaphors that provide users with a familiar conceptual models. The first graphics-based user interface was implemented on the Xerox Star workstation announced by Xerox Corporation in April 1981. The Star's user interface designers established explicit guidelines which raised 'simplicity' as a standard within Graphical User Interfaces (GUIs) for office information systems. Simplicity was gained by providing users with controls they are already familiar with to make the systems easier to use and learn (Malone 1984). The design of the Star's interface was based on the metaphor of a physical office (Smith et al. 1982). The design resulted in creation of the 'electronic desktop'.

The Star's desktop illusion was complete with standard office equipment including printers, file drawers, folders, and documents, available as small icons (Smith, Irby and Kimball 1982). The user interacted with the icons of the Star system by means of a mouse and a series of function keys on the keyboard. The user would select an icon by pointing to it with a mouse controlled arrow graphic and pressing a button on the mouse. This would highlight the selected icon. The user would press a function button on the keyboard to apply a command to the object represented by the highlighted icon. For example, once selected, a text file associated with a highlighted document icon would be deleted when DELETE function button was pressed. This interface design offered direct manipulation,

that is - 'seeing and pointing', as an alternative to traditional interfaces that depend on remembering a command and typing it in. This design led the way for user interfaces offered in subsequent systems including the Apple Macintosh, Microsoft's Windows, Hewlett Packard's New Wave, Microsoft and IBM's OS/2 Presentation Manager, Sun Microsystem's Open Look, and Open Software Foundation's OSF/Motif.

The use of icons by the Star's Graphical User Interface designers was an innovative application of pictorial expressions to establish a communication link between machines and humans. This technique is an adaptation of a form of inter-human communication dating back to Chinese 'Jiaguwen' pictographics inscribed on bone and tortoise shells during the Shang and Yin Dynasties spanning the sixteenth through eleventh centuries B.C.. The use of visual cues in desktop icons and a mouse to select them was a major contribution to easing the use of computer systems. These cues, however, need to be both easily recognizable and understandable to the system's users. Vivid representation of objects and concepts will result in icons that are easy to learn and hard to forget (Shu 1988). The task of creating icons must be well thought out, or the results will be ill-conceived portrayals expressed in patterns that are undetectable to tile user's eye.

Symbolic clues can be combined with locational clues within graphical user interfaces to help users find information or data stored in traditional databases. The result is a spatial data management system (Fields and Negroponte 1977). In spatial data management, location is used as a clue to help users find desired information. This complements the use of icons to represent

symbolic information with a spatial framework for laying them out on a screen. This technique has application in user interfaces for nonprogrammers, programmers, and database designers.

GUI's for Non-Programmers

Non-programmers can directly manipulate visual representations of their data to retrieve it from a database. These graphical representations facilitate browsing for needed information without having to use formal query languages or specify the location of the data within the database (Donelson 1978; Herot 1980; Friedell, Barnett, and Kramlich 1982; Friedell, 1984; McDonald 1984). Wu et al. (1989) introduce a visual query language for GIS. Their system provides a graphical way the user can browse a GIS database. The mouse is used to select layers and processing is determined by pointing to query commands within the interface. These graphical depictions of layers stored in a GIS are useful alternatives to directory listings of file names. In addition to querying, GUI systems are useful for laying out the logical structure of the database. In such systems, database designers manipulate graphical representations of entities, - relationships, and attributes to create and integrate conceptual models of database views (Wong and Kuo 1982; Reiner et al. 1984; King and Melville 1984; Goldman et al. 1985; Bryce and Hull 1986; Abiteboul and Hull 1986;).

Contemporary researchers are designing GUIs to help users learn how to use GIS technology. Such systems rely on a graphical interface to build friendlier systems (Egenhofer and Frank 1988; Raper and Green 1989; Wu et al. 1989). Cowen and Love (1988) suggest this approach insulates the user from complexities of the GIS. They illustrate this by providing an interface that creates linkages between maps. In this system the user interacts with intelligent maps to access relevant spatial or attribute information. Cowen and Love found that icons can be used to provide relational access to GIS information. Raper and Green's use of a GUI provides students of GIS with an intelligent tutor (Raper and Green 1989). The resulting system stresses connections between subjects so students can dynamically access preconfigured cross referenced links between instructional material. This frees the student from a sequential path through the educational material and permits the curious to follow intellectual tangents in a self designed learning session.

The design of the GUI of the Forest Management Decision Support System (FMDSS) incorporates a choropleth map to provide novice users with intuitive views of information typically obscured within the rows and columns of computer spread sheets and standard tabular reports. FM-DSS provides foresters formulating alternative timber harvest strategies to visualize associated environmental. risks. It takes tabular output from the USDA Forest Service's standard FORPLAN forest harvest schedualing model and cartographic data from a vector-based GIS. The interactive map, pull-down menus, and dialog boxes of FM-DSS's GUI demonstrate the use of graphic and locational cues to provide users with meaningful views of data unavailable in standard tabular reports.

The forest map consists of a number of administrative units ('compartments'). Each compartment is shaded to represent the degree to which the FORPLAN designated harvest threatens the viability of the remaining land (Figure 3a). The user can query for a compartment's attributes by pointing and clicking the mouse on the desired compartment within the map. The system responds with a dialog box containing all the attributes for the selected compartment (Figure 3b). Maps of environmental risks resulting from FORPLAN designated harvests typically reveal that some compartments are undercut and others severely undercut. FM-DSS users can select alternative approaches to harvesting timber from a pull down menu. The results are immediately displayed in an updated map. For example, users can apply and view the result of a optimization algorithm that balances the effects of harvesting across all compartments while not violating any environmental constraint (Figure 4a). Another algorithm choice results in a balanced timber harvest schedule that yields the same amount of timber as FORPLAN (Figure 4b).

3a.

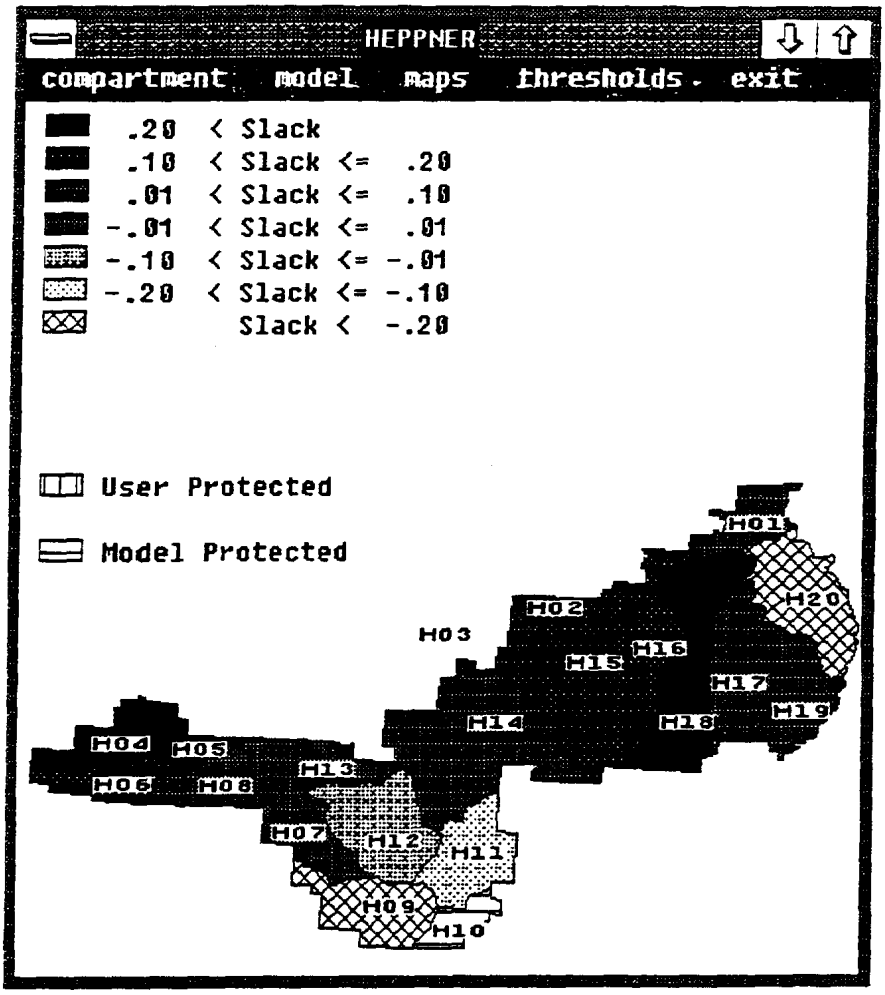
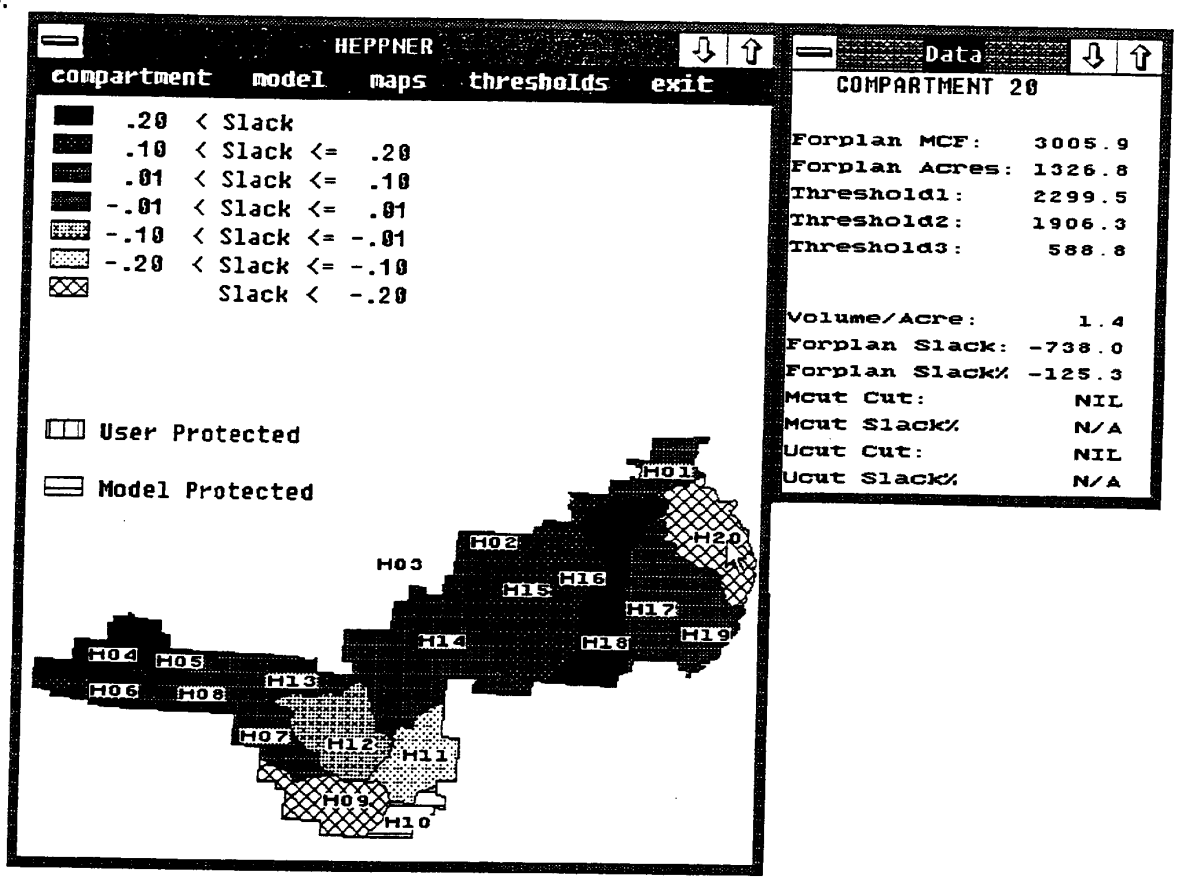


Figure 3. The GUI of FM-DSS incorporates an interactive choropleth map. 3a) Each compartment is shaded to represent the degree to which a FORPLAN designated harvest threatens the viability of remaining land. 3b) The user queries for data attributes by clicking the mouse on the desired compartment.

3b.


HEPPNER
compartment
model
maps
thresholds
exit
.20 < Slack
.10 < Slack <= .20
.01 < Slack <= .10
-.01 < Slack <= .01
-.10 < Slack <= -.01
-.20 < Slack <= -.10
Slack < -.20
User Protected
Model Protected
H01
H02
H03
H04
H05
H06
H07
H08
H09
H10
H11
H12
H13
H14
H15
H16
H17
H18
H19
H20
Data
COMPARTMENT 20
Forplan MCF: 3005.9
Forplan Acres: 1326.8
Threshold1: 2299.5
Threshold2: 1906.3
Threshold3: 588.8
Volume/Acre: 1.4
Forplan Slack: -738.0
Forplan Slack% -125.3
Mcut Cut: NIL
Mcut Slack% N/A
Ucut Cut: NIL
Ucut Slack% N/A

4a.

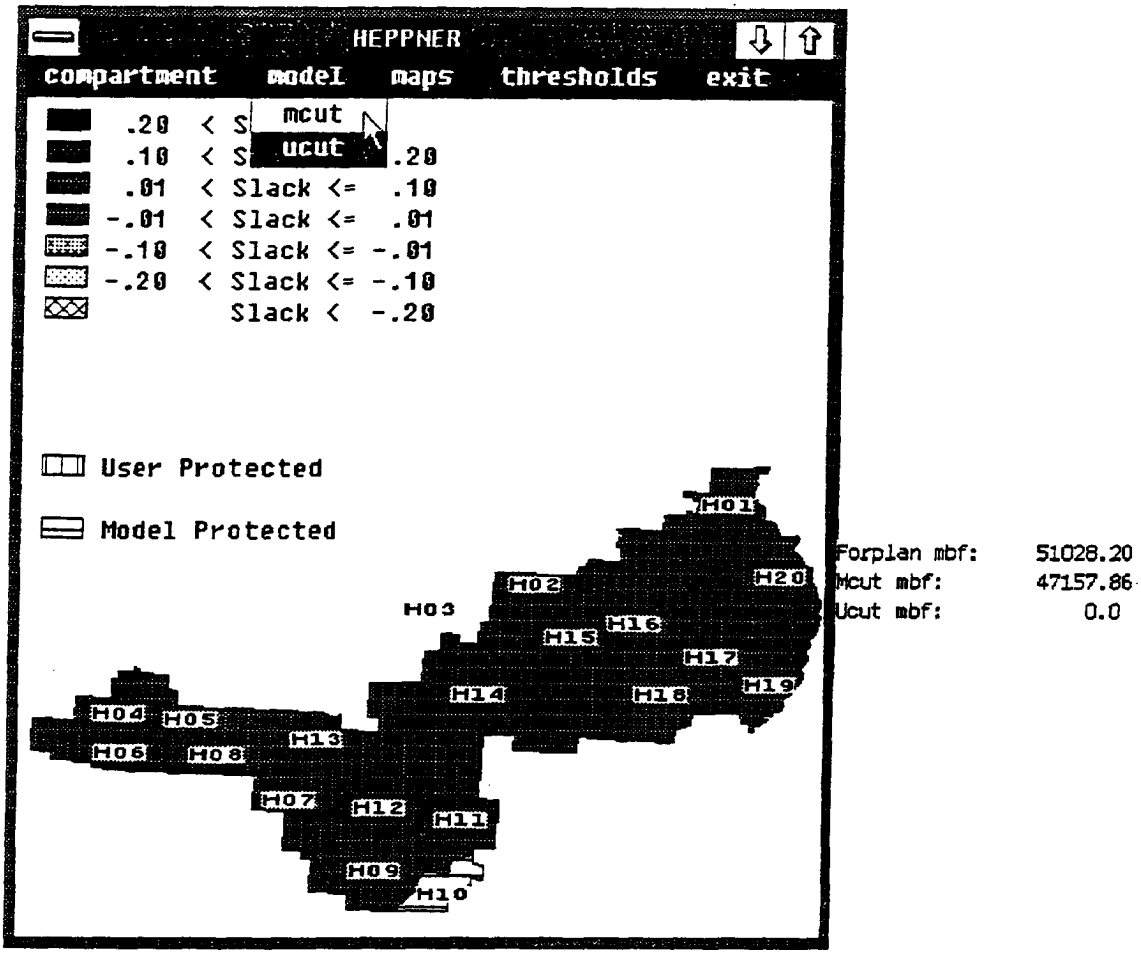
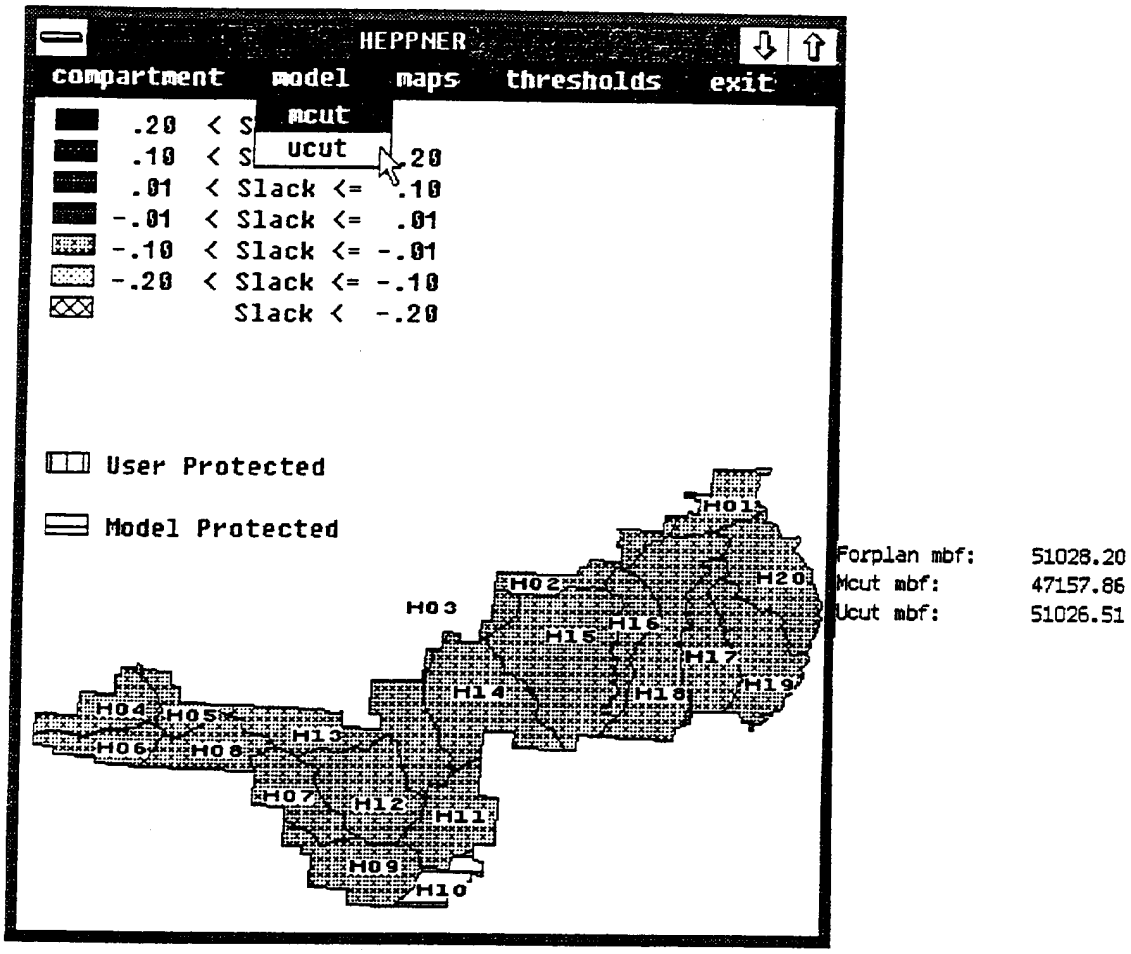


Figure 4. Users can select optimization algorithms from pull-down menus and can immediately view the results in the cartographic GUI.
4a) Results of applying an algorithm that balances the effects of harvesting while not violating any environmental constraint.
4b) Results of applying an algorithm that balances the efffects of harvesting while delivering the same amount of timber as FORPLAN.

4b.


HEPPNER
compartment
model
maps
thresholds
exit
mcut
ucut
.20 < S
.10 < S
.20
.01 < Slack <= .10
-.01 < Slack <= .01
-.10 < Slack <= -.01
-.20 < Slack <= -.10
Slack < -.20
User Protected
Model Protected
H01
H02
H03
H04
H05
H06
H07
H08
H09
H10
H11
H12
H13
H14
H15
H16
H17
H18
H19
H20
Forplan mbf: 51028.20
Mcut mbf: 47157.86
Ucut mbf: 51026.51

There are often times when compartments contain sensitive ecological zones that must be protected. In these cases users can protect them from harvesting by choosing the PROTECT option from a pull down menu and pointing to and clicking on them with the mouse. Protected compartments are removed from consideration by subsequent runs of the optimization algorithms discussed above (Figures 5a and 5b). Once protected, the user can select the UNPROTECT option from the pull down menu and re-designate protected compartments for harvesting in subsequent runs.

FM-DSS's cartographic interface is seen by users at National Forest District offices in Oregon and Washington States as a natural way to view and query a database of attributes concerning the forests they manage. The simplicity offered by the locational and symbolic clues within the map and the locational and symbolic clues within the pull-down menus and dialog boxes enables foresters to experiment with alternative timber harvest strategies that minimize environmental risks.

GUI's for Program Designers

GIS software and application program designers can benefit from graphical representations as well as non-programming users. Programmers can interact with GUI representations of their data structures to gain better understanding of the algorithmic processing of their data (Myers 1983; Boecker and Nieper 1985; Boecker, Fisher, and Nieper 1986). Such interfaces help both beginning and expert programmers to visualize what a program does and how it works. This supports code writing, testing, and debugging. GUI representations of the algorithms applied to these data structures is also of great use. Data flow diagrams are one way of communicating the flow of control through computer programs. These have been automated and animated within GUIs to display the inner workings of programs and are useful as both a software design tool and a documentation device (Nievergelt 1980; Clark and Robinson 1983; Brown and Sedgewick1984, 1985; Reiss 1985; Reiss, Golin, and Rubin 1986; Levien 1986; Hughs and Moshell 1986; London and Duisberg 1985).

The data flow diagram has been referred to in GIS as the 'Cartographic Model' (Tomlin and Berry 1979) and 'Graphic Map Algebra' (Kirby and Pazner 1990). Lanter (1989, 1991) implemented this approach in a system that maintains a self-updating data flow diagram as users update the GIS database. This provides users with a simple conceptual model of the geographic database that closely matches the user's model of GIS operation by concentrating attention on the organization of the database. Such a graphical view of the database is useful to users because it documents the sequencing of GIS operations, something that is often abstract and difficult to visualize. It simplifies GIS concepts by bringing out the structure of data flows within spatial analytic applications. This is an ideal source of feedback to the user, needed as a reference for upcoming data manipulations (Mark and Roussopoulos 1983).

A continuously updating data flow diagram helps users to both perform source assessment, and track where they are and where they've been in the course of their map deriving applications. This focus is an alternative to the traditional emphasis on the functionality of the GIS software itself. In addition to the input/output relationships expressed in the data flow diagram, users require lineage attributes pertaining to source materials, transformations applied to the sources materials, and usage of GIS derived data products (Lanter 1990). This information can be made accessible to the user in the user interface through icons representing thematic data layers stored within the GIS database. Icons are representations of interaction with iconic representations of spatial data layers presented as part of the GIS's graphical user interface.

5a.

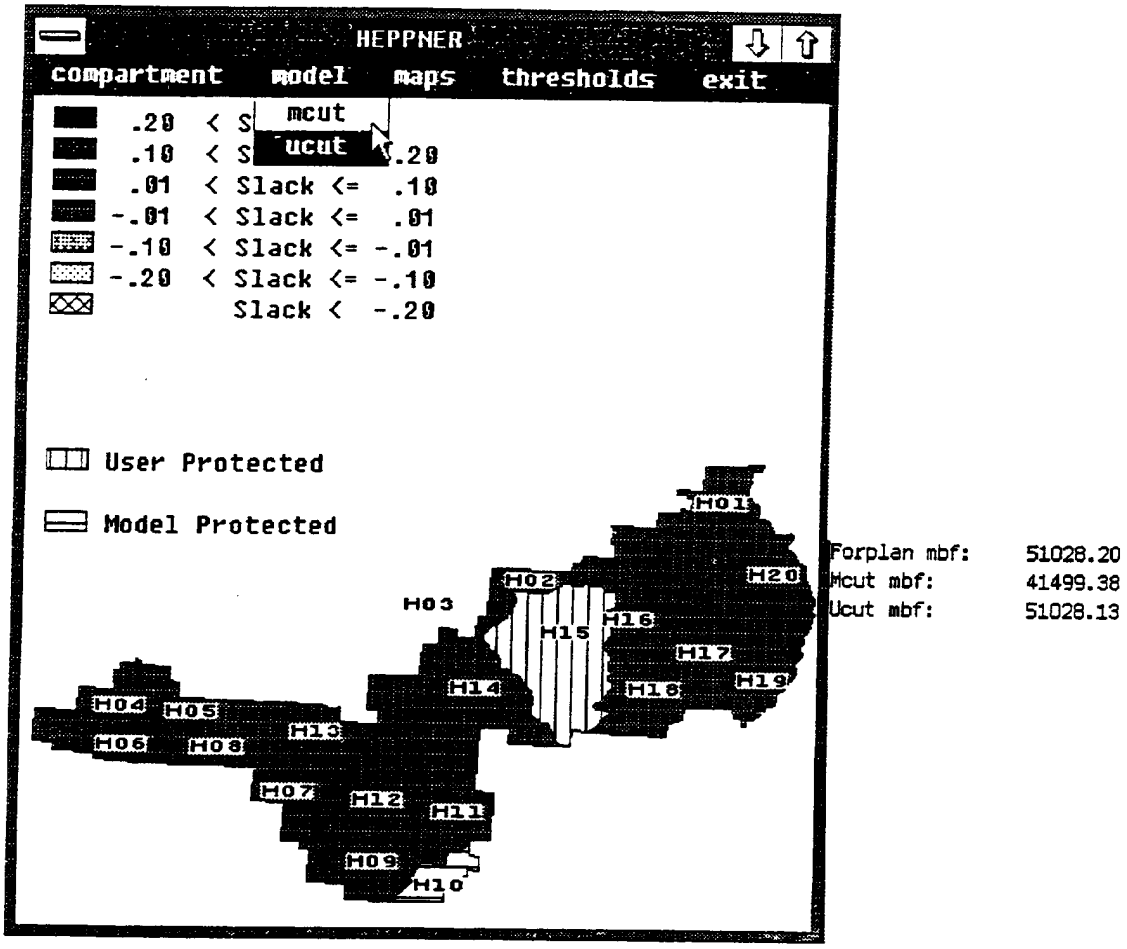
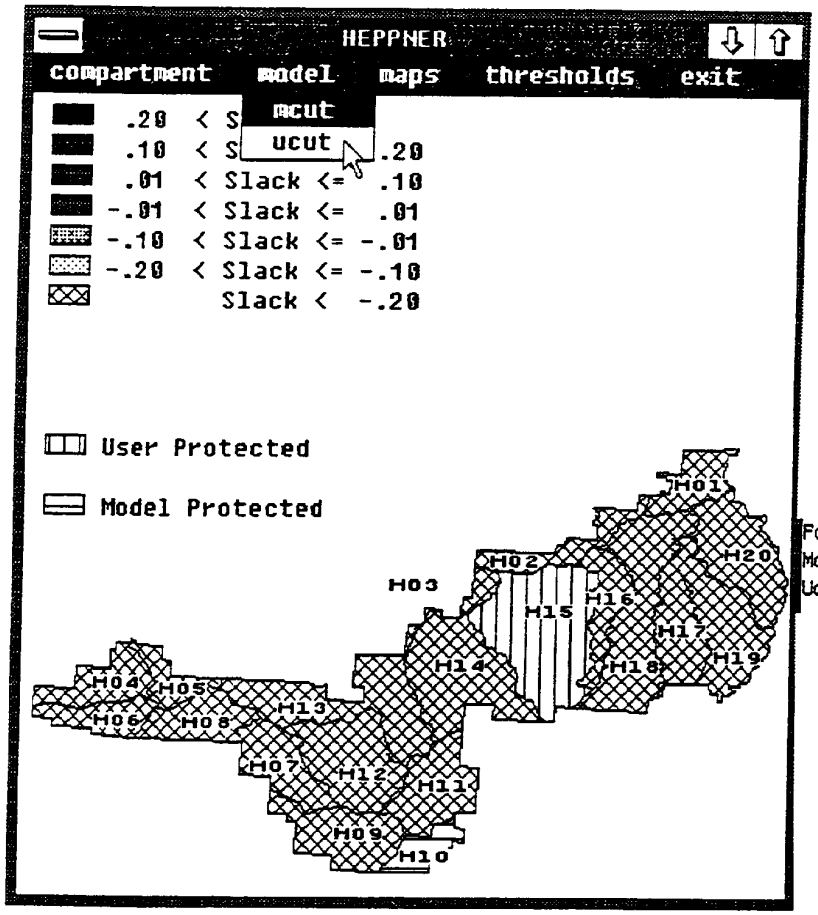


Figure 5. Users can remove a compartment from consideration harvesting by choosing the PROTECT option from the COMPARTMENT pull-down menu and clicking with the mouse. 5a) Results of applying the algorithm that balances the effects of harvesting while not violating environmental constraints is displayed. 5b) Results of applying an algorithm that balances effects while delivering the same amount of timber as FORPLAN is displayed.

5b.



Forplan mbf:	51028.20
Mcut mbf:	41499.38
Ucut mbf:	51028.13

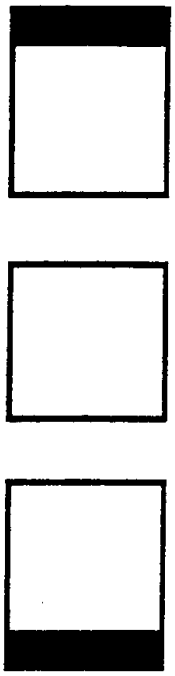
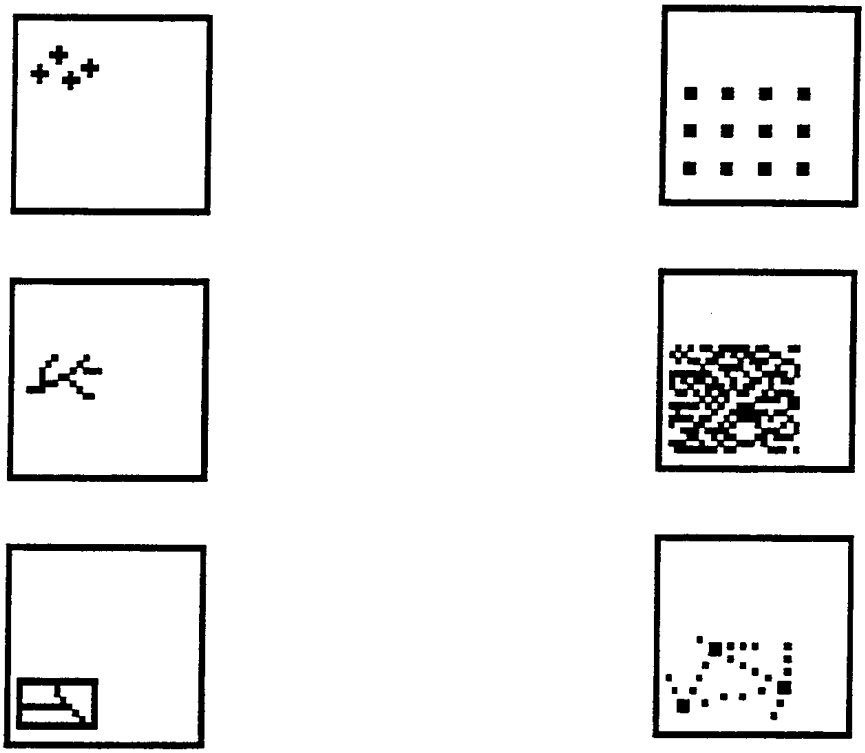


Figure 6. Source, intermediate, and product layer icons.

Source, intermediate, and product layer icons are illustrated in Figure 6. The source layer icon is a square icon that has a thick bar along the top. Source attributes concerning name, feature types, dates, responsible agency, scale, projection, and accuracy are associated in each source layer stored in the database. Intermediate layers are derived by the application of GIS transformations to source layers. They are documented with information concerning the nature of the transformation used in their derivation. There are two types of intermediate icons. One is a simple square, and the other is a square with a thick bar along the bottom. The former represents layers that are used as intermediate steps between source and product layers. The latter represents a derived layer output from the system for use in decision making. These product layers are associated with additional documentation concerning their use. Such information might include the users, role in decision making, release dates, and those responsible for their maintenance.

These icons can be enhanced with additional information. For example, symbols can be placed within layer icons to reveal the type of GIS data elements collected within a particular database layer. Such symbols can represent the three cartographic types: points, lines, areas (Figure 7a); or underlying data structure such as: Digital Elevation Model, raster image, or triangular irregular network (Figure 7b). In addition to graphic symbols, numeric values, such as the data accuracy indices propagated by Lanter and Veregin (1990), can be placed in the layer icons.

Arrows between layers can complement the pictographic and number information place within icons by documenting the topological lineage relationships between layers derived with spatial analytic operations applied to the GIS database. The result is a data flow diagram that places each icon within a GIS application's cartographic model. This can helps the user understand the nature of sources and derived maps stored in the database (Figure 8). The result is an informative interactive graphical user interface. For example, source, intermediate, and product attributes can be retrieved and viewed by pointing and clicking on the desired layer's icon (Figure 9).



7a. 7b.

Figure 7. Symbols in icons reveal the nature of data in layers stored in the GIS database. 7a) They can include cartographic types: point, line, and polygon, or 7b) Digital Elevation Model, raster image, or triangular irregular network.

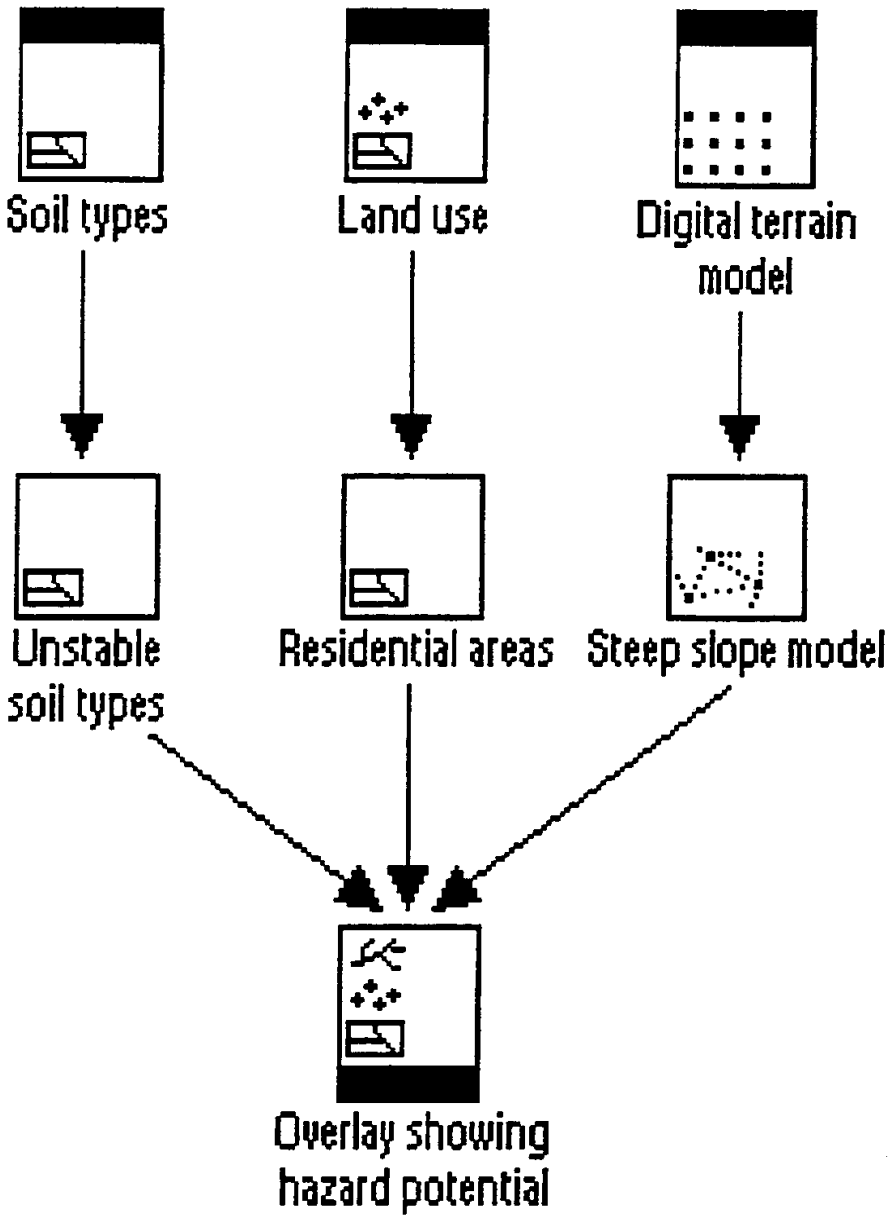
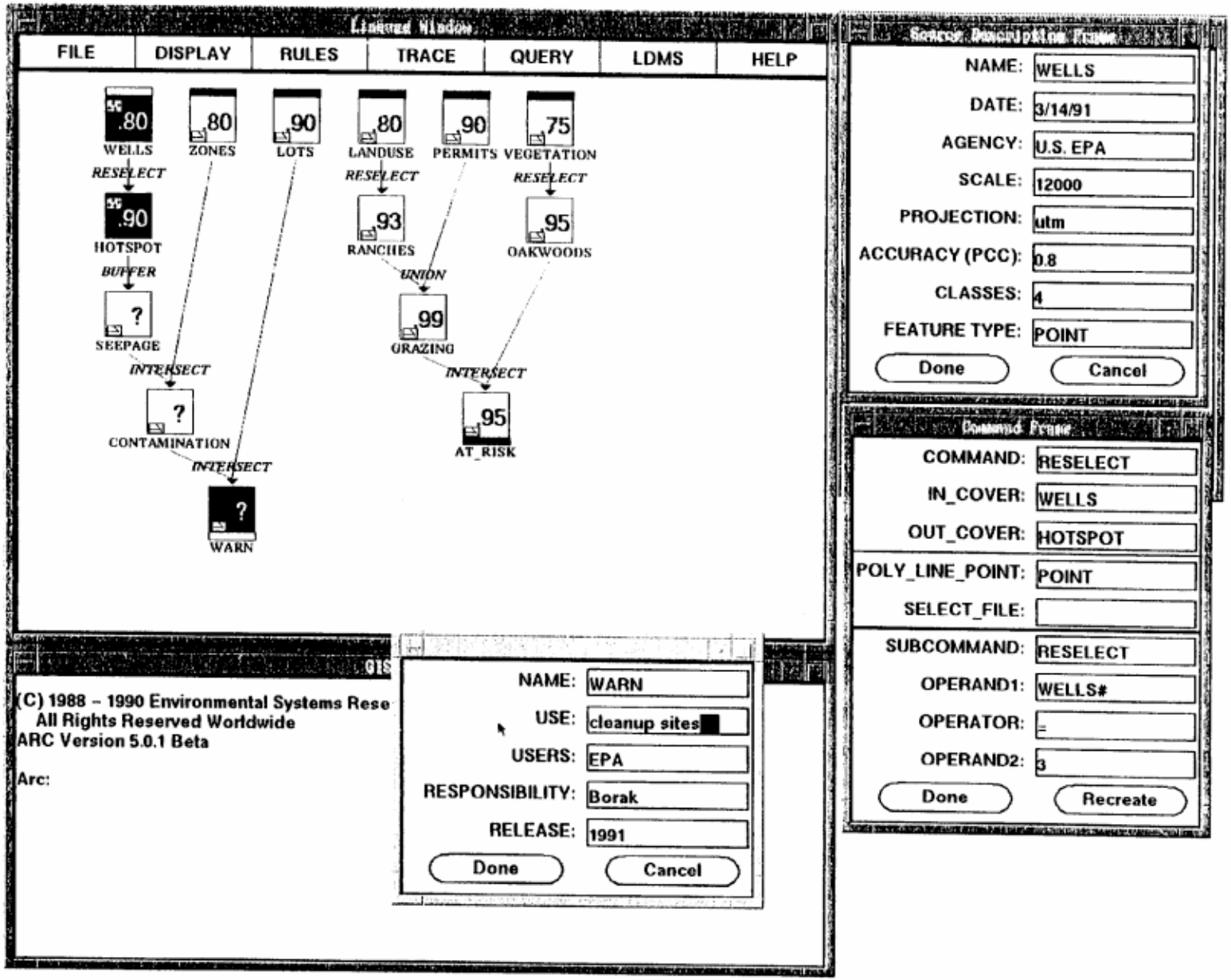


Figure 8. A network of connecting directed links place each icon within an application's cartographic model.

Figure 9. Interactive graphical user interface for GIS applications programmers. Users can click the mouse on layers and bring up source, intermediate, and product attributes. Layer icons include cartographic type pictograms and percent correctly classified (PCC) error indices where calculatable.



Conclusion

Traditional user interface design ignores the user, focusing instead on how to represent the software functionality in the system interface. This typically results in arbitrary design decisions that often fail the expectations of the user. The result is an over reliance on system documentation, training, and end-user support to help users adapt to poorly designed system interfaces. User-Centered Design is offered as an alternative that focuses on how to map the system functionality onto the user's existing conceptual models and how to influence these models while they interact with the system. This results in systems that are easy to learn, appear natural, and present themselves to the user as a complete system rather than as disjoint collections of data, data structures, and algorithms.

Graphical user interface techniques making use of graphic and symbolic clues can help make Geographic Information Systems that are easier to understand, learn, and use. Incorporating a map in a graphical user interface was shown to enable novice users querying a database, experiment with alternative model strategies, and visualize attributes resulting from both thematically. Integration iconic expressions in a dynamic dataflow diagram and presenting the result in a graphical user interface helps make the structure of a spatial analytic databases available to GIS application program designers. The authors are designing graphical user interfaces for the user to specify spatial analytic algorithms to the GIS, and for the GIS to provide feedback to users concerning the status of the executing algorithm and the data involved in the algorithm.

References

Abiteboul, S. and R. Hull 1986. "IFO: A Formal Semantic Database Model", Technical Report TR-84-304. University of Southern California, Computer Science Department.

Boecker, H., G. Fisher, and H. Nieper 1986. "The Enhancement of Understanding through Visual Representations", Proceedings of the CHT86 Conference, Human Factors in Computing Systems. pp. 44-50.

Boecker, H. and H. Nieper 1985. "Making the Invisible Visible: Tools for Exploratory Programming" Proceedings of the First Pan Pacific Computer Conference, Tle Australian Computer Society, Melbourne, Australia.

Brown, M.H. and R. Sedgewick 1984. "A System for Algorithm Animation", ACM Computer Graphics 18(3) pp. 177-186.

Brown, M.H. and R. Sedgewick 1985. "Techniques for Algorithm Animation", IEEE Software 2(l), pp.28-39.

Bryce, D., and R. Hull 1986. "SNAP: A Graphics-Based Schema Manager", Proceedings of the Second IEEE International Conference on Data Engineering. pp. 151-164.

Collins, S.H., G.H. Moon, and T.H. Leham 1983, "Advances in Geographic Information Systems", Proceedings of the Sixth International Symposiwn on Automated Cartography; Steering Committee of the Canadian National Committee for the Sixth International Symposium on Automated Cartography, Vol 1 pp. 324-334

Cowen, D.J., and S.R. Love 1988, "A Hypercard Based Workstation for a Distributed GIS Network", Proceedings of GISILIS '88, American Congress on Surveying and Mapping; Falls Church, VA, Vol. 1 pp.285-294

Carroll, J.M. 1984 "Minimalist design for active users.", Interact '84 First IFIP Conference on human-comRuter interaction, Shackel B. (ed). Elsevier Science.

Clark, B.E.J. and S.K. Robinson 1983. "A Graphically Interacting Program Monitor", The Computer Journal, 26(3), pp.235-238.

Donelson, W.C. 1978. "Spatial Management of Information", Proceedings of ACM SIGGRAPH 1978, pp. 203-209.

Driver, B. and W. Liles 1983, "A Communication Model for the Design of a ComputerAssisted Cartographic System ",Proceedings of the Fifth International Symposium: on Cartography and Computing, American Congress on Surveying and Mapping; Falls Church, VA, pp. 267-274

Egenhofer, M. and A. Frank 1988, "Designing Object-Oriented Query Languages for GIS: Human Interface Aspects", Proceedings of the Third International Symposiwn on Spatial Data Handling, International Geographical Union Commission on Geographical Data Sensing and Processing, Williamsville, NY, pp. 79-96

Fields, C. and N. Negroponte 1977. "Using New Clues to Find Data", Proceedings of the Third International Conference on Very Large Databases, pp. 15 6-15 8.

Foley J. D., Wallace V. L. and Chan P. 1984. "The Human Factors of Computer Graphics Interaction Techniques", IEEE Computer Graphics and Applications, 4(11) pp. 1348.

Friedell, M. 1984. "Automatic Synthesis of Graphical Object Descriptions", ACM Computer Graphics 18(3), pp. 53-62.

Friedell, M., J. Barnett, and D. Kramlich 1982. "Context-Sensitive Graphic Presentations of Information", ACM Computer Graphics 16(3). pp. 181-188.

Goldman, K. et al. 1985. "ISIS: Interface for a Semantic Information System", Proceedings of ACM SIGMOD International Conference on the Management of Data. pp. 328-342.

Herot, C.F. 1980. "Spatial Management of Data" ACM Transactions on Database Systems 5(4), pp. 493-514.

Hughs, C.D., and J.M. Moshell 1986. "Visible Pascal: A Graphics-Based Learning Environment" Proceedings of Computer Graphics 86, National Computer Graphics Association pp.401-41 1.

King, R. and S. Melville 1984. "The Semantics-Knowledgeable Interface", Proceedings of the Conference on Very Large Databases, pp. 30-37.

Kirby, K.C. and M.Pazner 1990. "Graphic Map Algebra", Proceedings of the Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland, Vol. 1 pp. 413-421.

Lanter, D.P. 1989. "Techniques and Method of Spatial Database Lineage Tracing", Dissertation, Department of Geography, University of South Carolina, Columbia, South Carolina.

Lanter, D.P. 1990. "Lineage in GIS: The Problem and a Solution", Technical Paper 90-6, Santa Barbara, CA:National Center for Geographic Information and Analysis.

Lanter, D.P. 199 1. "A Lineage-based Meta-Database for Geographic Information Systems", Cartography and Geographic Information Systems, (In Press).

Lanter, D.P. and H. Veregin 1990. "A Linage Meta-Database Program for Propagating Error in Geographic Information Systems", GISILIS '90 Proceedings, Vol. 1 pp. 144-153.

Levien, R. 1986. "Visual Programming", Byte 11(2).pp. 135-144.

London, R.L. and R.A. Duisberg 1985. "Animating Programs Using Smalltalk", IEEE Computer 18(8), pp.61-71.

Malone, T.W. 1984. "Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games", Human Factors in Computing Systems, editors I.C. Thomas and M.L Schneider, Ablex Publishing Corp.

Mark, L. and N.,,Roussopoulos 1983, "Integration of Data, Schema and Meta-Schema in the Context of Self-Documenting Data Models", En~tijy-Relationshjp Approach to Software Engineering, North Holland, New York

McDonald, N.H. 1984., "A Multi Media Approach to the User Interface", Human Factors and Interactive Computer Systems, edited by Y. Vissiliou, Ablex Publishing Co. pp. 105-116.

Myers, B.A. 1983. "INCENSE: A System for Displaying Data Structures", ACM Computer Graphics, 17(3), pp. 115-125.

NCDCDS 1988, "The Proposed Standard for Digital Cartographic Data", The American Cartographer, Volume 15, Number 1

Nicholson, R. L. 1983, "The Use of Large Scale Interactive Mapping Systems in a Decision Support Role", Proceedings of the SUth International Symposium on Automated Cartography;Steering Committee of the Canadian National Committee for the Sixth International Symposium on Automated Cartography, Vol. 1, pp. 271276

Nievergelt, J. 1980, "A Pragmatic Introduction to Courseware Design", IEEE Computer 13(9). pp. 7-21.

Norman, D. A. The Psychology of EveZ&ay Things. Basic Books Inc. (1988)

Raper, J.F. and N.P.A. Green 1989, "GIST: An Object-Oriented Approach to a Geographical Information System Tutor", Proceedings of the Ninth International Symposium on Computer-Assisted Cartography, American Congress on Surveying and Mapping; Falls Church, VA, pp. 610-619

Reiner, D. et al. 1984. "The Database Design and Evaluation Workbench (DDEW) Project at CCA", Bulletin of IEEE Technical Committee on Database Engineering, 7(4) pp. 10-15,

Reiss, S.P. 1985. "PECAN:Progam Development Systems That Support Multiple Views", IEEE Transactions on Software Engineering, 11 (3). pp.276-285.

Reiss, S.P., E.J. Golin, and R.V. Rubin 1986. "Prototyping Visual Languages with the GARDEN System" Proceedings of the IEEE Computer Society Workshop on Visual Languages. pp.81-90.

Shu, N. 1988. Visual Programming, New York:Van Nostrand Reinhold Company 315pp.

Smith,D.C., C. Irby and R. Kimball 1 982. "The Star User Interface: An Overview", Proceedings of the National Computer Conference. pp. 515-528.

Tomlin, C.D. and J.K. Berry 1979, "A Mathematical Structure for Cartographic Modeling in Environmental Analysis", Proceedings of the American Congress on Surveying and Mapping, American Congress on Surveying and Mapping, Falls Church, VA, pp. 269-283

Wong, H. and 1. Kuo 1982. "GUIDE: A Graphical User Interface for Database Exploration", Proceedings of the Conference on Very Large Databases. pp. 22-32.

Wu et al. 1989Wu, S.T. et al. 1989, "QPF: A Versatile Query Language for a KnowledgeBased Geographical Information System", Int. J. of Geographical Information Systems, Vol. 3 No. 1