Patterns for Human-Computer Interaction
Studies of Principle Aggregation and Pattern Naming

Ingjerd Skogseid Michael Spring

November 22, 1995

Abstract

This paper examines evidence to support the existence of patterns for human-computer interaction. It is based on work done as part of a master's thesis [14] in the department of Information Science at the University of Pittsburgh.

1 Introduction

Over the years, numerous authors have suggested principles for human-computer interaction. The principles have been intended to assist analysts, designers, and implementers of systems by providing guidance at one or another level of specificity. Principles have ranged from highly specific directives such as "Use no more than five to seven colors in a display"[6, page 171] to much more general guidelines such as "Strive for consistency"[13, page 72]. Despite the articulation of principles and the growth of a research infrastructure on which to base design decisions, interactive systems are still most frequently designed by "see if it feels right" techniques, rather than an engineering approach in which one starts with facts and bases the design on relevant scientifically grounded principles. We believe that there are several reasons why principles are not used more extensively and explicitly in the analysis, design, and implementation process. These include the following:

1. The number of principles and guidelines are so large that it is to hard for the analysts, designers and implementors to know what principle to use in a situation.

2. The principles are generally for designers, or analysts, or programmers. There are few principles that work across team specializations – that help the analyst know what to look for, the designer to know what to envisage, and the programmer to know what to code.

3. Many of the guidelines are so specific as to lose applicability when technology changes. For example, color principles that made good sense for character based displays are less relevant to high resolution graphics displays, at least as initially stated.

This paper looks to the work of Christopher Alexander, an architect, for cues as to how to specify principles for the design of interactive systems Alexander [1, 2] has suggested an approach in architecture for developing urban communities and buildings based on "patterns". This paper explores a similar process for the development of patterns related to human-computer interaction. We suggest that patterns will provide analysts, designers, and programmers with a set of well-defined principles that imply rules and procedures that can serve as a model upon which to merge the art and science of user interface design.

2 Toward a definition of a pattern language

The term 'pattern' is used frequently in the literature without a precise definition, or at best with a definition stipulated for the particular situation. We propose a definition below consistent with the history and usage of the term and based on the necessary characteristics of patterns in the area of human computer interaction.

The dictionary defines a pattern as "a model or design or instruction according to which something is to be made." Alexander defines a pattern as follows:

> "... the elements of this language are entities called patterns. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [1, p. x].

Alexander[2, p267)] states that a pattern must have a name, and must be drawable.

Love[11, p 43] views patterns as "design rules" that allow for successful design. To be successful the rules must not constrain the design unnecessarily or enforce unneeded uniformity. The object-oriented design and development community have embraced the notion of patterns. Object-oriented design and development technologies are based on the notion of reusable patterns of code[7, 10, 11, 19]. Coad[7, pp 152-153] sees patterns as the lowest level elements, classes and objects, and the relationships between them. Johnson states "patterns are problem oriented, not solution oriented. Each pattern describes how to solve a small part of the large design problem. Sometimes the solution is to make a new subclass, sometimes it is to parameterize an object of an existing class and sometimes it requires connecting several objects together."[10, p 64] Yourdon points out that the "OOPL community assumes that a pattern DOES exist and the job is merely to find it," and that in the traditional software analysis, design and implementation community they invent a new solution every time a problem does not fit an old, familiar 'pattern.'[19, p 18]

The various uses suggest a hierarchy that we have found useful in our thinking. There seem to be four distinct levels at which one might use the term "pattern":

1. components, conventions and interfaces
2. styles, combinations and rules
3. metaphors and models
4. approaches and schools

The first level, components and interfaces works well for software objects, widgets, and API's. It would be our contention that this is as much a pattern as outlets, pipe sizes, and doorknobs are patterns for architects. While they are no doubt reusable and consistent across applications, they are not the central examples of patterns. The second level may contain patterns, although many styles and rules are to specific for the definition of patterns we would propose. Thus, rules like "the file menu will contain open, close, save, and save as options connected to file dialog boxes", are not patterns. Stylistic conventions at this level are like the rules of home design that specify a kitchen gets a 220 volt service to a grounded outlet or that a bathroom gets a fault tolerant circuit. On the other hand, styles, combinations and rules may specify higher level relations and these would be patterns. For example, "the order of menu information should be related to ordering of action in the application" may well be a style or rule that would be classed as as a pattern. For the most part, principles related to metaphors and models are patterns. For example, under the direct manipulation model, principles like "make things visible and concrete rather than invisible and abstract"[15, 9], begin to establish patterns to guide the analysis and design process. As these metaphors and models become all encompassing,

we would suggest that they subsume sets of patterns under a common name. This would provide a parallel to schools in the architectural world such as modernism or bahaus. Thus we would suggest that "ubiquitous computing" or "virtual reality" suggest overall approaches to the interface that validate or invalidate the use of selected patterns. Thus, while affordances and mappings are important patterns for direct manipulation and virtual reality, they are less important for ubiquitous computing where transparency and immediacy become more critical.

In line with these views and definitions of a pattern we suggest the following operational definition of a pattern:

> A pattern is an abstraction that may be instantiated. It is described by a simple phrase and relates to an observable or discernible aspect of the interface or interaction. Operationally, a pattern:
>
> 1. has a consistent or corresponding meaning for by users, analysts, designers, and programmers,
> 2. helps in resolving design conflicts by improving communications across the phases of system development,
> 3. is relatively immune to changes in technology, and
> 4. may be applied in multiple situations at both specific and general levels

3 Methodology

Alexander[1, 2], in developing his architectural patterns visited "places that worked" to gather the characteristics of these places for later analysis. While it would be possible to examine interactive systems in this same way, it was our opinion that the literature already contains patterns and nascent patterns in the form of the many principles and guidelines that have been articulated. These principles and guidelines endeavor to abstract the characteristics of "interfaces that work." Design principles for interactive systems abound in the literature. [1] It was our initial belief that many of the operational principles in the literature would be more concrete and particular than the patterns. Similarly, we felt that the more conceptual principles might be very close to patterns, subsuming other principles. We sought patterns in line with what Thimbleby[18, p 202] has referred to as "generative user engineering principles." That is, a pattern should be easy to instantiate in a given situations and it should provide specific guidance about what should be done. Our contention was that a pattern such as "provide for ordered information" would be more easily understood and used than an abstract high level principle such as Norman's "reduce the gulf of evaluation"[12].

3.1 Generating Data for Pattern Identification

We began by collecting design principles and guidelines from the literature [4, 3, 8, 12, 13, 16, 17]. To limit the scope of the experiment we decided to use a subset of these principles, specifically Brown's collection of about 250 guidelines [5]. This collection was chosen because of its size and completeness. Each principle from this collection was placed on an index card. These cards were then grouped and regrouped, in an effort to co-locate principles that seemed related. If a principle seemed related to two groups, an evaluation of the groups was made, with one of three possible outcomes:

1. the groups were merged,
2. the principle was placed in both groups,

[1] It was in part the proliferation of these principles that caused us to begin to ask if there were not underlying patterns for human-computer interaction consistent with the principles that have been developed over the years.

	Analysis	Design	Implementation
Semantics			
Syntax			
Lexical			

Table 1: Matrix for HCI Principles in a Related Pattern

3. the principle was found to be more closely related to one of the groups, and was placed in that group.

Initially the groups created were based on what can be called visible surface structures such as punctuation, color coding, order, grouping, commands. As the groups were studied and reorganized, the groups that was created tend toward more conceptual relationships.

3.2 Identifying Patterns

To assist in the process of seeing patterns, we developed the matrix shown in Table 1. Sets of principles were placed on a matrix. While the implication of asking whether the principle related to analysis, design, and/or implementation is self explanatory, the classification of principles as lexical, syntactic or semantic requires further explanation. In practice, we came to interpret these classes as relating to the granularity or specificity of the principle being placed. For example, we classed a principle such as "avoid the use of red and green" as a lexical level principle – highly specific, granularity detail. We would class a principle about the maximum or optimal number of colors on a screen at the syntactic level, and one about using red as a danger color at the semantic level. While we were able to use these conceptual classifications operationally, we were never quite satisfied that we were able to capture meaningful general conceptualizations about the specificity/granularity/level of the principles.

After a matrix was populated from the initial grouping, it was examined to determine whether one of three conditions existed:

1. If sets of principles existed in a given cell, we sought to reduce the number of individual entries by generalizing the principles.

2. If cells were empty or sparsely filled in we looked to determine if the adjacent cells implied principles that might be inserted.

3. If principles from the initial group had not been placed, or if there were principles that had been placed on the matrix or that were clearly related, but not central, they were listed outside the matrix.

After studying the principles on the matrix, adding and subtracting principles, etc. a first attempt was made to give the clusters descriptive names.

3.3 Pattern validation

Having generated tentative patterns, we sought to validate of these patterns. With the initial pattern set we examined the grouping and content of the patterns to validate:

1. that the defined patterns are more general than the existing principles.
2. that others would organize principles in similar clusters, both with and without the aid of the defined patterns.

The study involved three tests, described below. All of the tests were run on twenty-nine (29) subjects. The subjects were from a graduate class in interactive system design. The experiment was run early in the course, before any discussion in the class of patterns and generative user-engineering principles. The subjects were part of one of four sessions which lasted up to 90 minutes.

3.3.1 Comparison of Principles and Patterns

A comparison was made of of pattern and principle statements aimed at identifying "more general statements". This test sought to verify that patterns can be distinguished from principles, that is that patterns are more general or descriptive than the principles used to define the pattern. The subjects were given ten sets of statements, and asked to select the most general or descriptive statement in each set. Each set had six statements, in nine of the sets there was a pattern statement as one of the statements, the last set had only principle statements, that is no statement should be more general than the other statements. Every effort was made to balance the statements so that they could not be distinguished based on surface structure, such as length, alone.

3.3.2 Grouping of principles

A grouping problem was proposed to identify the groups in which principles would be placed as seen by subjects. This test was aimed to see if the subjects grouped principles similar to the grouping done by the researchers, that is if the subjects 'saw' the same relations as the researcher. The subjects were asked to sort principles into sets of related principles. They were given 3 sets of 15 principles. A free form sorting procedure was used, which does not restrict the subjects to pre-defined categories. The groups created were analyzed using a pooled similarity matrix which was analyzed with regard to random grouping and compared to the groups anticipated by the researchers.

3.4 Classification of Principles

The classification problem sought to identify principle/pattern matches. The test aimed to see if the subjects could see the relation between given patterns and the principles from which they were defined. The subjects where given eight of the patterns and 35 principles, divided in two parts, each part containing four patterns and 18(17) principles. The task was to associate each principle with a pattern.

4 Results

4.1 Pattern Identification

Initially, we had hypothesized that the more of the matrix a set of principles covered, the more likely it would be that a pattern would be found. A matrix with principles in every box would be an optimal pattern. The ability to identify additional principles from a sparse matrix proved to be so easy as to make this kind of culling difficult. More importantly, we discovered that principles from different authors typically addressed only one phase of the development process. While this was not surprising, it provided additional incentive to find patterns which might provide a means through which cognitive scientists focused on the analysis phase might better communicate with computer scientists focused on the implementation phase.

After some fine-tuning of the clusters, nine patterns were defined and named.

- consistent information
- meaningful information
- informative interface
- universal commands
- meaningful sequence
- accommodating data
- ordered information
- recognizable objects
- appropriate information

See Appendix A for a full description of the patterns. Each of the patterns was described using the same format.

1. a title descriptive of the pattern's content but at the same time brief and easy to remember.
2. a paragraph describing the context of the problem solved by the pattern.
3. a rational of for pattern the empirical background of the pattern?).
4. examples of the use of the pattern in the design of an application.
5. a statement describing how to use the pattern.
6. reference to the different principles that constitute the pattern.

Studying the clusters of principles, possible additional patterns were identified at set aside for study at a later date. We did note that the patterns that were defined seemed to fall into the following categories:

- patterns related to the interface,
- patterns related to the interaction across the interface, frequently called the dialog, and
- patterns related to the information or objects that are represented both in the human mind and in the computer system.

4.2 Validation of Patterns

4.2.1 Distinguishing patterns and principles.

Using chi-square test on the results we found that for five of the ten tests the pattern statement was significantly chosen at the .05 level. Specifically, the patterns named were found to be more general than the accompanied principles. For the set without a pattern statement, it was expected that the statement "Not able to distinguish" would be the most frequent chosen, but instead the subject's choices were evenly distributed among the statements, from which one can make a similar conclusion. Four sets showed no preference for a pattern over a principle or showed a tendency toward an alternative (principle) statement. This result suggests that there is a need to go back to revise the pattern and/or the pattern statement of the affected patterns. These patterns included "universal commands", "meaningful sequence", "accommodating dialog", and "recognizable objects". It is worth noting that three of these, "universal commands", "meaningful sequence", and "accommodating dialog" all belong to the same group of patterns, specifically the patterns addressing the interaction across the interface.

4.2.2 Sorting principles into related sets

The data from this test yielded no conclusive results. At the same time, the data provide some information which might be used to revise the existing patterns. The findings from this test can be summarized as: subjects tend to group based on surface similarities, a few subjects appeared to group based on higher level relations, but this may be attributed to chance. By surface similarities we refer to groups created based on key words in the principle text, such as color coding, wording, layout. Many of the groups parallel those identified by the researchers in their initial grouping which also was based on surface similarities. Time may also have been a factor in the result; the subjects where only given 15 minutes to complete each set, the subjects were asked to name their groups with the 'feature' they saw as the relation between principles.

4.2.3 Classifying principles into patterns

Of the thirty five (35) principles to be classified, 26 were classified as anticipated – at a significance of .05 better. Two principles showed a positive but less than significant association to the anticipated pattern. Two principles showed no association trend. For one of the principles the subjects were divided between two patterns, the anticipated and an alternative. Closer examination suggests that the principle could easily be related to either pattern. Of particular interest were the four (4) principles anticipated to be associated with the pattern "consistent information." Of these, the subjects associated (significant at the .05 level) three of principles to the pattern called "ordered information" and one principle to the pattern "accommodating dialog". This leaves only one principle which significantly supports the pattern called "consistent information." This result would appear to suggest a reexamination of the validity of the pattern.

Thus, for seven of the patterns in the test the data show that the pattern statements subsume the principle used to define the pattern. One pattern "consistent information" needs to be revised, and compared to the patterns called "ordered information" and "accommodating dialog".

5 Conclusion

The work of defining patterns for human-computer interaction is far from complete. A first step has been taken – an initial set of patterns has been defined and validated to the extent that the patterns identified seem to be consistent with subject perceptions of valid patterns. The following conclusions appear justified at this point:

1. Principles can be clustered into groups of related principles. Placing the related principles on a matrix revealed that:
   (a) Many principles can be footnoted into a more general statement.
   (b) Principles can often be easily identified for empty matrix cells based on the principles in adjacent cells of the matrix,.
   (c) Patterns can be identified from clusters of principles.
2. We found that patterns span three aspects of the human-computer interaction: the interface itself, the interaction across the interface, frequently referred to as the dialog, and the information presented both in the interface and in the dialog.
3. In line with the operational definition, we found it possible to describe the pattern by a simple phrase and also to give them a descriptive name.

In the second part of, the research we tried to validate empirically the findings from the initial studies, from this we can conclude that:

1. Patterns are seen as more general than principles.
2. Subjects found clusters of principles. Both the initial groups created by the researcher and the results from test 2 show that principles can be clustered into groups of related principles. The grouping might vary some, depending on the focus of attention.
3. The study showed that the patterns defined as a part of this research subsume the principles that were used in the discovery of the pattern.
4. Half of the patterns identified in this study show some intersubject validity. These are "consistent interface", "meaningful interface", "informative interface", "ordered information" and "appropriate information". The other 50generality, and content. These are "universal commands", "meaningful sequence", "accommodating dialog", "recognizable objects" and "consistent information".

6 Future Research

While the results of the work so far indicate that patterns exist for human-computer interaction, full validation of the use of patterns must await the empirical evidence that patterns improve the design process and the quality of the interaction. It is left to demonstrate that the patterns can be used with ease in the analysis, design and implementation of an application. Further, we must be able to validate that the use of patterns in the development of an application creates a product that is more robust and resistant to technology changes than an application developed according to traditional software engineering methods. This section elaborates some of the research issues that remain in this area.

- Can a "simple" array of patterns be defined that completely subsumes and clarifies the many principles in the literature?

  The work of defining patterns is far from complete. While there is a need for more patterns, there is also a danger in generating too many patterns. Having too many patterns would put the designer in the same position as today, where principles are too numerous to be used successfully. Can a limited set of patterns be generated that clearly subsumes the existing principles in such a way that selecting a random principle, it can be said that there exists a pattern "X" that addresses the topic of the principle?

- Can communication between the phases of interactive system design be shown to improve with the use of patterns?

  While general design principles such as those proposed by Shneiderman (1992), Norman (1988), and Brown (1988) are powerful, they tend to focus on activities in a single phase of the project. This might maximize the information gain and minimize the loss through noise in that specific phase, but do not guarantee the quality of information in other phases. For example, a cognitive scientist might be able to gather important information from the user environment, but if this is not communicated to the designer and implementor in such a way that they can understand and use it, the effort was futile. It is the belief of this researcher that a pattern language can be used to improve the communication among the different phases of the design process, and that the use of patterns as such a tool will maximize the information gain and minimize the information loss throughout the whole process.

- Can novel situations be addressed more easily by the use of patterns?

  When plotting the principles on a matrix it was discovered that new principles could be generated based on the existing principles. The patterns defined are based on these matrices. Can the ability to generate new principles have been transferred to the patterns in such a way that existing patterns can be used to solve problems in novel situations?

- Will the use of patterns have an impact on programming of software?

  Can the use of patterns in the design of software affect the quality of the software produced? Will series of qualitative and quantitative software performance measures show different results if patterns are used in the design?

Each of these issues must be addressed in order to completely verify and validate the existence and applicability of patterns in human-computer interaction. The most important issue is to develop a larger set of patterns, which is a requirement for the other research issues. We are currently reviewing the clustered principles and patterns to make necessary changes and to see if we find new additional patterns in the existing material. Further it will be interesting to take a new set of principles to see how they will fit into the current framework and if they will spur the development of new patterns.

References

[1] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language.* Oxford University Press, New York, NY, 1977.

[2] Christopher Alexander. *The Timeless Way of Building.* Oxford University Press, New York, NY, 1979.

[3] Susanne Bødker. *Through the Interface: A Human Activity Approach to User Interface Design.* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1991.

[4] Paul Booth. *An Introduction to Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1989.

[5] C.M. Brown. *Human-Computer Interface Design Guidelines.* Ablex Publishing Corporation: Norwood, NJ, 1988.

[6] Judith Brown and Steve Cunningham. *Programming the User Interface: Principles and Examples.* John Wiley and Sons, 1989.

[7] Peter Coad. Object-Oriented Patterns. *Communication of the ACM*, 35(9):152–159, September 1992.

[8] Joseph S. Dumas. *Designing User Interfaces for Software.* Prentice-Hall, Inc., Englewoods Cliff, NJ, 1988.

[9] Jeff Johnson, Teresa Roberts, William Verplank, David C. Smith, Charles Irby, Marian Beard, and Kevin Mackey. The xerox star: A retrospective. *IEEE Computer*, 22:11–30, September 1989.

[10] Ralph E. Johnson. Documenting Frameworks using Patterns. *OOPSLA'92*, pages 63–76, 1992.

[11] Tom Love. Timeless design of information systems. *Object Magazine*, pages 42–48, nov/dec 1991.

[12] D.A. Norman. *The Design of Everyday Things.* Basic Books, New York, NY, 1988.

[13] Ben Shneiderman. *Designing the User Interface: Strategies for effective Human-Computer Interaction.* Addison-Wesley Publ. Co.: Reading, MA, 1992.

[14] Ingjerd Skogseid. A Preliminary Analysis of Architectural Patterns in Human-Computer Interaction. Master's thesis, School of Library and Information Science, 1995.

[15] David C. Smith, Charles Irby, Ralph Kimball, and Bill Verplank. Designing the Star User Interface. *Byte*, 7(4):242–282, April 1982.

[16] Sidney L. Smith and Jane N. Mosier. Guidelines for Designing User Interface Software. Technical Report ESD-TR-86-278, The MITRE Corporation Bedford, MA, August 1986.

[17] Alistair Sutcliffe. *Human-Computer Interface Design.* Macmillan Education, 1988.

[18] Harold Thimbleby. *User Interface Design.* Frontier Series. Addison-Wesley Publ. Co.: Reading, MA, 1990.

[19] Ed Yourdon. Object-Oriented Design. *American Programmer: Ed Yourdon's Software Journal*, 3(3):14–24, March 1990.

Appendix A: Interface Patterns

Consistent Interface

The consistency of an interface is important for the user while building an understanding of the system, developing a mental model of the system.

Rationale for the pattern: *Individuals have a limited attention span and working memory store. To the extent that novel items demand attention to decode them, they detract from the main task at hand. The purpose of developing a consistent interface is to minimize this distraction.*

Relevant considerations: A consistent system allows users to build mental models that will aid in the interaction and in the operation of the application. If consistency is missing, users have to rely more heavily on memorization instead of building and relying on mental models. When a user has a mental model of how the application performs the task, it will be easier for the user to perform his or her task.

A consistent system is easier to learn and use. To achieve consistency it is important that the development team of the application specify a set of conventions to be used during the development of the product. In the analysis phase it is therefore important to keep your eyes open for conventions followed in the present environment that can be moved into the new environment.
Here are some things that can help keep the interface consistent:

- Make information follow conventional standards as much as possible. For instance, use conventional English punctuation whenever possible and use both upper and lower case for the text. When displaying long numeric fields, use conventional punctuation schemas if they exist, punctuated with spaces, commas, hyphens, slashes, or whatever is most appropriate.

  Lists of alphabetic data should be left-justified and numeric data should use decimal alignment, so that data without decimals are right-justified.

- Each screen should have a heading that defines its contents.

- Label each data field to identify it for the user, thus reducing the memorization and providing for recall. Position the labels consistently, e.g. above columns and to the left of fields. Make the labels so they are easy to distinguish from the data. It may also be desirable to separate or partition groups of related data with blank spaces or any other means of separation.

- When expecting user input use one convention to prompt the user. Make it clear what actions are expected of the user.

Therefore: Classify information to identify similarity. Similar information should be dealt with in consistent form.

References: Brown (1988) 2.1, 2.9, 2.11, 2.12, 2.14, 2.15, 2.16, 2.17, 2.18, 2.28, 2.29, 7.7, 7.13, 7.22, 7.23, 7.28

Meaningful Interface

A display that consists of a set of what might seem like an arbitrarily selected set of commands or controls will confuse the novice and intermittent user.

Rationale for pattern: *To make the designer think about how to design the layout of a screen and also how to best combine the controls that make up a display.*

Relevant considerations: In a meaningful interface each label and control needs to be given a considerable amount of thought. The layout and combination need to have a clear purpose and meaning to the user. He needs to intuitively know or recognize how it is to be operated and its function. Label each control so it is clear to the user what it controls and what its intention is.

Use meaningful codes, as they are easier for the user to both recognize and recall, likewise for mnemonic codes and accelerator keys. Single letter mnemonic codes will be easier to remember if one uses the initial letter in the option name.

Use familiar units so the user recognizes the information without having to estimate its value in a known unit. If translations are needed to another unit, the computer can easily do this.

Use conventional colors to display data, but never use color alone. If there is a color convention that is generally recognized in the users environment, apply it. Here is a list of recommended color conventions:

Red: Present data associated with alarms, error conditions, and danger in red. Red also has a strong association with hot temperatures and is therefore well suited to display this kind of data.

Yellow: Present warnings and non critical data in yellow.

Green: Present normal conditions in green. Green also has a strong relation to "go".

Blue: Use blue to present cold temperatures. Blue can also be used as background, graphics and data that are not critical; it should not be used for primary data.

Therefore: Each control in the display should have a specific purpose, and should be selected with extra care to make the relationship between two controls and between control and its effect clear. It is also important to make the labels clear and descriptive so the user is aware of the choices and options available at all times.

References: Brown (1988) 2.27, 4.19, 4.20, 4.21, 4.22, 4.23, 4.24, 4.25, 7.6, 7.16

Informative Interface

Without sufficient information the user will get lost, regardless of whether it is a novice, an intermittent or an expert user. An interface without sufficient information about the application and/or the system state will cause frustration and the users will abandon the application.

Rationale for the pattern: *In any task, the 'task doer' requires orientation time and focus. To the extent the user must make a new script to orient themselves, e.g. if it is a complicated process to get on-line help, the more the user will be distracted from the task at hand.*

Relevant considerations: An informative interface provides the user with sufficient information at all times. By sufficient information we mean;

- Instructions should be distinguishable from data.
- The user should be able to distinguish optional from required entries.
- The results of user actions should be visible in the interface without delay.
- Acknowledge user input and progress, indicating percent done or hourglass while waiting, and showing changes immediately after.
- Echo user entries instantaneously.
- The user should be able to identify all the data displayed.
- The interface should provide information on how to continue.
- Display all available/valid options.
- Display continuing page indicator.
- The user should be able to determine the settings of all the dialog parameters.
- Have an indicator displaying the application/system status.
- Display transaction type.
- Show the units for each variable in the heading or the label associated with it.
- Show default values.
- Show correct input format/syntax.
- Indicate length of field.
- Display heading on each page.
- List heading defines list.
- Display headings on each page.
- Keep display uncluttered.
- User should be able to access help at each stage in the process.

Therefore: Always display enough information. The user should be able to rely on recognition and not on recall. The user should be able to assess how to continue. The user should find all the needed information on the same screen as the task. The user should be able to read the state of the application or system from each display.

References: Brown (1988) 2.25, 2.30, 2.32, 2.36, 2.41, 2.42, 3.13, 6.1, 6.2, 6.3, 6.4, 6.5, 6.7, 6.9, 6.11, 6.12, 6.41, 6.44, 6.45, 6.46, 7.8, 7.9, 7.10, 7.11, 7.15, 7.18, 7.24

Appendix A: Interaction Patterns

Universal Commands

When designing interactive systems it is important to look for and to recognize universal commands and operations.

Rationale for the pattern: *To ease the memory load of the user it is essential that commands with the same function are executed in the same way throughout the application.*

Relevant considerations: Consistency is important, for both single actions and classes of actions, to reduce the user's memory load. The activation of an action should be the same at all valid points throughout the application. The naming of an action should also be consistent. In addition it is important to use a naming schema that is in line with the user's expectations. Use unambiguous action words as much as possible. It might also be desirable to keep activation consistent across applications; that is, some procedural activities should be standardized.

Therefore: Command structure across the application domain should be identified and common commands abstracted in the system.

References: Brown (1988) 2.3, 6.34, 6.35

Meaningful Sequence

A dialog should have a meaningful sequence. The sequence of operations can be essential to the task performed.

Rationale for the pattern: *Many humans have a tendency to organize information into lists and order them after some organizing principle. It is therefore important that computer applications provide the same possibility, and present the instructions and information in a way that is meaningful to the user.*

Relevant considerations: Use existing organizing principles if they are obtainable. Some standard organizing principles are: chronological, alphabetic, sequential, functional order, level importance, frequency of use, and physical proximity. Use the organizing principle that is most useful or appropriate at each instance, but for consistency, use the same organizing principle throughout the application. It may be desirable to combine organizing principles, to use one for groups of commands and another within the group.

Therefore: The dialog sequence in the application domain should be identified and replicated in the system.

References: Brown (1988) 2.21, 2.26, 2.39, 3.26, 6.10, 6.19, 6.36, 7.25

Accommodating Dialog

When designing interactive systems the designer should incorporate the needs of novice and intermittent users as well as expert users.

Rationale for the pattern: *Each user has different needs and requirements of the interaction with the computer. There are personal differences, but also differences dependent on how experienced the user is with computers in general, and with the specific application.*

Relevant considerations: User vary along numerous dimension: knowledge of the domain, the system expertise , frequency of use, etc.

Each user has different needs in interaction with the computer. A good interface should provide support for each type of user. To ease exploration and learning of an application or a system it is important to make fundamental functions easy to learn, make frequent functions easy to perform, and to encourage exploration by minimizing the consequence of an error, by providing recovery options. In addition system defaults should produce the most likely outcome.

It should be easy to access the menus in the system. The wording of the menus should be distinct and specific to make them easier to remember. For consistency one can gray-out options that are not in use or are not valid within a specific context. Shortcuts to options/functions that are frequently performed should be provided. Also permit users to create command sequences or macros that will make their work easier. Reduce keying requirements by providing command keyword completion. Do not require the user to reenter data that can be extracted from data already in the computer. Do not require users to enter the units of measures.

The user should be able to update the information both during and after entry. That is, the user can update the information he has authority to alter. Also he should be able to recover from unwanted or incorrect actions by, a function that exist in many systems that achieve this is the "undo" function.

Therefore: With respect to utilizing commands, the application or the system should provide for recognition when it is needed and for recall when it is more efficient. Feedback from the application and the system should show the outcome of operations and the state of the system.

References: Brown (1988) 6.13, 6.14, 6.15, 6.16, 6.20, 6.38, 6.40, 7.14, 7.17, 7.19,7.20, 7.21, 7.27

Appendix A: Information Patterns

Ordered Information

When designing interactive systems it is important to be aware of the order that is set to present information. The users mental model is frequently based on an organizing principle.

Relevant considerations: Information should be organized and presented in a useful order. Depending on the task and the user community, different organizing principles can be used: chronological, alphabetical, sequential, functional, proximity, frequency of use, or level of importance. Information needs to be organized into logical groups. When the information describes how to perform a complex task, the order of the instructions should mirror the order in which the sub tasks need to be performed. When enumerating items, use numbers not letters, and separate the numbers clearly from the text.

Therefore: Information should be organized so as to identify the organizing principle in the users environment. This principle should be used to present the information in a consistent manner.

References: Brown (1988) 2.10, 2.21, 2.22, 3.37, 2.38, 2.40

Recognizable Objects

There is no reason to require users to learn new terminology in addition to learning a new application. A new system may be hard enough to learn without this additional hindrance.

Rationale for the pattern: *Humans are better at recognition than at recall. It will be easier for users to learn and perform a task if they can rely on recognition.*

Relevant considerations: Use of terms and analogies that are familiar to the user are of great help when learning a new system. At the same time there needs to be some caution to in the use of analogies which might cause more harm than gain in the end.

Follow standard data formats or common conventions in the user community. Let users use the data units that are most appropriate in the community, and the system can perform necessary conversions.

Abbreviate only when space is a concern, and only when the abbreviation is significantly shorter. Be consistent, using only one abbreviation per word. Abbreviate only commonly abbreviated words, otherwise it may be hard for the user to understand the meaning of the instructions.

Use of color codes should either reflect normal color meaning (see the pattern meaningful interface on page 12), or task relevant color codes adopted from the user's community.

Therefore: Objects should be identified and named in the system consistently with their naming in the users domain.

References: Brown (1988) 2.5, 2.6, 2.19, 3.1, 3.2, 3.6, 3.12, 3.15, 3.16, 4.11, 4.12, 7.16

Appropriate Information

To perform an operation, the user needs access to appropriate and sufficient information at every stage in the application.

Relevant considerations: The program should respond to every user input or action. If an error should occur, inform the user by displaying a message describing the problem. Make the message instructive by informing the user about which error occurred, where it occurred and maybe suggest corrective actions. The messages should be brief, specific, and appropriate to the user's level of knowledge.

All documentation for the application should be available on-line, and in a search able form. Context sensitive help should be available. That is, by giving one command the user should be given help related to the screen he is currently on, with the ability to explore other issues. Help should be given at all levels of granularity, for screens, fields, codes, and messages. Help should be accessible at all points in the application. The user should be able to choose different levels of help, since a novice user needs more assistance than an expert user.

Therefore: At every stage an application should provide the information necessary to perform the operation intended and provide context-sensitive help so the user can get additional help when it is desired. All documentation should be available on the computer, as well as on paper.

References: Brown (1988) 2.13, 3.4, 3.11, 3.17, 6.8