Empirical Software Engineering, 4, 71–104 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

The Usability Problem Taxonomy: A Framework for Classification and Analysis

SUSAN L. KEENAN, PH.D slkeenan@tiac.net
Shrewsbury, MA 01545

H. REX HARTSON, PH.D. hartson@vt.edu
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061

DENNIS G. KAFURA, PH.D. kafura@cs.vt.edu
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061

ROBERT S. SCHULMAN, PH.D. schulman@vt.edu
Department of Statistics, Virginia Tech, Blacksburg, VA 24061

Abstract. Although much can be gained by analyzing usability problems, there is no overall framework in which large sets of usability problems can be easily classified, compared, and analyzed. Current approaches to problem analysis that focus on identifying specific problem characteristics (such as severity or cost-to-fix) do provide additional information to the developer; however, they do not adequately support high-level (global) analysis. High-level approaches to problem analysis depend on the developer/evaluator's ability to group problems, yet commonly used techniques for organizing usability problems are incomplete and/or provide inadequate information for problem correction. This paper presents the Usability Problem Taxonomy (UPT), a taxonomic model in which usability problems detected in graphical user interfaces with textual components are classified from both an artifact and a task perspective. The UPT was built empirically using over 400 usability problem descriptions collected on real-world development projects. The UPT has two components and contains 28 categories: 19 are in the artifact component and nine are in the task component. A study was conducted showing that problems can be classified reliably using the UPT. Techniques for high-level problem analysis are explored using UPT classification of a set of usability problems detected during an evaluation of a CASE tool. In addition, ways to augment or complement existing problem analysis strategies using UPT analysis are suggested. A summary of reports from two developers who have used the UPT in the workplace provides anecdotal evidence indicating that UPT classification has improved problem identification, reporting, analysis, and prioritization prior to correction.

Keywords: Usability problem classification, usability problem analysis, problem prioritization

1. Introduction

Researchers and practitioners in all disciplines involved in creating interactive software systems have found that the collection and analysis of software data contribute significantly to both product and process improvement strategies. These data include coding data, process data, data about software defects (bugs), and data about usability problems (Grady, 1994; Nielsen, 93; Preece et al., 1995; Ostrand and Weyuker, 1994; Pfleeger, 1991; Sommerville, 1989; Pressman, 1997). Two critical aspects of an interactive software product are software defects and usability problems. Consequently, much attention has been focused on the identification, classification, and analysis of both software defects in the software engineering community and usability problems in the human computer interaction community.

Software defects classified according to the development phase in which they were detected (or according to the activity performed) enable developers to determine the effectiveness of individual development activities (Weiss, 1979). Classification according to module size, name, number of modules affected by a proposed change, or system function can guide system tests by identifying where defects are most likely to be found (Myers, 1979). Assessment of defect severity, cost-to-fix, or time-to-fix helps developers prioritize defects prior to correction (Pressman, 1997). In addition, many development teams perform defect causal analysis to determine when, and what, corrective actions are needed (Card, 1993; Card, 1998; Giblin, 1992).

In the context of user interface development, current research suggests that, in addition to examining individual usability problems, much can be gained by analyzing sets of problems. One approach to problem analysis involves having a trained individual examine the entire set of usability problems detected on a given project looking for trade-offs, contradictions, and consistency issues (Jeffries, 1994). A second approach focuses on thinking about the problems from a global perspective and looking for problem clusters (Dumas and Redish, 1994). Although the ability to organize or group usability problems is crucial to the success of either approach, there is no overall framework in which potentially large sets of problems can be classified. The goal of this research effort was to build a framework for classification that supports problem analysis from various perspectives.

Many different techniques are currently used to organize and/or classify usability problems. Some clustering results from using problem checklists (lists of problems observed repeatedly on a variety of products) to search for global problems, i.e., problems that have scope larger than one screen or page (Dumas and Redish, 1994). Problems are also classified as a result of performing usability evaluation techniques such as heuristic analysis (Nielsen and Molich, 1990; Nielsen, 1992) and perspective-based usability inspection (Zhang, Basili and Shneiderman, 1999). Some researchers have explored classification according to human error (slips, mistakes, omissions, substitutions, and/or unnecessary repetitions) in an attempt to provide a deeper understanding of usability problems (Mack and Montaniz, 1994; Senders and Moray 1991). Others have investigated classification according to location in the dialogue (Mack and Montaniz, 1994; Nielsen, 1992; Nielsen, 1994), core problems (Jeffries, 1994; Nielsen, 1994), severity (Desurvire, 1994; Nielsen, 1993; Nielsen, 1994; Rubin, 1994), frequency (Dumas and Redish, 1994; Nielsen, 1994), impact (Karat et al., 1992; Nielsen, 1994), and cost-to-fix (Hix and Hartson, 1993). A new method for designing hypermedia that utilizes task models and user types during design and evaluation groups usability problems according to user type (Paternò and Mancini, 1999). Questionnaires that measure user satisfaction with an interactive software system such as SUMI (Porteous et al., 1993) and QUIS (Chin and Norman, 1988; Shneiderman, 1998) also classify problems to some extent as questionnaire responses provide a high-level view of difficulties users encountered with the system. This high-level view guides system redesign by enabling managers to identify potential areas for improvement.

Although current techniques for usability problem classification and analysis reveal much about usability problems, difficulties with these approaches often limit their success. Global problem checklists intended for use in reviewing product usability are a good first step in organizing usability problems; however, the lists are not sufficiently comprehensive to be used in systematic classification. The utility of classification that is essentially a by-product of an evaluation technique such as heuristic analysis can be limited because classification was not the intended purpose of the technique. In section 2.4, we discuss specific difficulties we encountered in our attempt to use the heuristics (Nielsen, 1994) as the basis for a classification scheme. Techniques used in human error analysis provide insufficient information for problem correction (Vora, 1995). Classification according to criteria such as frequency, location in the dialogue, and severity does provide information about problem pervasiveness and importance. However, when these techniques are applied to large data sets, we believe that isolated and more-or-less unrelated characteristics of the set of problems are revealed. User satisfaction questionnaires do provide a useful, high-level view of usability problems encountered by the subjects while using the software system. However, the focus of the questionnaires is on the user's feelings about, or reactions to, the user interface, not on analysis of individual usability problems encountered on specific screens.

We believe that an overall framework for classification will enable developers to analyze large data sets more easily. A framework containing categories that capture the essence of individual usability problems will provide a new interpretation of problem "type." This kind of classification will result in more meaningful problem clusters and thus facilitate the identification of global problems and those that share common characteristics. Our own experiences indicate that using an appropriate classification scheme advances our understanding of very large sets of problems. Trends, patterns, surprises, and unusual occurrences across problem sets are more easily identified. We have also found that using a structured environment to compare, contrast, and explore usability problems from various perspectives has altered our approach to problem prioritization and correction.

This paper presents the Usability Problem Taxonomy (UPT), a taxonomic model for classification of usability problems detected on graphical user interfaces with textual components. In section 2 we describe the structure of the UPT (section 2.1), problem classification (section 2.2), the UPT Web tool (section 2.3), and how the UPT was developed (section 2.4). In section 3, we present the results of the UPT reliability study in sections 3.1 and 3.2, discuss the study data from a non-statistical perspective in section 3.3, and examine factors that influenced the reliability results in section 3.4. Using UPT classification of a set of usability problems detected on a CASE tool, we discuss how UPT analysis enables developers to identify characteristics of usability data in section 4.1. Section 4.2 outlines how the UPT analysis can augment and/or complement existing problem analysis techniques. In section 4.3, we explore new approaches to problem prioritization. Reports from two developers who have used the UPT in the workplace are summarized in section 5. We conclude with a brief examination of future research directions in section 6.

2. The UPT

2.1. Structure of the UPT

The UPT, illustrated in Figure 1, contains an artifact component and a task component (Keenan, 1996). The two components are divided hierarchically into five primary cat-

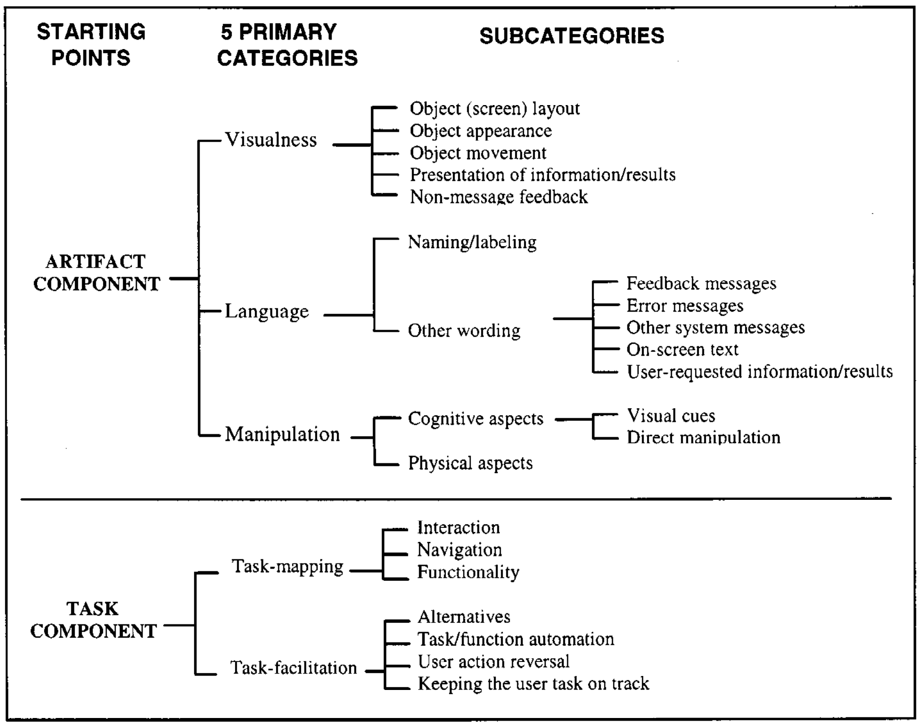


Figure 1. The Usability Problem Taxonomy (UPT).

egories of which the artifact component contains three (visualness, language, manipulation) and the task component contains two (task-mapping, task-facilitation). Each primary category contains multiple subcategories (e.g., the visualness category contains five subcategories: object (screen) layout, object appearance, object movement, presentation of information/results, and non-message feedback). Two subcategories, other wording and cognitive aspects, also contain subcategories.

Categories in the artifact component focus on difficulties observed when the user interacts with individual user interface objects. Specifically, artifact classifications concern aspects of problems related to the way the user (1) examines or views user interface objects (visualness), (2) reads and understands words (language), or (3) manipulates user interface objects (manipulation). Categories in the task component focus on difficulties encountered as the user moves *through* a task. In particular, task classifications are about aspects of problems related to the way a user task is structured on the system (task-mapping) and the system's ability to help the user follow the task structure and return to the task when a deviant path has been taken (task-facilitation).

The overall structure of the UPT is based on the premise that usability problems should be examined from two perspectives: the artifact perspective and the task perspective. This approach to problem classification is compatible with two approaches used during the design phase: the task-artifact approach (Carroll et al., 1991) and the Object-Action Interface Model (Shneiderman, 1998).

2.2. Problem Classification

To classify a given problem, an evaluator uses both the artifact and task components. Within a component, however, the categories and subcategories at any given level are mutually exclusive, i.e., classification within a given component results in one final categorization. To classify a problem within a component, the evaluator proceeds as deeply as possible within that component, i.e., until no further progress can be made. The evaluator then classifies the problem as deeply as possible in the other component.

The classification process along a component may result in one of three outcomes: full classification (FC), partial classification (PC), and null classification (NC).

- Full classification implies that a problem was classified in a rightmost subcategory, i.e., the problem was classified to the deepest level in Figure 1.
- Partial classification implies that a problem was classified in either a primary category or a subcategory that is not at the deepest level, e.g., the other wording subcategory in Figure 1.
- Null classification implies that no category was selected along a given component. An NC classification is not a non-result, but is a specific outcome that conveys certain information about the problem and its description.

Thus, classification results in a pair of outcomes, e.g., a problem could be fully classified in the artifact component and partially classified in the task component.

The three usability problems described in Table 1 demonstrate different pairs of classification outcomes. Problem 1 was fully classified in the artifact component, but received an NC classification along the task component. Problem 2 was partially classified in the artifact component (to the primary category level only) and received an NC classification along the task component. Problem 3 was fully classified in both components. Note that the Artifact Classification column outlines the path taken through the artifact component to achieve a final classification. Similarly, the Task Classification column outlines the path through the task component.

During the course of this research effort, we noted that vague or incomplete descriptions as well as descriptions of superficial (and possibly easy-to-fix) problems could not be fully classified in one or both components. Vague descriptions lack sufficient information about exactly what occurred. Consider problem 2 in Table 1. Problem 2 is about language, but the lack of specificity prevents the classifier from classifying more deeply than the primary category level. Unless the classifier was present during the user testing session, he/she would not know which "kind of" words caused the problem (e.g., words used in naming, in on-screen text, in feedback messages, etc.). Incomplete descriptions lack information about the user task or what the user was doing at the time the problem occurred. In

Table 1. Three sample classifications.

#	Problem Description	Artifact Classification	Artifact Outcome	Task Classification	Task Outcome
1	The OK button on the different size from the XYZ screen is a OK button on the previous screens.	Visualness Object appearance	FC	Task	NC
2	The user does not understand the words used in dialogue box XXX.	Language	PC	Task	NC
3	The highlighted name field (in the disk format dialogue box) is much wider than the allowable length of a disk name. The user thought that the disk name could be at least as long as the highlighted field width.	Manipulation Cognitive aspects Visual cues	FC	Task-facilitation Keeping the task on track	FC

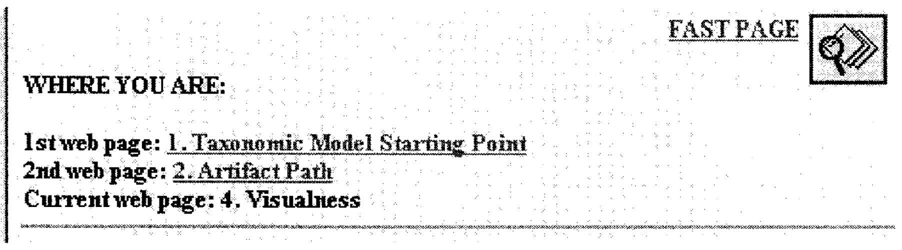
our experience, incomplete descriptions often impact classification in the task component. Superficial problems often receive an NC classification in one component. For example, the description of problem 1 in Table 1 is solely about inconsistent button sizing and does not contain any information about the user's movement through the task. As a result, this problem receives an NC classification along the task component.

The UPT does not have an "other" category to handle the situation that arises when a classifier can proceed no further using available categories. Instead, the classifier backs up one level in the UPT and records that category. Consider problem 2 in Table 1. The classifier cannot proceed further than the language category in the artifact component. Instead of classifying the problem in an "other" subcategory of the language category, the classifier records the classification as simply a language problem. Similarly, instead of classifying the problem in the task component as an "other" problem, the classifier records a classification of just task.

2.3. UPT Web Tool

The UPT was implemented as a collection of the following Web-based documents:

- an overview of the UPT approach to problem classification,
- descriptions of each component and category,


FAST PAGE

WHERE YOU ARE:

1st web page: 1. Taxonomic Model Starting Point
2nd web page: 2. Artifact Path
Current web page: 4. Visualness

Figure 2a. Header section of UPT visualness web page.

- instructions with sample classifications,
- a glossary of terms, and
- a site map, called the “Fast Page,” for experienced classifiers desiring fast navigation.

The URL for the Web-based documents is: http://www.tiac.net/users/slkeenan/upt/uptcover.html.

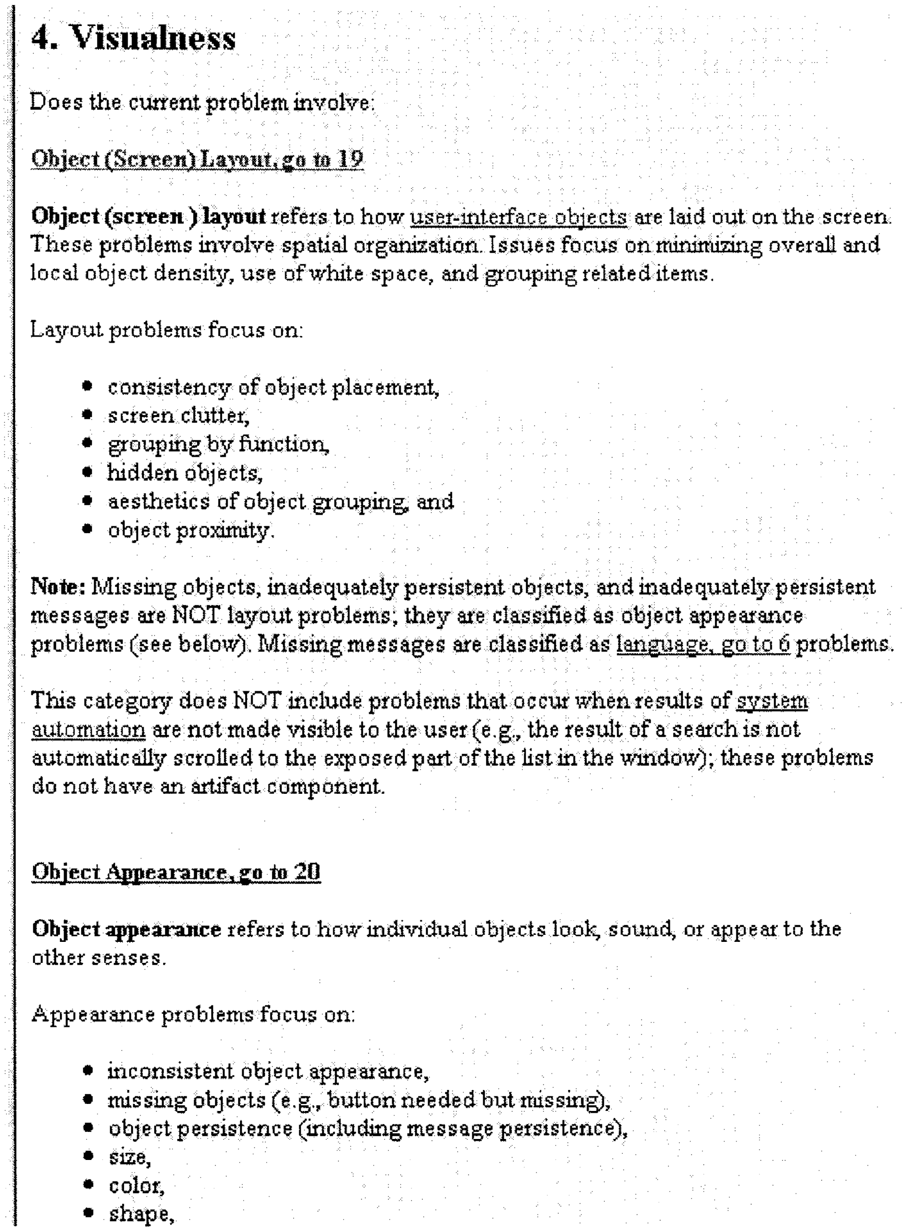
Each category (and subcategory) in the UPT hierarchy corresponds to one web page. The web pages are color coded to help novice classifiers more easily associate categories with components. Red font is used in the documentation when categories in the artifact component are referenced. Blue is used for categories in the task component. Correspondingly, artifact pages have a pink background and task pages have a pale blue background.

UPT Web Page Structure

Each UPT web page is divided into three sections: a header section, a middle section, and a “None of the Above” section. These sections for the visualness category are shown in Figures 2a, 2b, and 2c, respectively.

The header section contains a link to the Fast Page on the right and an outline of “Where You Are” in the hierarchy on the left. The Fast Page is a web page that contains a list of all UPT web pages. The Where You Are outline contains the most direct path from the starting point in *that* component to the current page. The Where You Are outline is not dynamic, i.e., it may not reflect the path the classifier actually used to arrive at the current web page. The header section for the visualness web page is shown in Figure 2a. Note that the Where You Are outline shows the most direct path from the starting point in the artifact component to the visualness page.

The middle section of each web page contains descriptions of, and links to, subcategories that could be chosen to more deeply classify the problem within a component. This section also contains explanations and guides for classifying specific problems, examples of problems that should *not* be classified in that category, and suggestions regarding choosing the correct category if an erroneous path has been taken. Figure 2b shows the top portion of



4. Visualness

Does the current problem involve:

Object (Screen) Layout, go to 19

Object (screen) layout refers to how user-interface objects are laid out on the screen. These problems involve spatial organization. Issues focus on minimizing overall and local object density, use of white space, and grouping related items.

Layout problems focus on:

- consistency of object placement,
- screen clutter,
- grouping by function,
- hidden objects,
- aesthetics of object grouping, and
- object proximity.

Note: Missing objects, inadequately persistent objects, and inadequately persistent messages are NOT layout problems; they are classified as object appearance problems (see below). Missing messages are classified as language, go to 6 problems.

This category does NOT include problems that occur when results of system automation are not made visible to the user (e.g., the result of a search is not automatically scrolled to the exposed part of the list in the window); these problems do not have an artifact component.

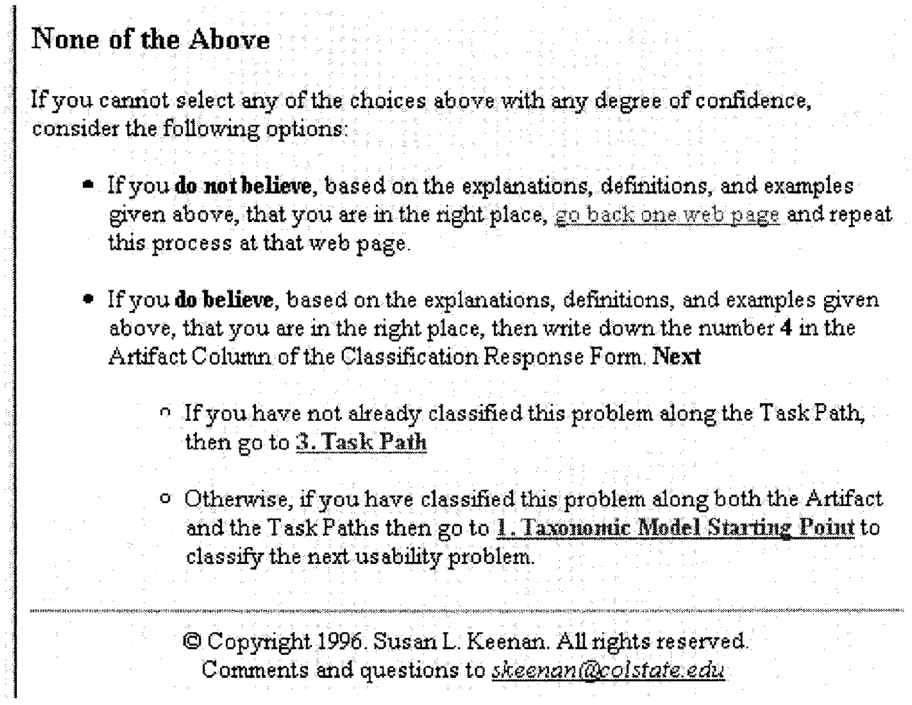
Object Appearance, go to 20

Object appearance refers to how individual objects look, sound, or appear to the other senses.

Appearance problems focus on:

- inconsistent object appearance,
- missing objects (e.g., button needed but missing),
- object persistence (including message persistence),
- size,
- color,
- shape,

Figure 2b. Portion of middle section of UPT visualness web page.



None of the Above

If you cannot select any of the choices above with any degree of confidence, consider the following options:

- If you **do not believe**, based on the explanations, definitions, and examples given above, that you are in the right place, go back one web page and repeat this process at that web page.
- If you **do believe**, based on the explanations, definitions, and examples given above, that you are in the right place, then write down the number **4** in the Artifact Column of the Classification Response Form. **Next**
  - If you have not already classified this problem along the Task Path, then go to **3. Task Path**
  - Otherwise, if you have classified this problem along both the Artifact and the Task Paths then go to **1. Taxonomic Model Starting Point** to classify the next usability problem.

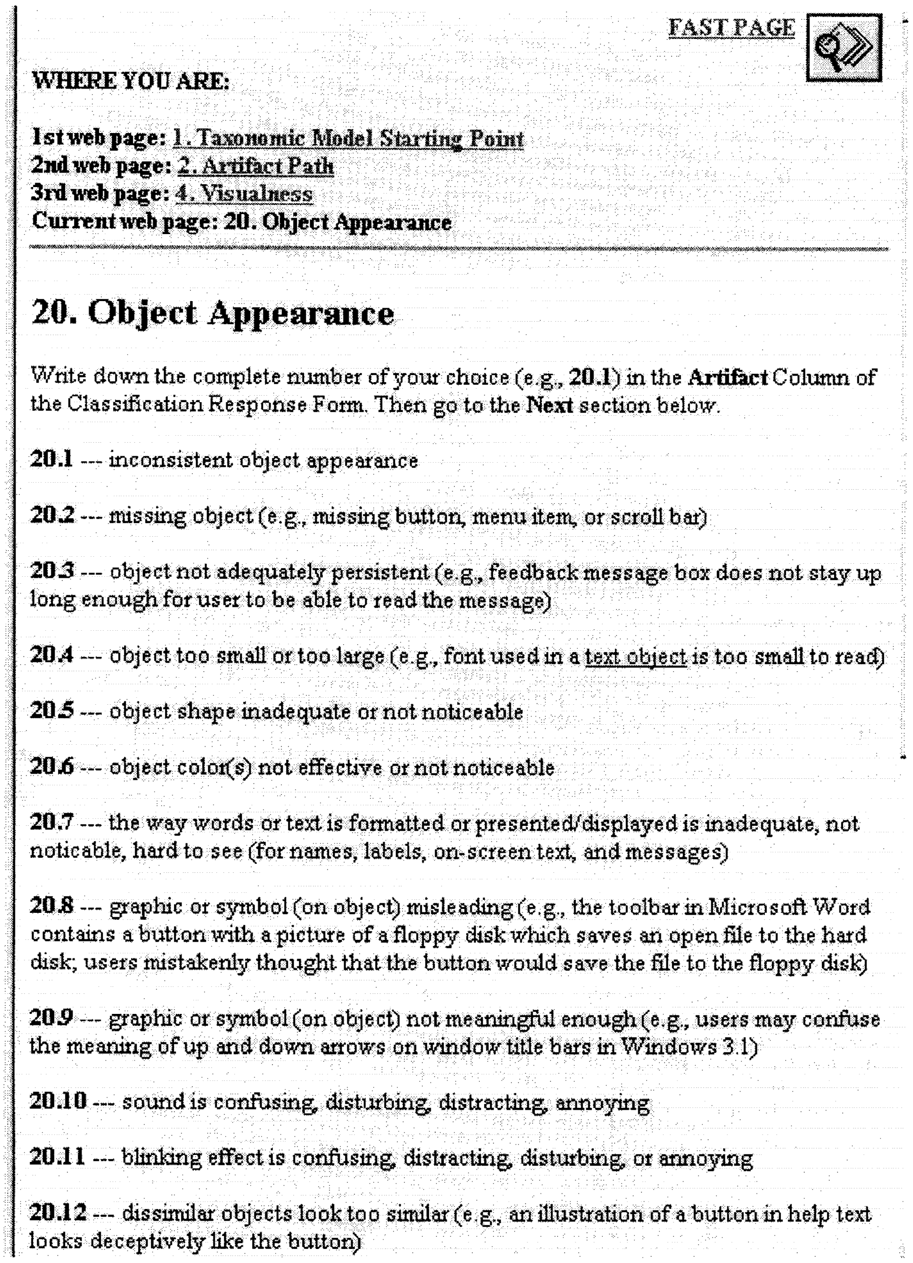
© Copyright 1996. Susan L. Keenan. All rights reserved.
Comments and questions to *skeenan@colstate.edu*

Figure 2c. None of the Above section of UPT visualness web page.

the middle section for the visualness web page. Links and explanations for two of the five visualness subcategories (object layout and object appearance) are illustrated in the figure.

The “None of the Above” section of the visualness web page is illustrated in Figure 2c. This section provides instructions for classifiers if none of the links in the middle section seem appropriate. Classifiers who believe that they are in the “wrong place” can back up and recheck the explanations provided in the higher level category. Classifiers who believe they are in the correct category but cannot choose a subcategory are instructed to record the current category as the classification. Consider the visualness web page. If a classifier believes that visualness is not the correct category, he/she would back up one level and recheck the explanations and options on that page. If the classifier believes that visualness is the appropriate category, but is unable to progress further, a classification of visualness would be recorded.

The Web page for each rightmost subcategory in Figure 1 contains examples of the kinds of problems that would be classified in that subcategory. The problem lists assure the classifier that he/she has correctly classified a problem. The problem list for the object appearance subcategory contains 12 problem descriptions as shown in the middle section of the Web page in Figure 3.


FAST PAGE

WHERE YOU ARE:

1st web page: 1. Taxonomic Model Starting Point
2nd web page: 2. Artifact Path
3rd web page: 4. Visualness
Current web page: 20. Object Appearance

20. Object Appearance

Write down the complete number of your choice (e.g., **20.1**) in the **Artifact** Column of the Classification Response Form. Then go to the **Next** section below.

20.1 --- inconsistent object appearance

20.2 --- missing object (e.g., missing button, menu item, or scroll bar)

20.3 --- object not adequately persistent (e.g., feedback message box does not stay up long enough for user to be able to read the message)

20.4 --- object too small or too large (e.g., font used in a text object is too small to read)

20.5 --- object shape inadequate or not noticeable

20.6 --- object color(s) not effective or not noticeable

20.7 --- the way words or text is formatted or presented/displayed is inadequate, not noticable, hard to see (for names, labels, on-screen text, and messages)

20.8 --- graphic or symbol (on object) misleading (e.g., the toolbar in Microsoft Word contains a button with a picture of a floppy disk which saves an open file to the hard disk; users mistakenly thought that the button would save the file to the floppy disk)

20.9 --- graphic or symbol (on object) not meaningful enough (e.g., users may confuse the meaning of up and down arrows on window title bars in Windows 3.1)

20.10 --- sound is confusing, disturbing, distracting, annoying

20.11 --- blinking effect is confusing, distracting, disturbing, or annoying

20.12 --- dissimilar objects look too similar (e.g., an illustration of a button in help text looks deceptively like the button)

Figure 3. Part of UPT Object Appearance web page

The UPT is based on problem characteristics; and, since some characteristics encompass various user interface guidelines, principles, and heuristics, UPT categories contain references to relevant guidelines and/or heuristics when appropriate. For example, problem description 20.1 in Figure 3 refers to the consistency heuristic.

2.4. How the UPT Was Developed

The UPT was developed empirically using an iterative, bottom up approach that involved a detailed, two-phase examination of over 400 usability problems captured in real-world development environments. Initially, we collected 645 usability problem descriptions from five, interactive software development projects. System developers, evaluators, and/or the researchers observed the usability problems during usability evaluation and recorded each problem in prose form. The problem descriptions were brief, usually containing one or two sentences. The amount of contextual information that was provided varied among projects. Some development organizations provided little or no contextual information, others either provided adequate information or permitted the researchers to have access to the evaluators who observed the problems. Evaluation techniques varied from formal user testing sessions to co-discovery learning and expert evaluation. No compensation was given to participating organizations and confidentiality was guaranteed. Although each of the five software systems utilized a graphical user interface with a textual component, they spanned multiple application domains. The intended users had varying levels of expertise (novice to expert) in both computer-related and application-domain knowledge.

In phase 1, we classified 406 of the 645 problems (using the prose descriptions) according to the usability heuristics that were violated (Nielsen, 1993). We examined problems as we received them from the participating organizations rather than “pre-selecting” the problems because we did not want our choices to impact the results. Periodically, during this phase, we paused to evaluate our progress and note any difficulties we had encountered. Eventually, we determined that the heuristics could not be used as the basis for a classification scheme and we concluded this phase.

We identified four difficulties with using the heuristics as the basis for a classification scheme: insufficient distinguishability, lack of mutual exclusiveness, incompleteness, and lack of specificity (Keenan, 1996). A brief explanation of each difficulty is given in Table 2. Although these difficulties precluded our using the heuristics as the basis for a taxonomy, heuristic analysis proved to be a good first step in organizing the data.

We began phase 2 by re-examining the 406 descriptions, one heuristic category at a time. We also examined the group of problems we were unable to classify heuristically. We looked for commonalities and similarities among the descriptions to further group the problems within each category as well as across categories. We looked for characteristics that differentiated descriptions. After examining all heuristically categorized problems, a total of 74 new groupings emerged.

The 74 groupings were likewise examined for commonalities, similarities, and differences. We identified 21 problem types which became the deepest (rightmost) sub-categories in Figure 1. Similarly, the 21 problem types were divided into five clusters which became

Table 2. Four difficulties with a heuristic-based taxonomy.

Difficulty	Explanation
Insufficient distinguishability	Very different problems are classified according to the same heuristic.
Lack of Mutual Exclusiveness	A single problem is classified according to more than one heuristic.
Incompleteness	Some problems cannot be classified according to any heuristic.
Lack of specificity	A single problem is classified according to one or more heuristic but is not adequately captured by those heuristic(s).

the five primary categories. The five clusters were separated into the artifact and task components.

3. UPT Reliability Study

An empirical study was undertaken to show that the UPT can be used to classify usability problems reliably, i.e., a given problem would be classified similarly by different classifiers (Keenan, 1996).

3.1. Methodology

The methodology involved participant selection, problem selection, and protocol.

Participant Selection

In our study, seven participants classified the same set of 20 usability problems without remuneration. Due to the amount of time each participant spent on the study, we were constrained to limit the number of participants to seven highly qualified individuals. The participants had varying levels of experience in industry, government, and academic development environments. Six participants had substantial experience in both user interface and software development; one had experience in software development only. Five of the classifiers used guidelines and heuristics regularly; the remaining two never used guidelines and heuristics. At the outset of the study, three classifiers had limited exposure to the UPT; four had no prior UPT experience.

Problem Selection

Twenty problem descriptions were randomly selected from the 645 problems. Using a set of size 20 was a compromise between two factors: the sample had to be large enough to satisfy criteria for statistical significance at the primary category level, yet small enough to limit the time spent by each classifier so their participation in the reliability study could be assured.

The problems were chosen in the following way:

- Ten problems were selected randomly from the 406 that had been examined during the analysis phase prior to building the UPT.
- The remaining ten were selected randomly from the 239 problems that had not been examined prior to UPT construction.

The decision to use two groups of 10 problems guaranteed that no one project was over-represented and that an equal number of problems were used that had not been previously examined.

Protocol

Classifiers participated at locations and time(s) of their choosing. Each classifier was given a packet of materials containing:

- instructions,
- brief (20-minute) written tutorial on the UPT,
- sample problem classifications,
- Figure 1,
- a glossary of terms, and
- 20 problem descriptions.

Although the researchers often had access to contextual information (as discussed in section 3.1), the problem descriptions given to participants contained minimal additional information. For the participants, contextual information included the type of user interface, type of software system, and a very brief explanation of the task the user was performing at the time the problem occurred (possibly two sentences). There were two reasons for the decision to limit the amount of detail provided to study participants. First, we needed to ensure confidentiality of the products and organizations. Second, we wanted to make sure that our choice of what contextual information to include did not influence their classifications. We should also note that the descriptions themselves were modified only to ensure confidentiality of the products and organizations.

The participants classified each problem in the artifact component and in the task component. Six classifiers used the Web-based tool, one used a paper copy. The classifiers were not monitored and were allowed to classify the problems in any order, revisiting any problems they wished. There was no time limit on the participants. Upon completion, the participants returned the classifications to the researchers.

Table 3. Results of the six hypothesis tests.

Test	Problem Set	Component	κ	Z	p-value	Null Hyp.
1	all 20	artifact	.403	9.776	.000	rejected
2	all 20	task	.095	2.306	0.11	rejected
3	10 examined	artifact	.357	7.336	.000	rejected
4	10 examined	task	.112	2.144	0.16	rejected
5	10 unexamined	artifact	.437	6.442	.000	rejected
6	10 unexamined	task	.068	1.018	.154	not rejected

3.2. *Statistical Analysis and Results*

We performed six hypothesis tests. Each tested the null hypothesis that, for a given problem set and UPT component, the classifications were random versus the alternative hypothesis that, for the same problem set and UPT component, there was more agreement among classifiers than would be obtained by random classification. As shown in Table 3, tests 1 and 2 were performed on all 20 problems, tests 3 and 4 were performed on the 10 previously examined problems, and tests 5 and 6 were performed on the 10 problems not previously examined.

To assess the level of agreement among classifiers, the kappa statistic, κ, was used to analyze classifications (Cohen, 1960). We selected kappa as it is the appropriate tool for measuring agreement with categorical (i.e., non-numeric) data. We used the Fleiss extension of the kappa (Fleiss, 1971), since kappa is technically used for agreement between two subjects and we had seven classifiers.

Kappa is the proportion of agreement after chance agreement is removed from consideration, i.e., the proportion of agreement actually attained beyond chance. The upper limit of κ is 1 and occurs when the proportion of problems in which the classifiers agreed is 1. Kappa is 0 when the proportion of agreement equals what would be predicted by chance.

To test the null hypothesis that the classifications are random ($\kappa = 0$) against the alternative hypothesis that the classifications are not random ($\kappa > 0$), κ is divided by its standard error, which produces a ratio, Z, that is approximately distributed as a standard normal variate.

Kappa was computed at the primary category level only. Analysis at the subcategory level would not have been valid since the small number of observations within each primary category would have invalidated the approximate normality of kappa at the subcategory level.

Table 3 presents the values computed for each hypothesis test. The κ column contains the values for the kappa statistic. The Z column contains the observed values for the standard normal variate. The values in the *p-value* column represent the obtained level of significance, i.e., the likelihood of observing the associated z-value if in fact the null

Table 4. Agreement in artifact classifications.

# Classifiers	10 examined problems	10 unexamined problems	Total artifact agreement
5-7	6	7	13
4	2	2	4

Table 5. Agreement in task classifications.

# Classifiers	10 examined problems	10 unexamined problems	Total task agreement
5-6	3	3	6
4	5	5	10

hypothesis is true (the classifications are truly random). The Null Hypothesis column contains the final outcomes.

In tests 1 through 5, the null hypothesis is rejected. There is sufficient evidence to conclude that there is agreement greater than chance. In test 6, although we cannot reject the null hypothesis at the .05 level of significance, it is important to note that $\kappa = .068$ is positive. This suggests there may be agreement greater than chance in the task component for the 10 previously unexamined problems.

3.3. Further Examination of Raw Data

We also inspected the classification data from a non-statistical perspective. We discovered that the problems on which the classifiers agreed were fairly evenly spread over both (size 10) data sets. In addition, more than half of the classifiers agreed on 17 of the 20 artifact classifications and 16 of the 20 task classifications as shown in Tables 4 and 5, respectively.

Table 4 entries give the number of problems which participants classified in the same primary category in the artifact component. Five or more classifiers agreed on 13 problems as shown in the "5-7" row. Six of the 13 problems were in the examined set, the remaining seven were in the unexamined set. Four classifiers agreed on four additional problems as shown in the "4" row. Two were in the examined set and two were in the unexamined set.

Similarly, the entries in Table 5 are the number of problems which participants classified in the same primary category in the task component. Five or six classifiers agreed on 6 problems as shown in the "5-6" row. Four classifiers agreed on 10 problems as shown in the "4" row. The best agreement occurred among six classifiers, i.e., there were no problems on which all seven classifiers agreed. Agreement was equally distributed over both sets of 10 problems.

Although the sample size precluded a meaningful statistical analysis at the subcategory level, we also wanted to know the proportion of participants who classified a problem in a given subcategory given that they classified that problem in the same primary category. For example, assume that six of seven participants classified problem X as a visualness problem. We wanted to know what proportion of the six participants also classified problem X in the same visualness subcategory.

We anticipated that once classifiers identified a primary category, that there would be good agreement within that category. We inspected the problems classified within primary categories by at least four classifiers. Eighty-two percent of the classifiers who agreed on an artifact primary category also agreed on a subcategory. Likewise, 79% of the classifiers who agreed on a task primary category also agreed on a subcategory. Although we are encouraged by these results, more research is needed to assess the strength of this effect.

3.4. Discussion

We identified several factors that may have contributed to the non-significant result obtained for hypothesis test 6 (see Table 3).

- Poor problem description quality.

  Classifiers commented that many of the problem descriptions were vague, incomplete, and contained irrelevant information. One classifier commented that the vagueness of the descriptions prevented a thorough exploration of the task component. Several remarked that individual descriptions often contained multiple problems. One participant explained that it was difficult to decide which problem to classify and which one(s) to ignore.

- Descriptions lack sufficient contextual information.

  Participants were unable to ask for additional information during the study and indicated that they often ignored certain words or phrases and relied on their own experience and expertise to make assumptions about the problems and the context in which those problems occurred. They further indicated that the difficulty they experienced during problem interpretation impacted their responses as they felt less sure of the appropriate classification.

- Insufficient classifier experience with application domains.

  The 20 problems classified by participants were in application domains with which they had varying levels of experience. In a real-world development environment, classifiers should be sufficiently familiar with the application domain.

- Unfamiliarity with the software systems.

  Participants commented that their unfamiliarity with the systems made classification (especially in the task component) difficult. One classifier remarked that he/she did not have a full understanding of the user interface nor possible user tasks. That classifier also indicated that if he/she used the UPT on the job, that there would be a richer population (of classifications) within the task component.

- Lack of UPT training.

  The UPT is a novel approach to thinking about usability problems, and we anticipate that increased familiarity will improve reliability. We noted during a pilot study that participants became more comfortable with the taxonomy as the study progressed. This observation was confirmed when reliability study participants commented that they felt much more comfortable with the UPT after classifying several problems.

The significance of the reliability results are underscored by the fact that they were obtained in spite of these factors, that individually or in combination could have impacted the level of significance achieved in any of the six hypothesis tests. We believe that the reliability results would have been even stronger if the UPT had been tested in a development environment by participants who knew the software being evaluated, knew the application domain, had a good understanding of relevant contextual information, and were experienced in using the UPT. This observation is significant because it implies that greater reliability might be expected in practical use than was observed in the statistical study. In a realistic environment, we would expect that the evaluators could have received training in the use of the UPT and would either have sufficient knowledge of the software, the domain, and the context or that this information was readily available.

4. Problem Analysis

UPT classification can contribute significantly to problem analysis in two ways. First, UPT classification can be used to identify specific characteristics of usability data sets (discussed in section 4.1). Second, it can be used to complement, or extend, current analysis strategies (discussed in section 4.2). To illustrate these contributions, we use UPT classification of a set of ninety-one usability problems. The problems were identified during evaluation of a partially functioning prototype of a computer-aided software engineering (CASE) tool. The CASE tool supports system specification, checks specification consistency and completeness, and simulates system behavior. We conclude our discussion with observations regarding potential UPT impact on problem prioritization in section 4.3.

4.1. Identifying Characteristics Of Usability Data Sets

UPT classification and analysis enable developers to identify specific characteristics of large sets of usability problems. We first outline how UPT analysis supports examination of entire sets of usability problems. Next, we discuss how UPT analysis can guide the identification of global problems. Last, we consider how UPT analysis provides an alternative method for identifying outlier problems.

4.1.1. The UPT Supports Examination Of Entire Sets Of Usability Problems

Examining an entire set of usability problems enables the development team to make recommendations that can be widely applied, replace individual local solutions with different (possibly global) solutions, ensure that problem reports do not contradict each other, and identify any trade-offs that may exist (Jeffries, 1994). There are three aspects of UPT analysis that help developers to examine data sets more easily. First, the UPT provides a new perspective on what a problem type is. Second, UPT classification offers a new approach for identifying problem clusters. Third, the hierarchical structure of the UPT offers a new viewpoint on examining problem sets at varying levels of abstraction.

Aspect 1. New Perspective on Problem Types

Perhaps the most obvious contribution of the UPT is that it provides a new perspective on what a “problem type” is. Recall that UPT categories are based on problem characteristics. This perspective differs significantly from current definitions of “problem type” which include categories of human error such as the user’s inability to find the appropriate action, choosing the wrong action, and choosing the correct action in the wrong context (Mack and Montaniz, 1994; Senders and Moray, 1991). Jeffries discusses problem categorization with respect to core versus taste problems where a core problem is a problem report that most usability specialists would agree is a problem and a taste problem is a problem report that specialists might disagree about its validity (Jeffries, 1994).

Aspect 2. New Approach for Identifying Problem Clusters

UPT classification enables developers to identify problem clusters, i.e., problems classified within individual UPT categories. This approach contrasts with current clustering techniques that focus on problem attributes such as severity, impact, and cost-to-fix (Dumas and Redish, 1994; Neilsen, 1994; Jeffries, 1994). As shown in the CASE tool example below, obtaining distributions of problems across UPT categories enables developers/evaluators to identify what kinds of problems were encountered most often. A high percentage category in a UPT distribution is either due to user(s) encountering many similar, but different, problems or to many users experiencing the same problem (high frequency). Consequently, developers can use UPT analysis to assess problem scope and frequency as well as to direct attention to types of problems for which both global and local problem solutions may be considered.

Consider Figures 4a and 4b that show problem clusters for the CASE tool data set at the primary category level. As shown in Figure 4a, 30% of the artifact classifications were in the visualness category, 23% were in the language category, 11% were in the manipulation category, and 36% received an NC classification.

In the task component, 22% of the problems were classified in the task-mapping category, 26% of the problems were classified in the task-facilitation category, and 52% received an NC classification.

Using these distributions, CASE tool developers decided to focus on the visualness and language clusters in the artifact component as well as the task-mapping and task-facilitation clusters in the task component. The visualness and language clusters were chosen because they are high percentage categories. Developers were unable to decide which task cluster

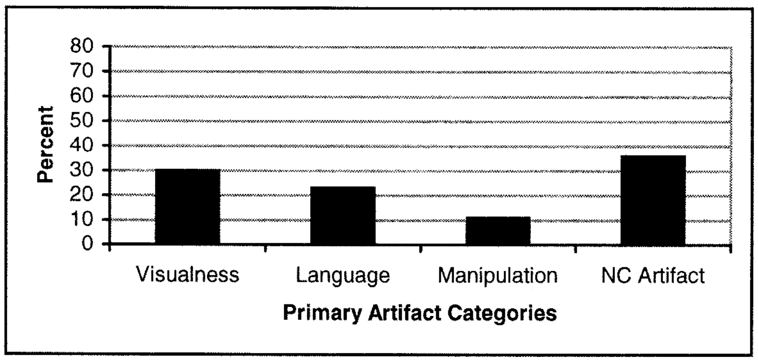


Figure 4a. Distribution of artifact classifications.

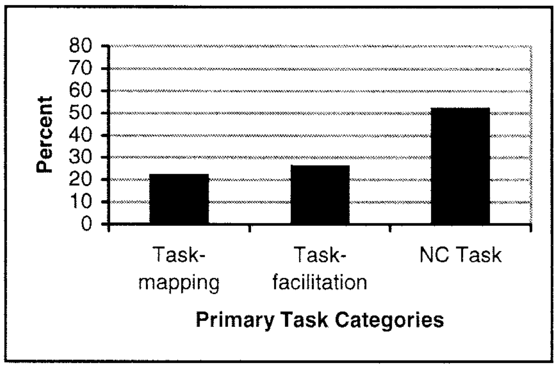


Figure 4b. Distribution of task classifications.

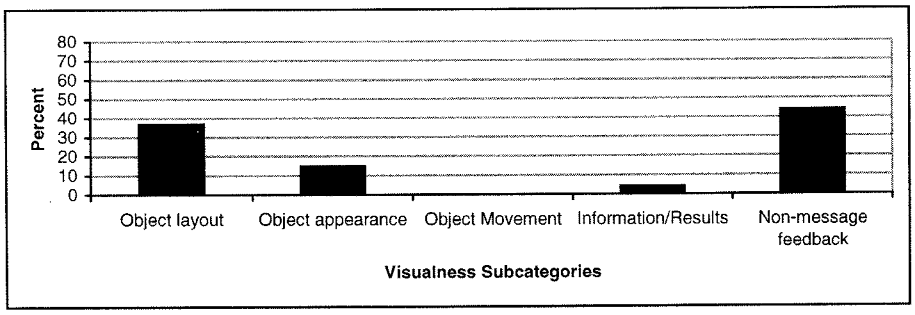


Figure 4c. Distribution over visualness subcategories.

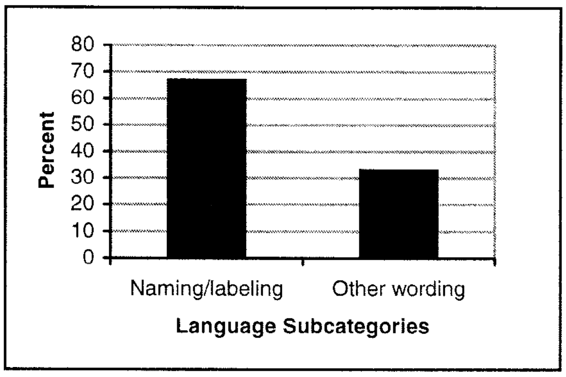


Figure 4d. Distribution over language subcategories.

to investigate since the problems were fairly evenly distributed. As a result, they decided to examine both task clusters. To investigate these clusters more fully, similar distributions were tabulated at the subcategory level as discussed below.

Interesting results for the CASE tool emerged from high percentage artifact categories. Figure 4c illustrates that most of the problems classified in the visualness category were due to object layout (37%) or non-message feedback (44%). Although most of the language problems were due to naming and labeling, as shown in Figure 4d, developers also looked closely at the other wording problem cluster. The distribution of problems classified as other wording, shown in Figure 4e, illustrates that these problems were primarily about feedback messages and error messages.

The high percentage of visualness problems in the object layout and object appearance categories may indicate that developers need to re-examine how user tasks are laid out on the system. The high percentage of problems in the language category could signal the need for increased familiarity with words commonly used in the application domain. The

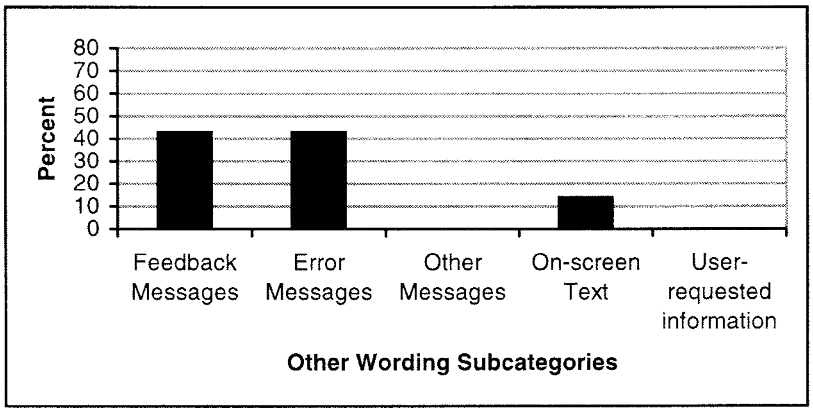


Figure 4e. Distribution over other wording subcategories.

percentage of naming/labeling problems could indicate that the responsibility for designing labels and writing messages should be given to team members who are more precise in their use of language. In the CASE tool example, two developers admitted that although they were familiar with the application domain, they had difficulty in choosing labels appropriate for specific user interface objects. The developers were also surprised at the high percentage of problems related to feedback (44% of the visualness problems were about non-message feedback; 43% of the other wording problems were about feedback messages). The developers did have some expertise and training in user interface development and felt that they had designed appropriate feedback into the interface. However, the number of feedback problems was a signal that additional effort was needed to correct current feedback problems and to avoid these issues in later designs. Additionally, since 86% of the other wording problems were due to feedback or error messages, developers decided that more effort needed to be placed on messages.

Equally interesting results were obtained among high percentage task categories. Recall that Figure 4b showed that the task classifications for the CASE tool were spread fairly evenly at the primary category level. Figures 4f and 4g show that most of the task-mapping problems were either about the interaction or system functionality; and, that task-facilitation problems were primarily about system automation.

Given that the CASE tool was an early prototype, the high percentage of functionality problems is not surprising. However, the problems classified in the functionality category could be used to check the requirements document to determine if additional functions are needed. The high percentage of interaction problems coupled with the system automation problems could indicate the developers need to re-examine how user tasks are mapped to the system and explore which parts of those tasks could be automated. Also of interest is the hypothesis that the large number of interaction classifications may be related to the object layout classifications. Developers could crosscheck the problems classified in the interaction subcategory in the task component and determine if a relationship exists among

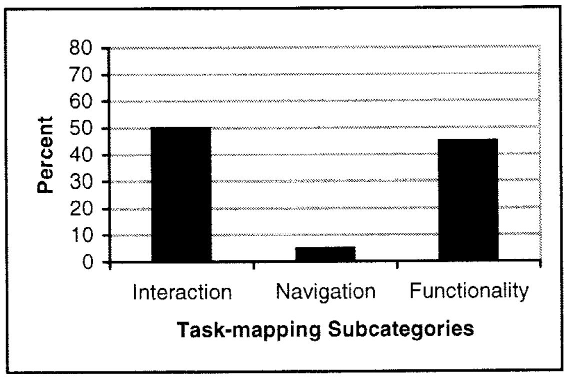


Figure 4f. Distribution over task-mapping subcategories.

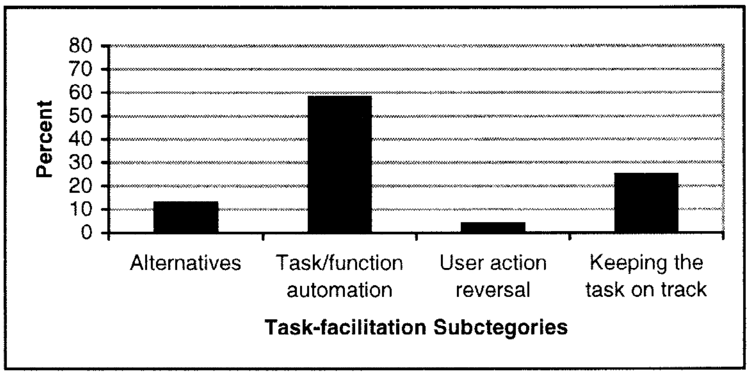


Figure 4g. Distribution over task-facilitation subcategories.

these problems and the problems classified in the object layout subcategory in the artifact component. This is discussed further in aspect 3 below.

Aspect 3. New Viewpoint on Levels of Abstraction

The hierarchical structure of the UPT enables developers to examine an entire set of problems at varying levels of abstraction. This extends Jeffries recommendations (Jeffries, 1994) which focus on examining individual problem reports at different levels of abstraction. The UPT primary category level provides the highest level of abstraction. The deepest subcategories correspond to the lowest level of abstraction. We believe that an examination of problems at various levels of abstraction across components will enable trained individuals to identify potential trade-offs and contradictions for a set of usability problems and to pursue questions that relate problems and solutions across multiple categories.

For example, assume the CASE tool developer explores problems classified in the visualness category and/or those classified in the task-mapping category more deeply. He/she tabulates the distribution of visualness problems as shown in Figure 4c and the distribution of task-mapping problems in Figure 4f. The developer then examines problems classified in the object layout and/or object appearance subcategories of Figure 4c and compares them with problems classified in the interaction subcategory of Figure 4f. The developer could investigate the following questions:

- Which (and how many) problems received an artifact classification of object layout or object appearance and a task classification of interaction?
- Which (and how many) problems received an artifact classification of object layout or object appearance and a task classification other than interaction?
- Would correcting specific interaction problems also address specific object layout and object appearance problems?
- Which (and how many) problems received a task classification of interaction and an artifact classification other than object layout or object appearance?
- Do any interaction classifications contradict object layout or object appearance classifications?
- Do the object layout (or object appearance) problems result from a failure to apply usability guidelines and heuristics to visual design?
- Do the interaction problems result from a poor understanding of the user tasks?
- If developers do have a thorough understanding of user tasks, are the interaction problems due to poor design, i.e., a failure to capture the essential structure and sequence of each task? If the essential structure of each task has been captured in the design, do these problems indicate the need to "polish the presentation," i.e., make the system more transparent to the user?

4.1.2. The UPT Guides Identification Of Global Problems

One important outcome of the three aspects discussed in section 4.1.1 is that UPT analysis can guide the identification of global problems. Global problems have wider scope (they span multiple screens or locations in the dialogue) and have many symptoms, some of which may not exposed during a usability test (Dumas and Redish, 1994). To identify global problems for the CASE tool example, we examined the problems in each UPT subcategory recalling specific contextual information about each problem such as the screen on which a problem occurred and the user task that was being performed. We found several global problems which we describe in Table 6. Problems 1-5 resulted from examining artifact classifications; problems 6-7 arose from the task classifications. Note that general terms

Table 6. Global problems for CAE tool example.

#	Subcategory	Global Problem
1	Object layout	Labels for fields and columns are inappropriately placed, i.e., they are too far away from the associated user interface object.
2	Non-message feedback	Non-message feedback is either uninformative, inconsistently provided, or absent from the user interface.
3	Naming	Many names (labels) are confusing and not representative of items needed in the current user task.
4	Other wording	Messages are either not helpful or absent from the user interface.
5	Cognitive aspects	Visual cues are inconsistent, misleading, or absent.
6	Interaction	By insisting that users use dialogue boxes to edit table cells rather than allowing them to type directly in the cells, many unnecessary steps have been added to several fundamental user tasks.
7	Task/function automation	Parts of many user tasks could have been automated but are not.

are used in the global problem descriptions to ensure the confidentiality of the development organization.

4.1.3. The UPT Provides An Alternative Method For Identifying Outlier Problems

Another consequence of the aspects discussed in section 4.1.1 is that UPT provides an alternative method for identifying outlier problems. In the context of usability data collection and analysis, the term "outlier" refers to an infrequent usability problem with an unusual performance metric, i.e., a performance score for one or more individuals that is much larger or much smaller than the average score (Dumas and Redish, 1994). In the context of the UPT, we use the term "outliers" to refer to problems in low percentage categories, i.e., problems that were encountered less frequently. Dumas and Redish recommend that outliers be examined to determine if they are likely to be representative of a large segment of the user population. Although UPT classification does not capture performance data, UPT analysis does identify problems that occur less frequently.

For example, we examined the problems in low percentage subcategories for the CASE tool. We found three problems that would be encountered repeatedly by many users, and therefore, should be addressed. All three emerged from low percentage task subcategories. See Table 7 for the three problem descriptions and the subcategories in which they were identified.

Table 7. Outlier problems for CASE tool example.

Subcategory	Outlier Problem
Navigation	There is no way for the user to advance quickly from one table cell to the next cell or from one table row to the next row. The user must remove his/her hand from the keyboard and use the mouse to advance.
User action reversal	There is no undo feature.
Keeping the task on track	Users can close a specification without being prompted to save changes.

4.2. Complementing Current Analysis Strategies

In this section, we examine how UPT analysis complements four commonly used analysis techniques. In particular, we discuss the advantages of identifying relationships among UPT classifications and user tasks, other usability data such as cost-to-fix and performance measures, and responses from user satisfaction questionnaires. We also consider how UPT analysis can be used to re-organize problems according to the usability heuristic(s) they violate. Although other techniques may be used to analyze usability data, we believe that the four discussed below are representative of the types of analysis currently performed.

Look for Relationships Among UPT Classifications and User Tasks

We believe that organizing UPT classified problems according to user task will reveal much about a set of usability problems. To accomplish this, associate usability problems and their classifications with individual user tasks. For example, several analyses are possible:

- Examine UPT classifications associated with each task. If, for example, most of the problems users experienced while performing task A were about naming, system developers should re-examine the language used in screens for that task. If classifications associated with task B are distributed across many UPT categories, it could mean that the developer's understanding of the task, and how to capture that task on the system, should be re-examined. It also suggests that many different problems may be addressed by carefully designing problem solutions. In the CASE tool example, if specific interaction problems and object layout problems are associated with the same task, developers could address the usability problems in both categories by redesigning the user task on the software system.

- Identify which user tasks are associated with individual UPT categories. If many tasks are associated with one category (such as non-message feedback), it could indicate the existence of global problems.

Look for Relationships Among UPT Classifications and Other Usability Data

Relevant usability data include whether or not the problems are core problems (i.e., problems related to the part of the user interface evaluated in depth (Nielsen, 1994), performance measures, problem severity (impact on the user), and cost-to-fix. Associating these data with individual UPT categories can then be used to augment prioritization strategies as we discuss in section 4.3.

Look for Relationships Among UPT Classifications and Results of User Satisfaction Questionnaires

User satisfaction questionnaires such as QUIS (Chin and Norman, 1988) and SUMI (Porteous et al., 1993) assess how users feel about various aspects of a user interface. Questionnaire results can be compared to the percentage of problems in individual UPT categories to augment UPT analysis. For example, consider a specific QUIS satisfaction scale question that asks subjects to rate (on a scale of 1 to 9) whether messages that appear on screen are confusing or clear. If questionnaire responses indicate confusion and the percentage of usability problems in those three message subcategories is low, then developers can use user satisfaction results to identify outlier problems that should be corrected. If questionnaire responses indicate little difficulty and the percentage of usability problems in those three subcategories is high, developers could, first, re-examine the problem descriptions classified in those subcategories and, second, look for differences among QUIS subjects and those involved in user testing that might explain the anomaly. Two additional scenarios will be examined in section 4.3 in the context of problem prioritization.

Organize Problems According to Specific Usability Heuristics

UPT categories contain references to relevant guidelines and/or heuristics when appropriate. As a result, problems that violate specific heuristics, such as consistency, can be identified. We identified 11 consistency problems in the CASE tool data set: nine artifact problems and two task problems. No problems were determined to be about consistency in both components. While CASE tool developers were not overly concerned about the 11 problems (considering the other types of problems that had to be addressed), they were interested to find that of the 11 artifact problems, four were about inconsistent naming and labeling and three were about inconsistent object layout. They agreed that the naming problems could be easily addressed and decided to postpone correction of the other consistency problems.

4.3. Problem Prioritization

Current prioritization analysis is based on calculating and ranking individual usability problems with respect to importance and cost-to-fix (Hix and Hartson, 1993). Although prioritization techniques may vary among organizations, importance is generally assessed using criteria such as scope, frequency, impact, and severity (Dumas and Redish, 1994; Mack and Montaniz, 1994). We have not formally studied the contribution UPT classification may make to problem prioritization; however, it is clear that the UPT adds the ability to augment that analysis with global or aggregate measures, i.e., metrics reflecting a large number of classified usability problems. We briefly outline several such possibilities below.

- Prioritize problems according to high percentage UPT categories. This approach could be done at any level in the UPT, i.e., from the primary category level to deepest subcategory level.
- Prioritize problems based on UPT distributions and severity. Calculate the average severity for each UPT category. Problems in UPT categories with a high average severity rating should be closely examined to identify specific problems that need immediate correction.
- Prioritize problems based on UPT analysis and cost-to-fix. Cost-to-fix can be approached two ways: use the average cost-to-fix for each UPT category and/or the total cost-to-fix for each category. Problems in low percentage categories with a small average, or small total, cost-to-fix could be easily addressed. High percentage categories with a high average, or high total, cost-to-fix need to be carefully examined to ensure that problem corrections are scheduled for the appropriate release.
- Prioritize problems based on UPT analysis and results from user satisfaction questionnaires. Recall the QUIS question presented in section 4.2 about on screen message clarity. If questionnaire responses indicate confusion and the percentage of usability problems in the feedback, error, and other system messages UPT subcategories is high, then developers should carefully examined those problems during the prioritization process. If questionnaire responses indicate little difficulty with messages and the percentage of usability problems in the three UPT subcategories is low, developers could decide to postpone further consideration of those problems until a later time.
- Prioritize problems based on a combination of cost-to-fix, importance, and UPT analysis. Current recommendations involve rating individual problems with respect to cost-to-fix and importance using a scale of low, medium, and high. Individual problems are then placed in the appropriate cell in a cost/importance table such as the one shown in Table 8 (Hix and Hartson, 1993; Rubin, 1994).

  We suggest that average problem importance and average cost-to-fix be assessed for problems in each UPT category. Individual UPT categories could then be placed in the appropriate cell in an average cost/importance table such as Table 9. Table 9 contains a hypothetical example that uses only a few UPT categories. Note that each cell contains one or more UPT categories and information about whether those categories contained a high percentage (H%) or low percentage (L%) of problems in the original distribution. A similar approach could be taken with average problem importance and total cost-to-fix for each category.

Although more research is needed to validate these approaches, we believe that the addition of the UPT framework will contribute significantly to problem analysis and prioritization.

Table 8. Cost/importance table for individual usability problems.

		PROBLEM	Importance	
		Low	Med	High
COST	Low			
TO	Med			
FIX	High			

Table 9. Average cost/importance table for UPT categories.

		AVERAGE	PROBLEM	IMPORTANCE
		Low	Med	High
AVERAGE	Low	Object appearance (L%)		
COST	Med		Object layout (H%)	Non-message feedback (H%)
TO				Visual Cues (L%)
FIX	High		Task/function automation (H%)	Interaction (H%)

5. Field Usage Reports

Two developers have used the UPT in their work environments. Developer 1 used the UPT several times a week and implemented a usability problem tracking and analysis tool incorporating UPT categories. Developer 2 used the UPT to analyze several sets of usability problems prior to her current position and has used the UPT once in her current work environment. We provide a brief background on each developer and summarize their remarks. We conclude with a short discussion of our own experiences.

Developer 1 has performed usability-related tasks for 1-2 years. She championed usability in her organization and convinced management to start usability testing. She designed user interfaces, recruited usability subjects, supervised usability lab personnel, planned and observed usability sessions, analyzed the resulting notes and debriefed observers, established priorities, and crafted possible solutions for identified problems. Developer 1 acquired usability expertise by reading books and attending a seminar on user-centered design and usability testing.

Initially, developer 1 had minimal exposure to the UPT and used online documentation to familiarize herself with UPT structure. She commented that the UPT was easy-to-learn, easy-to-use, and that classification became much easier (and faster) after she had classified over 20 problems. She attributed this to the fact that she had internalized much of the UPT structure and relied less on individual Web pages.

Developer 1 commented that the UPT was both useful and practical. Specifically, UPT classification helped her to:

- identify exactly what the problem was about,
- improve problem reporting,
- analyze problems from various perspectives,
- assess problem severity and frequency, and
- prioritize problems prior to correction.

She noted that, in many cases, the categorization process helped suggest solutions. In addition, UPT classification made her "... more confident and objective in her evaluations ... and ... grouped problems in a way that reflected the state of the product, which was an interesting effect. Defining the problems made them seem easier to solve and less mystifying. ... The UPT helps to bring a science to what she has always felt is an art in interpreting usability testing results."

Developer 2 has worked as a human factors engineer on a usability team for two years. She writes requirements specifications, designs parts of the user interface, and evaluates the level of usability achieved in the software. Developer 2 had some formal training in usability and acquired additional expertise by participating in usability consulting projects, attending conferences, and reading current usability literature.

Developer 2 had extensive experience with the UPT prior to using it in her work. Initially, she used online documentation, but commented that as she became more familiar with the UPT categories, she used the Fast Page to go directly to the desired category instead of traversing the entire set of UPT Web pages. She noted that for many problems, classification was instinctive, i.e., she did not need to use the Web tool at all.

Developer 2 indicated that the primary advantage of the UPT was improved problem reporting and description. She observed that "classification ... often suggested the solution (to a problem)" and that an examination of the high percentage categories indicated when a "higher level approach (was needed) ... perhaps indicating a return to the operational concept." In addition, she reported that the state of the product impacted problem identification, i.e., when little of the supporting functionality is implemented in a prototype, task problems are harder to identify. As a result, she noted that problems detected on early prototypes tended to be classified primarily in the artifact component.

Our own experiences with the UPT are similar to that reported by the two developers. First, we do not need to use the Web tool to classify a problem. Classification follows immediately after observation. Second, the UPT perspective has improved our problem reporting. Specifically our problem descriptions are much more:

- clear (unambiguous),
- precise (contain only one problem),

- comprehensive (contain both artifact and task information when appropriate), and
- problem-centered (information about the user is clearly distinguished from information about the problem).

More research is needed to show that using the UPT will, in fact, improve problem reporting in real-world development environments.

6. Conclusions and Future Work

The UPT is an overall framework in which large sets of usability problems detected on graphical user interfaces with a textual component can be reliably classified and then analyzed. Classified problems are grouped according to type (UPT category). UPT problem analysis enables developers to examine data sets from both a detailed and a global perspective, and thus, extends and complements current problem analysis strategies. The UPT is easy to learn and to use for individuals familiar with commonly used terminology, i.e., individuals with some background in usability.

We expect that developers and evaluators with varying backgrounds, expertise, and roles will use the UPT. One possibility is that the UPT will be used by a trained usability evaluator: an experienced individual who is part of a usability engineering team. In this scenario, classification may be based on direct observations by that individual or on reports received from others. A second scenario would be similar to those presented in section 5, i.e., the UPT would be used by a developer with novice usability expertise, who is possibly responsible for some aspects of user interface design as well as evaluation. In this scenario, classification would be based on direct observations. A third scenario would be for the classifier to classify problems using reports received from others. In this scenario, reports could be generated from formal and informal user testing or from problems reported from the field.

The degree of familiarity with the UPT will impact whether or not classification can be done in real time, i.e., as problems are observed. For usability engineers less comfortable with the UPT, the fast pace of events during user testing often precludes immediate classification. To minimize the loss of contextual information, it is desirable to do classification as soon as possible after observation. However, we have found that classification is beneficial long after the reports are generated as long as the reports contain sufficient information for the problems to be partially or fully classified, or the classifiers have access to individuals who recall the critical incidents.

We envision many research projects that extend this work, among which are the following:

- As discussed in section 3.4, we could not use the kappa statistic to assess the level of agreement at the subcategory level. We anticipate a new study in which the number of observations at the subcategory level is sufficient for a statistical analysis.
- Since the UPT was developed from data sets on single-user, graphical user interfaces with textual components, we imagine that UPT categories will need to be extended to accommodate usability problems detected on systems that utilize leading-edge user

interface technologies. Initially, we plan to explore three types of environments: immersive, collaborative, and the WWW.

- We plan to investigate the impact of the UPT on problem reporting. One area of particular interest is whether UPT classification helps developers/evaluators identify the exact nature of a problem.

- We anticipate that UPT analysis may lead to or suggest (possibly global) problem solutions. Although we currently have little anecdotal evidence to support this hypothesis, our conjecture is supported by comments made by the two developers who used the UPT in the workplace.

- We intend to investigate potential associations among UPT categories and other problem characteristics such as severity and frequency. We believe that a more thorough investigation of this aspect of problem analysis will contribute significantly to problem prioritization strategies.

In addition, we also plan to improve the Web implementation of the UPT. We are focusing on enhancements to tool usability and online documentation.

Acknowledgments

Thanks go to Lois E. Mitchell (currently of SilverStream Software, Inc. who participated while at Proactive Solutions, Inc.) and to Karen C. Bowen (of The MITRE Corporation) for using the UPT in the workplace and taking the time to report their experiences. Many thanks go to Michael A. Keenan for his contributions during data analysis and also final editing. We would like to express our appreciation to the Graduate Student Assembly of Virginia Tech for a grant in support of one phase of this research effort.

References

Card, D. N. 1993. Defect-causal analysis drives down error rates. *IEEE Software* 10.4: 98–99.

Card, D. N. 1998. Learning from our mistakes with defect causal analysis. *IEEE Software* 15.1: 56–63.

Carroll, J. M., Kellogg, W. A., and Rosson, M. B. 1991. The task-artifact cycle. *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge: Cambridge University Press, 74–102.

Chin, J. P., Diehl, V. A., and Norman K. L. 1988. Development of an instrument measuring user satisfaction of the human-computer interface. *Proc. ACM CHI '88 Conference*. Washington, D.C., 213–218.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement XX* 1: 37–46.

Desurvire, H. W. 1994. Faster, cheaper!! Are usability inspection methods as effective as empirical testing?. In J. Nielsen and R. L. Mack (eds.), *Usability Inspection Methods*. New York: John Wiley & Sons, Inc., 173–202.

Dumas, J. S., and Redish, J. C. 1994. *A Practical Guide to Usability Testing*. Norwood, NJ: Ablex Publishing Company.

Fleiss, J. L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76. 5: 378–382.

Giblin, D. 1992. Defect causal analysis: A report from the field. *Proceedings of the ASQC Second International Conference on Sofware Quality*, 1–5.

Grady, R. B. 1994. Successfully applying software metrics. *IEEE Computer* 27. 9: 18–25.

Hix, D., and Hartson, H. R. 1993. *Developing User Interfaces Ensuring Usability Through Predate & Process*. New York: John Wiley & Sons, Inc.

Jeffries, R. 1994. Usability problem reports: Helping evaluators communicate effectively with developers. In J. Nielsen and R. L. Mack (eds.), *Usability Inspection Methods*. New York: John Wiley & Sons, Inc, 273–294.

Karat, C., Campbell, R., and Fiegel, T. 1992. Comparison of empirical testing and walkthrough methods in user interface evaluation. *Human Factors in Computing Systems. CHI '92 Conference Proceedings*. Monterey, California, 397–404.

Keenan, S. L. 1996. *Product Usability and Process Improvement Based on Usability Problem Classification*. Ph.D. Dissertation. Department of Computer Science. Virginia Polytechnic Institute and State University.

Mack, R., and Montaniz, F. 1994. Observing, predicting, and analyzing usability problems. In J. Nielsen and R. L. Mack (eds.), *Usability Inspection Methods*. New York: John Wiley & Sons, 295–339.

Muller, M. J., Dayton, T., and Root, R. 1993. Comparing studies that compare usability assessment methods: An unsuccessful search for stable criteria. *Proceedings of INTERCHI Conference on Human Factors in Computing Systems (Adjunct)*. New York: Amsterdam, 185–186.

Myers, Glenford, J. 1979. *The Art of Software Testing*. New York: John Wiley & Sons.

Nielsen, J. 1994. Heuristic evaluation. In J. Nielsen and R. L. Mack (eds.), *Usability Inspection Methods*. New York: John Wiley & Sons, 25–62.

Nielsen, J. 1992. Finding usability problems through heuristic evaluation. *Human Factors in Computing Systems. CHI '92 Conference Proceedings*. Monterey, California, 373–380.

Nielsen, J., and Molich, R. 1990. Heuristic evaluation of user interfaces. *Proceedings ACM CHI'90 Conference*. Seattle, Washington, 249–256.

Nielsen, J. 1993. *Usability Engineering*. San Diego, California: Academic Press, Inc.

Ostrand, T. J., and Weyuker, E. J. Collecting and categorizing software error data in an industrial environment. *The Journal of Systems and Software* 4: 289–300.

Paternò, F., and Mancini, C. 1999. Engineering the design of usable hypermedia. *Empirical Software Engineering: An International Journal*, Special Issue on Usability Engineering 4.1: 11–42.

Pfleeger, S. L. 1991. *Software Engineering The Production of Quality Software*. New York: Macmillan Publishing Company.

Porteous, M. A., Kirakowski, J., and Corbett, M. 1993. *Software Usability Measurement Inventory Handbook*. Cork, Ireland: Human Factors Research Group, University College.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. 1995. *Human-Computer Interaction*. Reading, MA: Addison-Wesley Publishing Company.

Pressman, R. S. 1997. *Sotware Engineering A Practitioner's Approach Fourth Edition*. New York: McGraw-Hill, Inc.

Rubin, J. 1994. *Handbook of Usability Testing*. New York: John Wiley & Sons, Inc.

Shneiderman, B. 1998. *Designing the User Interface: Strategies for Effective Human-Computer Interaction, Third Edition*. Reading, MA: Addison-Wesley.

Senders, J. W., and Moray, N. P. (Eds.). 1991. *Human Error: Cause, Prediction, and Reduction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

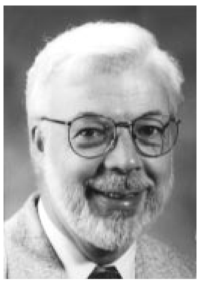
Sommerville, I. 1989. *Software Engineering*. New York: Addison-Wesley Publishing Company.

Vora, P. 1995. Classifying user errors in human-computer interactive tasks. *Common Ground*. Usability Professional Association. Vol. 5. No. 2. 15.

Weiss, D. M. 1979. Evaluating software development by error analysis: The data from the architecture research facility. *The Journal of Systems and Software* 1. 57–70.

Zhang, Z., Basili, V., and Shneiderman, B. 1999. Perspective-based usability inspection: An empirical validation of efficacy. *Empirical Software Engineering: An International Journal*, Special Issue on Usability Engineering 4.1: 43–70.

Susan L. Keenan's current research interests span human computer interaction, software engineering and Web application development. At this time, she is involved in projects that focus on user interface process improvement, techniques for Web development, and tool assessment. She has degrees in mathematics from Christopher Newport University and The College of William and Mary, and in computer science from Shippensburg University and Virginia Tech. She has worked as a consultant to several software development organizations on usability-related product and process improvement and as an Assistant Professor at Columbus State University in Columbus, Georgia. She now resides in Shrewsbury, Massachusetts.



H. Rex Hartson is Professor of Computer Science at Virginia Tech, Blacksburg, VA, where he started working in HCI at Virginia Tech in 1979, and where his current research interests include usability methods and tools and interaction design development methodology. He has degrees from The University of Michigan and The Ohio State University. He has industrial experience, working as a software engineer at The Ohio College Library Center, Columbus, Ohio and as an engineer and researcher at Xerox Corp., Webster, NY. He has been an HCI consultant to business, industry, and government. Hartson is co-author of Developing User Interfaces: Ensuring Usability Through Product and Process, Wiley & Sons, 1993, was series editor of Advances in Human-Computer Interaction, Volumes 1–4, Ablex Pub. Co., and is author of numerous papers in journals, conference proceedings, and edited books.

photo
not
available
at time of
print

Dr. Dennis Kafura is a Professor and Computer Science and Department Head at Virginia Tech. He received his Ph.D. degree from Purdue University in 1974 and joined the faculty at Virginia Tech in 1981. In software engineering he has worked on the development and validation of sortware metrics applied to real world data. His more recent interest in object-oriented programming has included the study of metrics and measures related to object-oriented programs and the use of object oriented techniques in concurrent, distributed and parallel systems. He is the author of the book “Object-Oriented Software Design and Construction with C++” published by Prentice-Hall in 1998. A Java version of this book is forthcoming. He can be contacted at kafura@cs.vt.edu.

Robert S. Schulman is an Associate Professor in the Department of Statistics at Virginia Tech in Blacksburg, Virginia, where he has been employed since 1974. He specializes in statistical instruction and consulting in a wide variety of fields of application. He has a BS degree in mathematics from Carnegie-Mellon University, a MA in Educational Measurement from the University of Maryland, and a PhD in Psychometrics from the University of North Carolina. He has provided statistical training and consultation in industry, government, and academia for a quarter of a century. His book, “Statistics in Plain English” is widely used as a textbook and reference in connection with professional development courses.