PRODUCT USABILITY AND PROCESS IMPROVEMENT BASED ON USABILITY PROBLEM CLASSIFICATION

by

Susan Lynn Keenan

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

APPROVED:

August, 1996

Blacksburg, Virginia

Keywords: Usability, Problem Classification, User-Interface Process Improvement

C. 2

LD
5655
V856
1996
K446
c. 2

PRODUCT USABILITY AND PROCESS IMPROVEMENT BASED ON USABILITY PROBLEM CLASSIFICATION

by

Susan Lynn Keenan

H. Rex Hartson and Dennis G. Kafura, Co-chairs

Department of Computer Science

(ABSTRACT)

Although research and practice have shown that the success of a usability engineering program depends on the identification and correction of usability problems, these problems remain an underutilized source of information. Insufficient guidance regarding the capture of usability problem data results in the loss of information during the problem reporting phase as problem reports are often vague, imprecise, and incomplete. In addition, the absence of a framework for understanding, comparing, categorizing, and analyzing those problems, and their relationship to development context, not only constrains product improvement, but hampers efforts to improve the user interface development process.

A new taxonomic model (the Usability Problem Taxonomy) is presented which contributes to both product and process improvement. The Usability Problem Taxonomy (UPT) is used to classify and organize usability problems detected on interactive software development projects. Individual UPT categories are associated with two aspects of development context: developer roles and skills, and development activities, methods, and techniques.

Two studies were conducted during the course of this research. The first study showed that the UPT can be used to classify usability problems reliably. Findings indicated that level of agreement among classifiers (beyond chance agreement) was statistically significant. Findings in the second study led to the identification of roles and activities that address individual UPT categories as well as those that do not.

Procedures for using the UPT in both product and process improvement are outlined. Examples are presented that illustrate how the UPT can be used to generate higher quality problem descriptions and to group those problem descriptions prior to prioritization and correction. In addition, steps that guide developers in diagnosing weaknesses in the current user interface development process are enumerated. Possible improvement strategies are presented that focus on the selection of specific development activities and team members appropriate for a given project.

DEDICATION

This research is dedicated to four people, without whose help I would not have graduated. To my husband, Dr. Mike Keenan, whose unfailing support kept me plugging away. To my parents, Mrs. Dolores Crispen and Dr. Wayne Crispen, who insisted all along that I would finish. And, to my dear friend, Ms. Karen Bowen, who generously contributed her time, expertise, and equipment during my research.

ACKNOWLEDGMENTS

I would like to thank Cindy Starks Hopeman, Tim Kilby, Jack Repenning, and Paul Sawyer who volunteered their time to critique the data collection forms used to collect data prior to the heuristic analysis. Thanks are due to the five participating, anonymous software development organizations for their willingness to provide anecdotal and quantitative information, usability problem reports and descriptions, videotapes, as well as copies of the interactive-software products. I also would like to thank a group of 11 developers and academics for participating in the study without remuneration. Thanks are also due to my two advisors, Dr. Dennis G. Kafura and Dr. H. Rex Hartson for bringing their complementary viewpoints to a project that spanned two different areas of computer science. I need to thank Dr. Robert S. Schulman for his help with the statistics and presentation. Lastly, I would like to express my appreciation to Dr. J. A. N. Lee for his continued support and encouragement throughout the research process.

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 Background 1

1.2 Problem Statement 6

1.3 Goal 13

1.4 Approach 13

1.4.1 Develop The Usability Problem Taxonomy 13

1.4.2 Reliability Of UPT Problem Classification 15

1.4.3 Association Of Categories With Development Context Factors 15

1.5 Summary 16

2 RELATED WORK 17

2.1 Data Gathering 18

2.1.1 Quantitative Data 18

2.1.2 Qualitative Data 23

2.1.3 Data Tracking 28

2.2 Data Classification 30

2.3 Data Analysis 39

2.4 Strategies Based On Data Classification And Analysis 42

2.5 Summary 50

3 CONSTRUCTION OF THE USABILITY PROBLEM TAXONOMY 52

3.1 Initial Data Collection 52

3.2 Initial Data Analysis 56

3.2.1 Usability Heuristic Analysis 56

3.2.2 Conclusions Regarding Heuristic Analysis 64

3.3 Development of the Usability Problem Taxonomy 66

3.3.1 Revisiting Usability Problem Data Subtypes 66

3.3.2 Clusters Of Problem Subtypes: The Usability Problem Taxonomy 72

3.4 Strengths Of The Usability Problem Taxonomy 78

3.5 Summary 79

4 THE UPT AND PRODUCT IMPROVEMENT 81

4.1 The Reliability Of The UPT And Its Use In Problem Classification 81

4.1.1 The UPT Reliability Study 81

4.1.2 Results Of The UPT Reliability Study 87

4.1.3 Remarks About The UPT Reliability Study 96

4.1.4 UPT User Satisfaction Survey 101

4.2 Product Improvement And The UPT 103

4.2.1 Two Critical Incidents 104

4.2.2 Problem Identification, Reporting, Analysis, And Correction Without The UPT 107

4.2.3 Using The UPT During Problem Reporting 110

4.2.4 Using The UPT After Problem Reporting 119

4.2.4.1 Using The UPT To Classify Problems After Reporting 120

4.2.4.2 The UPT As A Foundation For Analysis 121

4.3 Summary 126

5 THE UPT AND PROCESS IMPROVEMENT 129

5.1 The UPT Association Study 130

5.2 The Association Of Roles And Skills With UPT Categories 134

5.2.1 Association Of UPT Categories With Developer Roles 134

5.2.2 Comparing Responses On Roles Form 1 And Roles Form 2 143

5.3 The Association Of Activities, Methods, And Techniques With UPT Categories 149

5.4 A Comparison Of Roles And Activities 159

5.5 Process Improvement Using The UPT 170

5.6 Summary 173

6 SUMMARY AND CONCLUSIONS 175

6.1 Step One: The Usability Problem Taxonomy (UPT) 175

6.1.1 Summary Of UPT Development 176

6.1.2 UPT Evaluation 176

6.1.3 Contributions Of The UPT 178

6.1.4 Additional Discoveries Made During UPT Development 178

6.2 Step Two: Associations Between UPT Categories And Development Context 182

6.2.1 Summary Of The Association Study 182

6.2.2 Association Study Findings 183

6.2.3 Contributions Of The Association Study 186

6.3 Final Remarks 189

7 FUTURE WORK 190

7.1 Additional Reliability Studies 190

7.2 Refinements And Extensions To The UPT 191

7.3 Information About Usability Problems 191

7.4 New Methods 192

7.5 Tools 192

7.6 Summary 193

REFERENCES 194

APPENDIX A: UPT Categories Glossary 200

APPENDIX B: Roles Forms and Glossary 204

APPENDIX C: Activities Form and Glossary 211

APPENDIX D: Means And Standard Deviations For Roles Form 1 And The Activities Form 225

D.1 Means And Standard Deviations For Roles Form 1 226

D.2 Means And Standard Deviations For The Activities Form 234

VITA 242

LIST OF FIGURES

Chapter 1

Figure 1.1. The impact of usability problems on product and process improvement. 2

Figure 1.2. Three parts of the interactive-software development cycle. 3

Figure 1.3. Information loss in the interactive-software development cycle. 7

Figure 1.4. Product improvement using the Usability Problem Taxonomy. 10

Figure 1.5. Product and process improvement using the Usability Problem Taxonomy 12

Figure 1.6. Simplified view of the Usability Problem Taxonomy (UPT). 14

Chapter 3

Figure 3.1. Hierarchical structure of the Usability Problem Taxonomy. 76

Chapter 4

Figure 4.1. Dialogue box for disk initialization. 104

Figure 4.2. Usability problem with the name field occurs during disk initialization. 105

Figure 4.3. Error message for disk initialization 105

Figure 4.4. Data entry box. 106

Chapter 5

Figure 5.1. The overlap among roles for visualness, language, and manipulation. 138

Figure 5.2. The overlap among roles for task-mapping and task-facilitation........... 139

Figure 5.3. The overlap among roles for the artifact and task components............. 140

Figure 5.4. The overlap among activities for visualness, language, and manipulation. 153

Figure 5.5. The overlap among activities for task-mapping and task-facilitation...... 155

Figure 5.6. The overlap among artifact-oriented activities and task-oriented activities. 156

LIST OF TABLES

Chapter 2

Table 2.1. Problem Severity Classification (PSC) values.......36

Table 2.2. Cost/importance table.......49

Chapter 3

Table 3.1. Participant development projects.......53

Table 3.2. Levels of expertise of the intended user groups.......53

Table 3.3. Location, evaluators, and methods used.......54

Table 3.4. Usability Heuristics.......55

Table 3.5. The reporting techniques and number of identified problems.......56

Table 3.6. Number of problems categorized according to heuristic.......59

Table 3.7. Subtypes for simple and natural dialogue.......67

Table 3.8. Subtypes for speak the users' language.......68

Table 3.9. Subtypes for minimize users' memory load.......68

Table 3.10. Subtypes for consistency.......69

Table 3.11. Subtypes for feedback.......69

Table 3.12. Subtypes for clearly marked exits.......70

Table 3.13. Subtypes for shortcuts.......70

Table 3.14. Subtypes for precise, constructive error messages.......71

Table 3.15. Subtypes for prevent errors.......71

Table 3.16. Subtypes for other.......72

Table 3.17. The five primary categories with associated problem types.......74

Chapter 4

Table 4.1. Classifier experience 82

Table 4.2. Classifier experience 83

Table 4.3. Distribution of problems over the five projects. 84

Table 4.4. Distribution of problems over the five projects. 85

Table 4.5. Distribution of previously examined problems. 86

Table 4.6. Distribution of new problems. 86

Table 4.7. Counts for the artifact component. 88

Table 4.8. Counts for the task component. 90

Table 4.9. Counts for good and complete agreement. 91

Table 4.10. Statistics for all 20 usability problems. 95

Table 4.11. Statistics for 10 usability problems examined previously (OLD). 95

Table 4.12. Statistics for 10 usability problems not examined previously (NEW) 96

Table 4.13. Mean response per question 102

Table 4.14. Mean response per classifier. 102

Table 4.14. Problem classification for Disk Name critical incident 114

Table 4.15. Problem classification for Data Entry critical incident 119

Table 4.16. Distribution of problems across UPT categories. 122

Table 4.17. Distribution of problems across language, other wording categories 124

Chapter 5

Table 5.1. Classifier experience 130

Table 5.2. Classifier experience 131

Table 5.3. Interpretation of roles and skills results for Roles Form 1. 137

Table 5.4. A comparison of roles for the artifact component, task component, and overall UPT. 142

Table 5.5. Paired t test values for each participant. 145

Table 5.6. Mean responses for roles in Roles Form 1 also present in Roles Form 2 (5 participants). 146

Table 5.7. Mean responses for Roles Form 2 (5 participants). 147

Table 5.8. Types of activities on the Activities Form. 150

Table 5.9. Interpretation of results from the Activities Form. 152

Table 5.10. A comparison of activities for the artifact component, task component, and overall UPT. 158

Table 5.11. A comparison of roles and activities for the visualness category. 160

Table 5.12. A comparison of roles and activities for the language category. 162

Table 5.13. A comparison of roles and activities for the manipulation category. 163

Table 5.14. A comparison of roles and activities for the task-mapping category. 164

Table 5.15. A comparison of roles and activities for the task-facilitation category 165

Table 5.16. A comparison of roles and activities for the artifact component. 166

Table 5.17. A comparison of roles and activities for the task component.............. 167

Table 5.18. A comparison of roles and activities for the entire UPT structure. 168

Table 5.19. Number of each type of activity associated with UPT categories.......... 169

Table 5.20. Roles and activities that do not address UPT categories. 171

Chapter 6

Table 6.1. Summary of roles and activities that address UPT categories............. 184

Appendix D

Table D.1. Means and standard deviations for each role for visualness................ 226

Table D.2. Means and standard deviations for each role for language. 227

Table D.3. Means and standard deviations for each role for manipulation............. 228

Table D.4. Means and standard deviations for each role for task-mapping 229

Table D.5. Means and standard deviations for each role for task-facilitation 230

Table D.6. Means and means of standard deviations for each role for the artifact component.. 231

Table D.7. Means and means of standard deviations for each role for the task component.. 232

Table D.8. Means and means of standard deviations for each role for the UPT. 233

Table D.9. Means and standard deviations for each activity for visualness............ 234

Table D.10. Means and standard deviations for each activity for language. 235

Table D.11. Means and standard deviations for each activity for manipulation......... 236

Table D.12. Means and standard deviations for each activity for task-mapping........ 237

Table D.13. Means and standard deviations for each activity for task-facilitation...... 238

Table D.14. Means and standard deviations for each activity for the artifact component.. 239

Table D.15. Means and standard deviations for each activity for the task component.. 240

Table D.16. Means and standard deviations for each activity for the overall UPT.. 241

1 INTRODUCTION

1.1 Background

Although research and practice have shown that the success of a usability engineering program depends on the identification and correction of usability problems, these problems remain an underutilized source of information. Current methodology focuses on encouraging developers to incorporate *any* user interface development activity into the software process to raise the level of usability in a given product. Since this approach has been reasonably successful in many development environments, little effort has been placed on usability problem description, classification, and analysis, or on the connections between usability problems and product and process improvement. This lack of understanding not only constrains product improvement, but hampers efforts to improve the user interface development process.

Utilizing information about usability problems can lead to both product and process improvement efforts as illustrated in Figure 1.1. Usability problem classification and analysis helps developers generate both local solutions that address one problem and global solutions that address multiple problems. It also guides selection of team members, activities, methods, and techniques appropriate for a given project.

Based on private communications of the author with many interactive-software developers at several development organizations, usability problems and their relationship to product and process improvement are not well understood. As a result, the contributions illustrated in Figure 1.1 have not been realized. To investigate these possible contributions, a deeper understanding of usability problems is required.

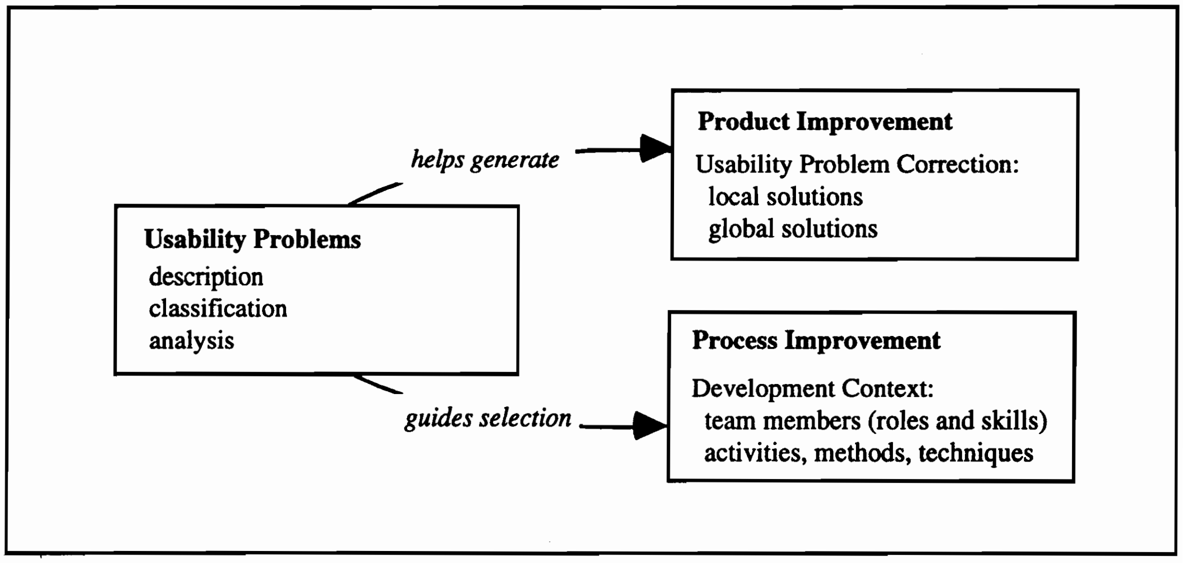


Figure 1.1. The impact of usability problems on product and process improvement.

To better understand the relationship of usability problems to product and process improvement, the relevant phases of the interactive software development cycle are examined. One way to view the interactive software development cycle is illustrated in Figure 1.2. This perspective divides the cycle into three parts. Part 1 occurs prior to problem reporting and is comprised of four phases: team member and activity selection, system design, evaluation, and problem identification. Part 2 focuses on problem reporting (the written description of detected usability problems). Part 3 occurs after problems have been reported and consists of two phases: product-oriented analysis and problem correction. While other important development phases exist in the life cycle of a software product (e.g., requirements, specification, software testing), only those phases directly related to this research are included in Figure 1.2.

The cycle begins when team members and specific development activities, methods and techniques are selected for a given project. As development proceeds, the system and user interface are designed and evaluated. During evaluation, usability problems are identified.

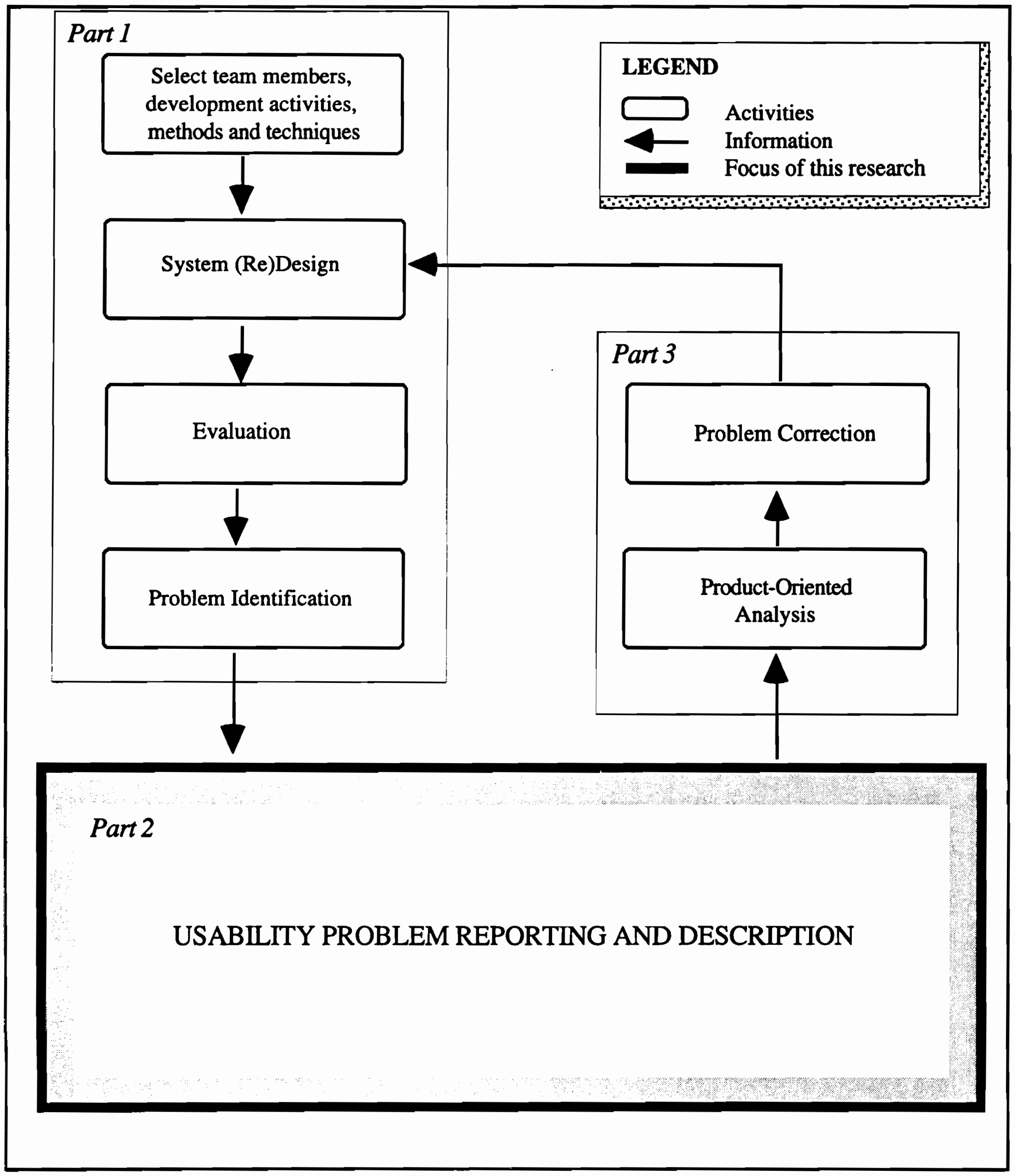

Part 1
Select team members, development activities, methods and techniques
System (Re)Design
Evaluation
Problem Identification
LEGEND
Activities
Information
Focus of this research
Part 3
Problem Correction
Product-Oriented Analysis
Part 2
USABILITY PROBLEM REPORTING AND DESCRIPTION

Figure 1.2. Three parts of the interactive software development cycle.

Part 2 of the cycle focuses on usability problem reporting. Problem reports can be generated by employees (user interface evaluators, software developers, customer service representatives) and by end users. The formality with which problems are reported varies among organizations. Some developers record usability problem descriptions in a database; others make less formal notations using pencil and paper.

The information generated by usability problem reporting impacts the success of Part 3 of the development cycle. During the product-oriented analysis phase, developers examine problem descriptions to:

- identify possible approaches to problem solutions,
- assess the cost to fix each problem,
- prioritize the problems in order of importance, and
- decide which problems will be corrected.

During the problem correction phase, product-oriented analysis is used in conjunction with the problem descriptions to select a solution for each usability problem selected for correction. The solutions are then channeled into the system design phase as the user interface is re-designed.

Although the cycle in Figure 1.2 imposes a partial ordering on the phases (e.g., problem identification must occur prior to problem correction), the elapsed time between phases may vary. For example, consider the timing of phases in the following scenarios.

- Developers practice participatory design and evaluate each user interface component as it is designed. The design and evaluation phases occur simultaneously.
- Developers use a formative evaluation technique to evaluate the user interface. The evaluation phase occurs immediately after the design phase.
- Developers do not perform user interface evaluation. Usability problems are phoned in from the field after the product has shipped. The problem identification phase occurs during maintenance after the user interface has been completed.

Regardless of the timing of individual user interface development activities, organizations allocate significant resources prior to, and during, evaluation. These expenditures include monies spent on usability laboratories, equipment, and usability-related training for developers, as well as the time spent on user interface evaluation (e.g., inspection methods, user testing). Additional resources are expended when usability problems are corrected and the user interface is redesigned. Since these resources are all invested in problem identification and correction, the problem reporting phase becomes a critical link in the development cycle.

Resources invested in user interface development activities do raise the level of usability achieved in a software system [Hix 93] [Nielsen 93b]; however, the success of a usability engineering program is limited by the quality of the problem descriptions. Since the problem reporting phase has received little attention in current research and practice, problem descriptions are often poorly written. Low quality descriptions do not translate consistently into a successful redesign effort, and as a result, limit the return development organizations realize on their investment.

1.2 Problem Statement

Despite the importance of usability problem descriptions to the success of a usability engineering program [Jeffries 94], much information is lost during the problem reporting and description phase. As illustrated in Figure 1.3, information is lost as problems are reported, and then again as problems are reconstructed from the descriptions and interpreted prior to correction.

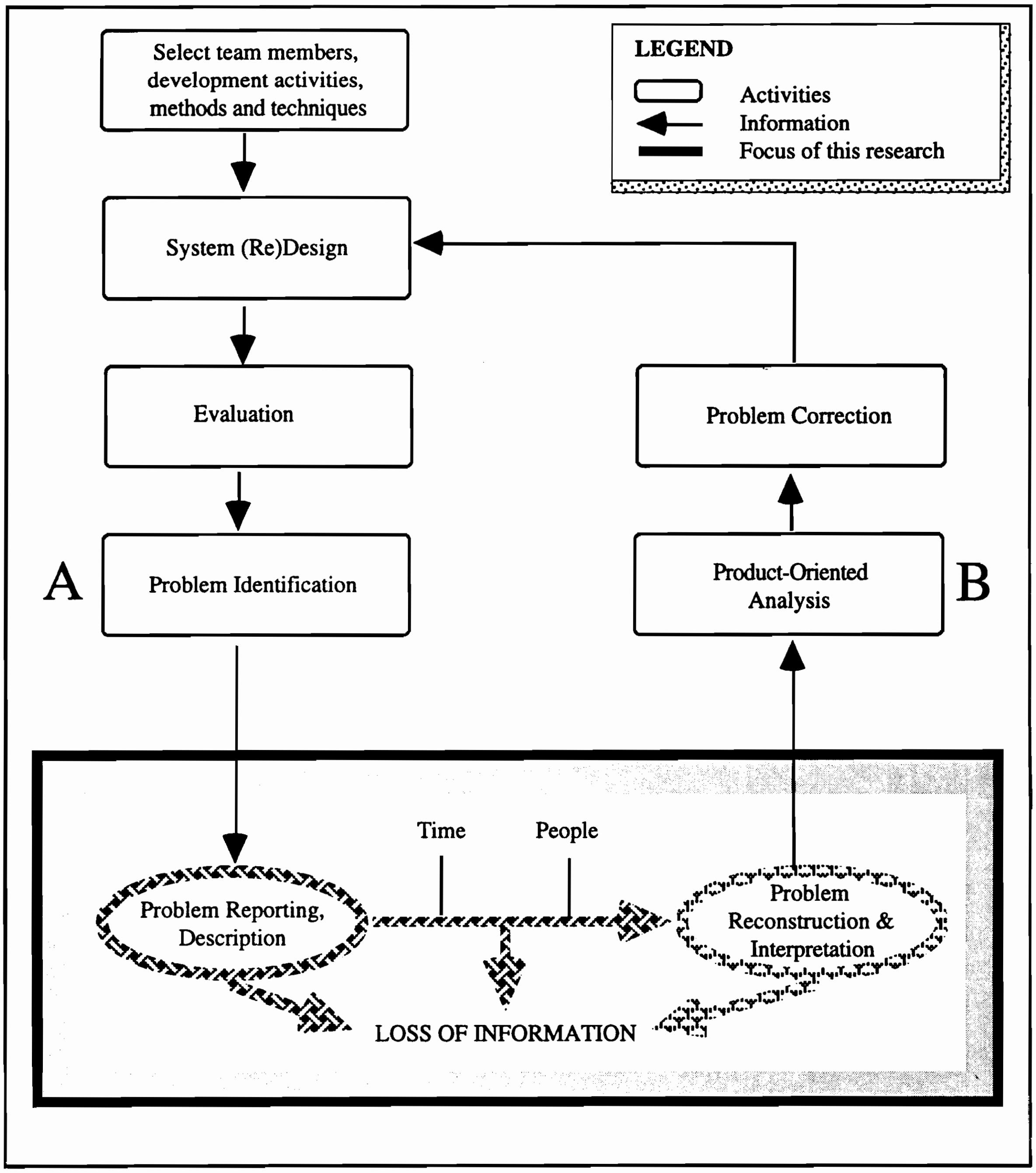
This loss of information has two causes. First, minimal guidance exists regarding the capture of usability-problem data. Second, a framework has not been developed for understanding, comparing, categorizing, and analyzing usability problems and their relationship to development context.

Writing high quality problem reports is a difficult task. Evaluators, absorbed in observing a user testing session, focus on critical incidents and, frequently, do not capture all relevant information. The large amount of contextual information, user task complexity, and time constraints complicate the reporting process. Limited evaluator expertise and experience in both usability engineering and in problem description further compound this problem.

As a result, problem reports are often vague, imprecise, and incomplete. An examination of data collected by the author on five real-world development projects indicates that many developers do not include the following information in the descriptions:

- contextual information (e.g., user task),

- what happened, and
- where it happened (location in the user interface).



Select team members, development activities, methods and techniques
System (Re)Design
Evaluation
A
Problem Identification
Problem Correction
Product-Oriented Analysis
B
LEGEND
Activities
Information
Focus of this research
Time
People
Problem Reporting, Description
Problem Reconstruction & Interpretation
LOSS OF INFORMATION

Figure 1.3. Information loss in the interactive software development cycle.

In addition, descriptions often focus on user reactions rather than what occurred or on solutions rather than on the problem. Others do not distinguish between multiple problems arising from one critical incident.

Inadequate problem reports contribute to the difficulty encountered by developers as problems are interpreted and reconstructed prior to analysis. Information loss at this point in the development cycle is exacerbated when problem interpretation and reconstruction are attempted in a later development phase, a subsequent release, by team members not present for the evaluation, or by developers at different physical locations.

The success of product-oriented analysis activities (problem prioritization, problem selection for correction) depends on the quality of the information available about each usability problem. During analysis, problem descriptions can be analyzed to determine the:

- frequency,
- severity,
- similarity, and
- clustering or grouping of the usability problems.

Let the amount and quality of information available during the evaluation and problem identification phases be denoted by A (see Figure 1.3). Similarly, let the amount and quality of information available during product-oriented analysis be denoted by B. Then A represents all observational data available. B represents the information present in the problem descriptions as well as information noted by developers as descriptions are interpreted, reconstructed, and analyzed.

When the information available during product-oriented analysis is less than that available during the evaluation and problem identification phases ($B < A$), the success of analysis, problem correction, and re-design is jeopardized. Ideally, there should be more information available during analysis than is available during evaluation ($B > A$). However, an examination of real-world data indicates that, in practice, the information loss during the problem description and reporting often results in $B < A$. At best, the surveyed organizations were only able to achieve $B = A$ for a subset of the problems identified on a given software project.

To preserve the amount and quality of information available during evaluation, a taxonomic model of usability problems is needed to help developers write high quality problem descriptions (see Figure 1.4). To enable developers to capture critical information about each observed usability problem, taxonomic categories that help developers think about each problem must be identified. Categories currently used by researchers and practitioners to classify usability problems are inadequate for this purpose.

While good problem descriptions are essential to analysis and correction, it is equally important for developers to glean as much information as possible from the entire set of usability problems detected on a given project. This additional data results from problem comparison and analysis. The absence of a taxonomic model increases the difficulty encountered by developers as (even well-written) problem descriptions are compared and analyzed.

By comparing and analyzing sets of usability problems, developers can find global solutions that may cost-effectively address multiple problems. Interviews with developers at several organizations indicated that developers tend to focus on local solutions that address individual problems rather than on global solutions. A taxonomic model that contains problem categories that facilitate description and analysis can provide the necessary structure for problem, and solution, comparison.

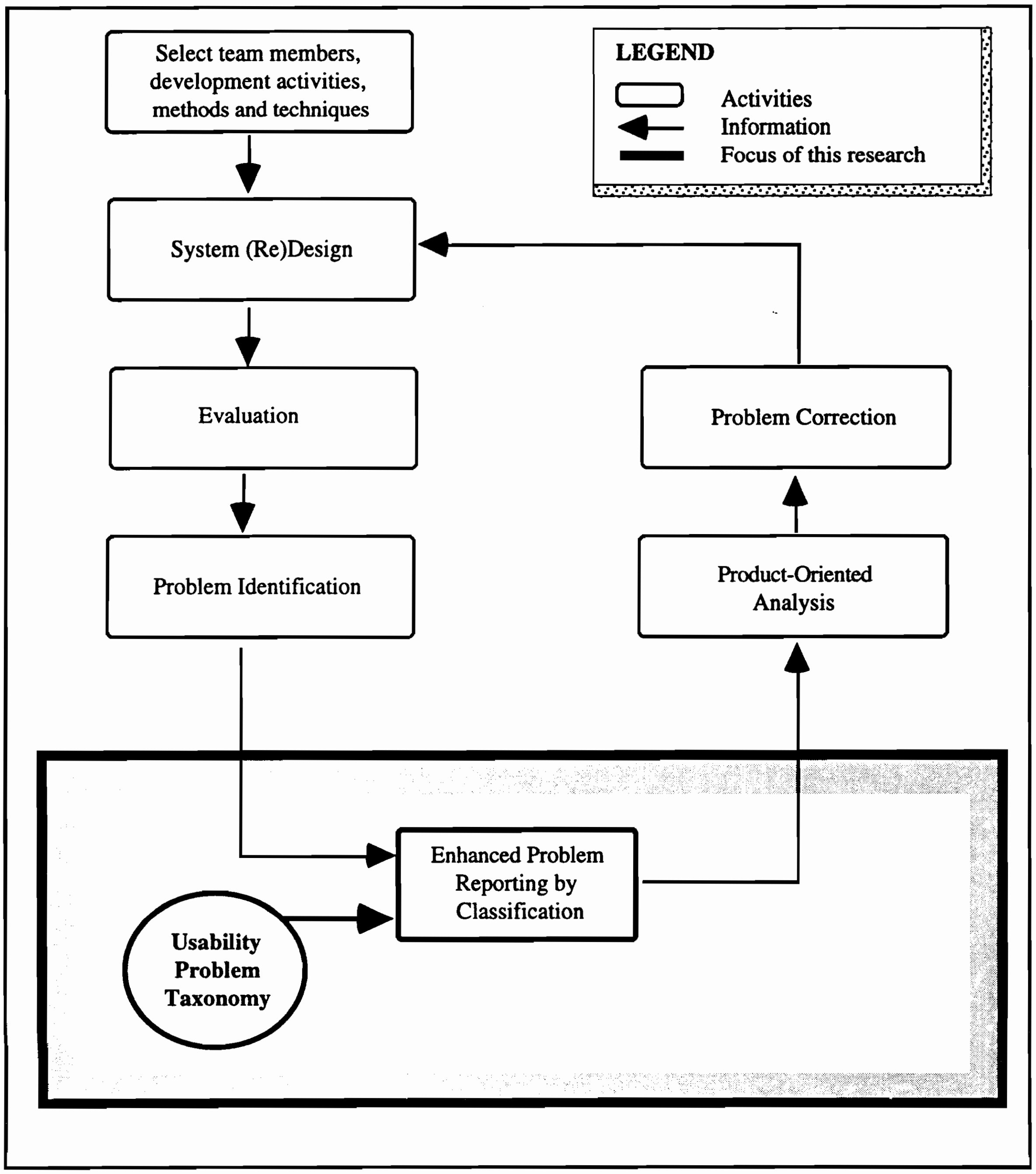
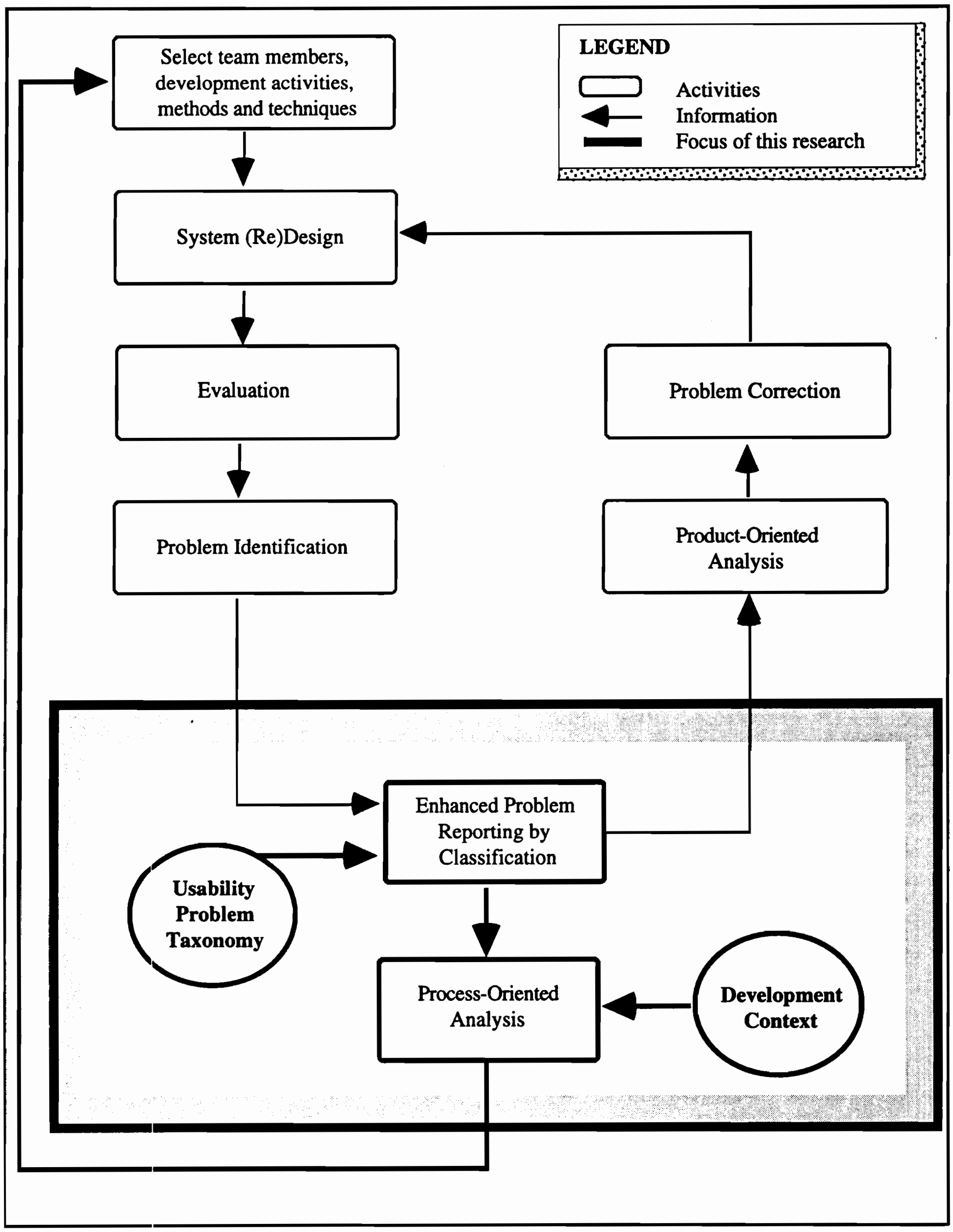

Select team members, development activities, methods and techniques
LEGEND
Activities
Information
Focus of this research
System (Re)Design
Evaluation
Problem Correction
Problem Identification
Product-Oriented Analysis
Enhanced Problem Reporting by Classification
Usability Problem Taxonomy

Figure 1.4. Product improvement using the Usability Problem Taxonomy.

Although Figure 1.4 illustrates the contributions to be made by a taxonomic model to a plan for product improvement, no clear associations exist between usability problems and development context. This reflects current software practice, i.e., no method for process decisions is based on collected usability data. Specifically, no recommendations are provided that guide managers in the selection of team members based on their skills and expertise or development activities appropriate for a given project.

A comprehensive framework for usability-related product and process improvement, shown in Figure 1.5, illustrates the dual function of a taxonomic model for classifying usability problems. The taxonomic model can be used to improve the quality of problem descriptions and guide analysis of usability-problem data. In turn, the analysis can be used to assess the relative benefits of both global and local solutions. The analysis can also be used in conjunction with development context to develop strategies for process improvement. Since the success of many user interface development methods depends on the developer's ability to report problems clearly and precisely, and the manager's ability to formulate process-improvement strategies from project-to-project, research that advances the level of understanding in this area is critical.



LEGEND
Activities
Information
Focus of this research
Select team members, development activities, methods and techniques
System (Re)Design
Evaluation
Problem Identification
Problem Correction
Product-Oriented Analysis
Enhanced Problem Reporting by Classification
Usability Problem Taxonomy
Process-Oriented Analysis
Development Context

Figure 1.5. Product and process improvement using the Usability Problem Taxonomy.

1.3 Goal

The goal of this research is to develop a framework for understanding, describing, comparing, and analyzing usability problems and their relationship to development context. The following two subgoals arise.

1. Develop a reliable, empirically-derived taxonomic model that categorizes (classifies) usability problems.
2. Establish associations between usability-problem categories and two aspects of development context, specifically development activities, and team roles and skills.

1.4 Approach

This section gives a brief overview of the three steps in this research project. First, the Usability Problem Taxonomy was developed. Second, problem classification using the Usability Problem Taxonomy (UPT) was shown to be reliable. Third, associations between the Usability Problem Taxonomy and two development context factors (user interface development activities, developer roles and skills) were identified. A detailed description of these steps can be found in Chapters 3, 4, and 5, respectively.

1.4.1 Develop The Usability Problem Taxonomy

An empirical study was undertaken to guide development of a taxonomic model of usability problems. Five projects were surveyed at four real-world, interactive software development organizations (see section 3.1). Six hundred and forty-five usability problems detected on those projects were collected by the author and used in this research. Initially, 406 of the 645 problems were examined and used to build the taxonomic model. All 645 problems were used in the UPT reliability study described below.

The researcher began by categorizing each of the 406 usability problem descriptions according to the usability heuristic(s) violated [Nielsen 90]. Four weaknesses of heuristic analysis were identified (see section 3.2.1). Although it was concluded that the heuristics were an inadequate classification tool, the analysis did provide an initial direction for the development of the UPT. Specifically, usability problems classified within each heuristic were re-examined and grouped according to commonalities among problems.

The Usability Problem Taxonomy, illustrated in Figure 1.6, was developed using an iterative process that focused on identifying commonalities among the detected usability problems. Five primary categories were identified: visualness, language, manipulation, task-mapping, and task-facilitation. These five categories were grouped into two components: the artifact component and the task component. The two components, illustrated in Figure 1.6 correspond to the two dimensions of a usability problem. When classifying usability problems using the UPT, each problem receives two classifications: one in the artifact component and one in the task component.

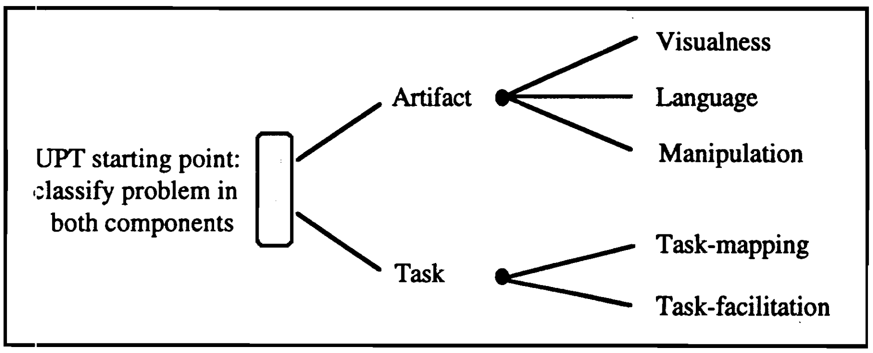


Figure 1.6. Simplified view of the Usability Problem Taxonomy (UPT).

1.4.2 Reliability Of UPT Problem Classification

A study was undertaken to show that, using the UPT, usability problems can be classified reliably, i.e., a single problem would be classified the same way by different classifiers. Seven classifiers were chosen from industry and academic development environments. Each classified 20 randomly selected usability problems. A set of 10 usability problems was selected from the group of problems examined previously. The remaining 10 were selected from the group of problems that had not been examined.

The seven classifiers categorized each usability problem using UPT forms that are on the World Wide Web (contact the author for URL). Responses were analyzed using the kappa statistic to assess the level of agreement among classifiers [Cohen 60] [Fleiss 71].

1.4.3 Association Of Categories With Development Context Factors

A second study was undertaken to identify associations between the five primary UPT categories and two development context factors: development methods, techniques, and activities (referred to as "activities"), and developer role and skills. Six experts from industry and academic environments were selected to participate. Each was asked to assess the degree to which various developer roles and skills, and individual development activities could be used to address usability problems in each of the five primary categories (for the purposes of this research, "address" means prevent, detect, think about, and/or correct). Developer roles and skills were based on developer background, training, and expertise critical to both user interface and software engineering. Development activities included those that focus on usability (user interface engineering) as well as those that focus on the software (software engineering).

The mean and standard deviation of responses were used to assess the level of agreement between experts. Roles and activities that could be used to address problems in each UPT category were identified. Roles and activities that would definitely not address individual categories were also identified.

1.5 Summary

This chapter discussed the impact that usability problems could have on interactive software development. Although they currently contribute to raising the level of usability achieved in the user interface, they remain an underutilized source of information. The importance of a taxonomic model in both product and process improvement was outlined. Contributions to product improvement included improved problem description, classification, and analysis. Contributions to process improvement included enabling developers to use the taxonomy as a diagnostic tool to identify developers roles and skills and development activities, methods, and techniques appropriate for a given project. The approach to this research consisted of three steps: development of the UPT, a study that showed that the UPT can be used to classify usability problems reliably, and a study in which associations of roles and skills, and activities, with UPT categories were identified.

2 RELATED WORK

This research concentrates on usability-related product and process improvement *within* the context of software development. Since plans for usability engineering are often implemented by usability specialists, software engineers, and managers, a usability engineering program must account for software engineering concerns, language, and methodology. To formulate product and process improvement strategies that can be understood and implemented by development team members with varying backgrounds and expertise, techniques employed by researchers and practitioners in both communities must be considered.

This chapter examines the similarities, commonalities, differences, and gaps among the approaches used by software engineers and those used by usability specialists. Four topics relevant to this research are explored:

- data gathering,
- problem classification,
- data analysis, and
- product and process improvement strategies.

Each of these areas is examined first from a software engineering viewpoint, and second, from a usability engineering viewpoint. The two perspectives are then compared.

The chapter is organized in the following way. Data gathering is discussed in section 2.1. Classification techniques used to categorize collected data are presented in section 2.2. Data analysis is outlined in section 2.3. Strategies for interactive software process and product improvement are covered in section 2.4. A brief summary is given in section 2.5.

2.1 Data Gathering

This section examines three topics: quantitative data, qualitative data, and data tracking techniques. Each topic is presented from a software engineering perspective as well as a usability engineering perspective. Each discussion concludes with a comparison of the two perspectives.

2.1.1 Quantitative Data

Software Engineering

While software engineers collect a variety of data, the following measures are most commonly used in product and process improvement efforts [Grady 94] [Lamb 88] [Pfleeger 91] [Pressman 87] [Sommerville 89]:

- design complexity (number of design modules suggested),
- code complexity (number of code modules),
- requirements complexity (number of object interfaces),
- discovered defects,
- discovered failures,
- cost-to-fix,
- time-to-fix, and

- number of modules affected by a proposed change.

The data are gathered by various team members at different points during the development process. Design, code, and requirements complexity are assessed by software designers and implementors. Defects and failures are discovered during testing before and after the product has shipped. Defects are also detected by marketers when complaints are phoned in from the field. Cost-to-fix and time-to-fix as well as the number of modules affected by a proposed change are determined by software developers.

Usability Engineering

Quantitative usability metrics are used by usability experts to assess the level of usability achieved in the product. Data are collected on many components of usability, including: efficiency, learnability, memorability, ease of use, and user satisfaction [Nielsen 93b]. Types of usability measurements include:

- discovered defects (usability problems),
- number of times the user failed to complete a task [Nielsen 9b],
- time to task completion,
- number of subtasks that can be completed within a given time limit,
- number of user errors,
- error rate, and
- time for error recovery.

The data listed above are collected by various development team members. Usability problems and task completion failure are observed by usability specialists, marketing personnel, and software engineers. The time measures and error rates are generally noted by usability specialists.

Although most of the data listed above are collected during user testing, some can be collected using other methods. For example, lists of usability problems can be obtained as user complaints are phoned in from the field after the product has shipped. Estimates for the number of user errors and number of times the user failed to complete a task can likewise be obtained over the phone as complaints are logged.

Quantitative data pertaining to the components of usability are also collected from users through questionnaires. The questionnaires ask users to rate their reactions to various system features and overall usability using a small scale (e.g., 1 to 9 where 1 represents an unfavorable response and 9 a favorable response, respectively [Hix 93]).

Although the quantitative measures used by usability engineers contribute significantly to improving product usability, the number of available measures (and techniques for obtaining them) is still modest. At this time, the measures focus primarily on the user, task completion, and product usability. More research is needed to develop additional measures that are equally useful at different stages during the product life cycle, as well as measures that can be used to improve the user interface development process.

A Comparison of Perspectives

Although the first three software engineering measures (design, code, and requirements complexity) may be applicable to user interface software and requirements, at this time no evidence supports this hypothesis. In addition, the relationship of each metric to usability and the user interface has not been demonstrated. Design complexity, based on the number of calls from a given module (fanout), has been related to the probability of defects in that module (i.e., fanout ** 2). This result has also not been investigated to determine its applicability to usability problems and user interface software code modules.

In addition, no research has considered the three metrics in terms of the complexity of the user interface. Some recent work in human-computer interaction has focused on the development of a technique to determine the complexity of the user interface [Miller 94]. While this technique shows promise, little progress has been made that links user interface complexity with either the complexity of the usability requirements or the level of usability achieved in the final product.

Listing discovered defects as well as system failures is a useful technique in a user interface context as well as in a software-engineering context. Current recommendations in human-computer interaction focus on reporting usability problems separately from software defects. However, interviews with software developers indicate that software defects and usability problems are often not only combined in the same list, but contained in the same problem report. While this approach indicates that, to many software developers, usability and functionality are intertwined, this method of reporting problems has disadvantages. These drawbacks affect the way usability is viewed within a development organization, influence how usability problems are selected for correction, impact the design of a potential solution, and make it difficult to use the usability problem data in process improvement efforts.

Cost-to-fix and time-to-fix have been used successfully by both researchers and practitioners in both communities. Especially in the context of the user interface these measures can be difficult to assess. When a usability problem is detected prior to implementation, it is virtually impossible to assign a cost-to-fix. In cases where the cost-to-fix cannot be estimated, usability specialists consider the cost associated with *not* correcting the problem. These metrics are valuable components of cost/benefit analysis, and help developers distinguish between usability problems that must be corrected immediately, and those which can be corrected in a later development phase [Meads 93] [Nielsen 93a]. More research is needed to determine the best technique to derive these metrics.

In a software engineering context, the number of modules affected by a proposed change is a useful measure of the effort required to correct a problem. While it is useful in procedural programs, it is especially useful in an object-oriented development project. In this context, identifying the modules that received the most activity during design, implementation, and testing can be used in redesign efforts as well as team structure [Henry 94]. While this metric may prove to be equally useful in a user interface context, no research has investigated this area.

Although software engineers and usability engineers both recognize the need to collect and analyze data to realize both product and process improvement, the approaches used by the two communities are very different. Much research in software engineering has been devoted to development improvement strategies based on software-product metrics (McCall's software quality factors, Halstead's software science, and McCabe's complexity measures [Pressman 87]), and more recently, software-process metrics [Henry 91] [Henry 93] [Humphrey 90]. Research in usability engineering has focused on usability metrics that assess the level of usability achieved in the user interface of a given software system [Hix 93].

While some software measures do include references to usability (McCall's software quality factors, the SEI's Capability Maturity Model [CMM 93a] [CMM 93b], usability is not the primary focus. Complexity measures could be applied to user interface software, but their relationship to usability has not been investigated. Process measures based on error rates, change data, and project management data are promising, but have not yet been extended to include the user interface development process.

2.1.2 Qualitative Data

Software Engineering

Some qualitative data are gathered by software engineers. These data include customer responses gathered during needs analysis interviews [Pfleeger 91], and prose descriptions of bugs and suggested solutions. Although the qualitative data gathered during needs analysis contributes to system requirements, they are not directly related to this research effort. An examination of the literature revealed that recommendations do exist for writing bug reports and suggested solutions [Lamb 88] .

The recommendations regarding the capture of qualitative, software engineering data focus on classifying the report in one of three categories:

- no action necessary (the bug is either already being fixed or was not a problem to begin with),
- a software change needed (the system does not meet the specified requirements), and

- a change request (system met the specified requirements, however, the requirements need to be changed).

The recommendations also address the need to estimate the resources needed for the requested change. Some developers list the modules affected by the proposed change in the reports [Henry 94]. Developers collect the data and write the reports as defects are noted during the testing and maintenance phases.

Usability Engineering

Several types of qualitative data are used by usability engineers. Interviews and focus groups conducted with end users provide useful qualitative information (i.e., with respect to requirements gathering, user studies, user satisfaction, product assessment, and verification of final acceptance) [Hix 93] [Schneiderman 87]. Although important in the life cycle of a interactive software product, these data are not a related to this research.

Usability problem descriptions, however, are a critical component of this research effort. Usability problems can be detected in many phases of the product life cycle including design, evaluation, and maintenance. Research has focused on the development of activities and techniques that can be used to detect usability problems. These activities include participatory design, verbal protocol taking, co-discovery learning, cooperative evaluation, inspection methods, and user testing. Although usability problems detected using these methods are identified by user interface evaluators, marketers, software developers, and end users, they are generally reported by evaluators, software developers, and marketers. A new technique, the use of incident diaries, relies solely on the user to describe problems that occur as the system is used and provide the necessary contextual information (e.g., what time did problem occur, what started the problem, which help facilities were investigated, how useful were the help facilities, the length of time spent resolving the problem) [Macaulay 95].

Although the advances in user interface evaluation activities have been substantial, little research has addressed problem reporting techniques. Few recommendations exist that address how usability problems should be reported. Since generating concise, informative, prose descriptions of usability problems and suggested solutions is critical in the context of usability engineering, more emphasis must be placed on research in this area.

The lack of guidance regarding problem reporting contributes to the loss of information between the problem identification and reporting phase and the problem correction phase (see sections 1.1 and 1.2). To improve communication between user interface evaluators/developers who identify the usability problems and developers who correct those problems, it has been suggested that the following additional information be provided [Jeffries 94]:

- various levels of abstraction of the problem description,
- proposed solution, separate from the problem description,
- justifications for each problem and each solution using usability principles, and
- trade-offs associated with multiple proposed solutions.

Achieving the appropriate level of abstraction in a problem description is difficult for most developers. However, current research has shown that by incorporating the appropriate level of abstraction in the problem description, the actual problem can be differentiated from the observed symptom(s) and result in better solutions [Jeffries 94].

For example, a problem report that focuses on the user's objection to the placement of an individual menu item within a menu [Jeffries 94] could result in a solution in which the recommendation is made to move the specific menu item closer to the top of the menu. If, in fact, the actual problem is more pervasive and requires a complete reorganization of all menus and items within those menus, this description obscures identification of the actual usability problem. This occurs because the level of abstraction in the problem description is too fine and focuses on a very specific detail of the user interface.

Three primary advantages result from the separation of the problem and solution descriptions. First, the problem description remains focused on the user and the task in which the problem occurred. Second, false alarms (something reported as a problem, but is not an actual problem) can be more easily detected. Third, developers can devise multiple solutions to individual problems. Some solutions may be local and address one problem. Others may be global and address multiple problems. Global solutions may be comprehensive or system-wide.

While prose problem reports take more time to write and read, the additional information they provide can be extremely useful. Including justifications for each problem and each solution, as well as the trade-offs associated with each solution, can help evaluators communicate more effectively with developers who have little background or training in usability.

A Comparison of Perspectives

While both software engineers and usability engineers report problems, the type of problems that are detected differ significantly (coding errors versus usability problems). These differences impact the type of information that has been collected. Qualitative data collected by software engineers are used to effect corrections in the software, and specific aspects of bug reports (such as the modules affected by a change) are a significant component of change management methodology. Since a focus on usability implies a focus on the user, usability problem reports focus on user errors, user reactions, and potential solutions. Research is needed to determine if the type of information collected about bugs can be applied to usability problems.

Few techniques focus on effective usability problem reporting. The recommendations discussed above are not widely used in practice as evidenced during interviews with many interactive software developers. These recommendations suggest that by using an appropriate level of abstraction, and separating the problem from the solution, the actual problems will be revealed. While this guidance is promising, identifying the appropriate level of abstraction is difficult for novice reporters [Jeffries 94]. No recommendations focus on problem reporters with varying levels of expertise. In addition, usability problems cannot be characterized fully by an appropriate level of abstraction. More research is needed to investigate other aspects of usability problems and to determine if usability problem characteristics can guide problem reporting.

2.1.3 Data Tracking

Software Engineering

Software engineers track both product and process data throughout development and maintenance [Henry 93] [Humphrey 90] [CMM 93a] [CMM 93b]. Two recommendations that guide process data tracking [Humphrey 90] are also applicable to product data. The first recommendation is to track items that are a natural result of the development process and, as a result, do not impose a significant additional burden on software developers. The second recommendation is to keep the tracking activity distinct from the reviewing activity (tracking data requires evidence that an item is complete, e.g., that a defect has been completely corrected).

Software engineers track specific types of quantitative data that are associated with product improvement. Product data include code size (lines of code), planned resources, number of modules completed, tests successfully completed, and number of defects found.

Process measurements are important regardless of what kind of system is being developed. These measures encompass project management data such as cost estimating, staffing, scheduling, efficiency, and progress [Henry 93]. Other metrics, based on the SEI's Capability Maturity Model, focus on measurements collected at the conclusion of each activity and/or development phase [CMM 93a] [CMM 93b]. Additional process data are based on rate charting, e.g., defect removal rate.

Usability Engineering

Although some usability engineers do track data throughout the product life cycle, usability data tracking is still quite rare. The data are primarily related to product improvement as little data related to the user interface development process have been identified. Data that can be tracked include usability goals, usability specifications, results of benchmark tasks, and usability problems.

One case study reported by Hewlett-Packard indicates that by using defect tracking, life cycles, and usability goals, efficiency of the development process was improved and product quality (including usability) increased [Rideout 91]. While this success story is encouraging, more work is needed to identify which data are important to track in the context of the user interface.

A Comparison of Perspectives

Research is needed to determine if the type of product data that are tracked by software engineers can be applied successfully to the user interface and user interface software. More work is needed to investigate how tracking usability problems can be used in both product and process improvement. In addition, research is needed to determine if important relationships exist between product data that pertain to user interface software and product data that pertain to system usability.

In principle, process measures can and should be applied to user interface development. Usability engineers are beginning to search for relationships similar to those identified by software engineers; however, too few results exist. Not only is the impact of user interface development activities difficult to measure, but no foundation exists on which to assess relationships between process and product data relevant to the user interface.

2.2 Data Classification

To facilitate analysis, data are often categorized according to various classification schemes. Taxonomies have proven to be important tools in empirical research, because a close relationship often exists between the way errors are classified and the way their occurrence is explained. In addition, the choice of classification scheme can impact how easily the frequency or the consequence of an error can be reduced or minimized [Senders 91]. Each type of taxonomy suggests particular kinds of explanations that are based on the focus the taxonomic categories. The explanations are often associated with strategies for error reduction as well as those that minimize the consequences of errors.

This section first describes general classification techniques used by cognitive psychologists to classify human error. Second, specific taxonomies used by software engineers to classify software data are presented. Third, methods for error classification used by usability engineers to classify usability problems are outlined. A comparison of the two perspectives is then presented.

Cognitive Psychology

Cognitive psychologists have offered the following types of taxonomies in which to classify human errors. These classification schemes are based on the "nature and source of human error" [Senders 91].

- A phenomenological taxonomy classifies at a superficial level. This type of taxonomy is based on events as they were observed. These taxonomies include categories such as omissions, substitutions, or unnecessary repetitions. In terms of a software system, applicable classes of events are recoverability, human error, or machine error.

- A cognitive taxonomy classifies human errors in terms of perception, memory, and attention. This kind of taxonomy is often viewed as a taxonomy of behavior and is strongly supported in the literature on experimental psychology.

- Another level of taxonomy approaches error classification through classes based on biases or deep-rooted tendencies of the subjects.

Binary taxonomies are also been used [Senders 91]. Two examples are given below.

- Endogenous versus exogenous taxonomies classify events according to whether the error was caused by something within the individual or by something in the environment.

- Some phenomenological taxonomies classify errors according to slips (a good plan but poor execution) versus mistakes (an incorrect intention) [Norman 88].

Taxonomies are often partitioned into various levels of analysis. These levels view each error in the following ways: what happened, how it happened, and why it happened.

Software Engineering

Various taxonomies are used by software engineers to guide product and process improvement efforts [Ostrand 84]. Software defects are classified according to:

- development phase the defect was discovered, i.e., requirements, design, coding, or testing,
- activity performed when the defect was discovered,
- module size,
- module name,
- number of modules affected,
- system function,
- origin (development phase the defect was introduced),
- cause, and
- severity, and
- cost-to-fix.

Classifying defects according to the phase in which they were ~~discovered~~ introduced allows developers to analyze the development process. Developers analyze the data to determine if additional activities should be incorporated into the current software process.

By classifying defects according to activity in which the defect was detected, the effectiveness of individual development activities can be assessed [Weiss 79]. This information can be used to identify improvements to made to the activities to maximize their effectiveness [Henry 93].

Defect classification schemes based on module size, module name, number of modules affected, or system function can impact system design or redesign. In addition, these taxonomies have been used to guide system tests by identifying where defects are most likely to be found [Myers 79].

Identifying the origin and cause of a defect helps developers track a causal chain of events. If a documented development history is not available, identifying the origin of a defect requires subjective judgments from the developer(s). Causal analysis techniques (see section 2.3) ask developers to select one of several predetermined causes, or summarize their opinions in one or two statements.

Software engineers also assess the severity, and cost-to-fix, of individual defects in order to prioritize the defects prior to correction. Severity is captured by whether or not the problem is superficial, produces incorrect output, or is considered a system failure. The cost-to-fix is dependent on the life cycle phase in which the problem is detected, and is often amplified in later life cycle phases [Pressman 87].

Usability Engineering

Some research in human-computer interaction has focused on classifying usability problems to draw conclusions about the relative effectiveness of individual activities methods [Brooks 94] [Desurvire 94] [Karat 94a] [Mack 94] [Nielsen 93b]. Four classification schemes are used most frequently. These taxonomies classify usability problems according to:

- whether or not the problem is a core problem or a non-core problems (a binary taxonomy),
- the usability heuristic violated,
- severity rating, and
- location in the dialogue.

Each classification scheme is described briefly below.

A usability problem is classified as a core problem if it is associated with the portion of the interface which is examined in depth during a heuristic evaluation. Non-core problems are other problems identified during a cursory examination of the remaining parts of the interface. Studies which have investigated the differences between inspection methods and user testing have compared respective sets of core problems in an attempt to identify and explain observable differences [Karat 92] [Karat 94a].

Some researchers and practitioners classify usability problems according to the usability heuristic that is violated. The 10 commonly used usability heuristics are outlined below:

- simple and natural dialogue,
- speak the users' language,
- minimize the users' memory load,
- consistency,
- feedback,
- clearly marked exits,
- shortcuts,

- precise and constructive error messages,
- prevent errors, and
- help and documentation [Nielsen 93b].

Various techniques exist for assessing the severity of a usability problem. Three different techniques are presented below.

The first technique focuses on three factors: the frequency with which the problem occurs, the impact of the problem, and the persistence of the problem [Nielsen 93b]. Using these three factors, developers/evaluators rate each problem from 0 to 4 as follows:

0 - not a usability problem,
1 - cosmetic problem, fixed only if extra time available,
2 - minor problem, attach low priority for fix,
3 - major problem, attach high priority for fix, or
4 - catastrophe, must be fixed prior to release.

This particular technique has been used in conjunction with usability inspection methods such as heuristic evaluation.

The second technique assesses severity according to a three point scale: a minor annoyance or confusion, a problem which caused an error, and a problem which caused a task failure [Desurvire 94].

The third technique assesses severity by focusing on two factors: impact (on the user) and frequency (percentage of users who experience the problem). The relationship between these two factors is referred to as the Problem Severity Classification (PSC), and is illustrated in Table 2.1 [Karat 92][1]. The entries in the table are the numeric ratings that are attached to each usability problem (1 represents the most severe, 3 represents the least severe).

Table 2.1. Problem Severity Classification (PSC) values.

Impact on Task		Frequency (% of users) High	Frequency (% of users) Moderate
	High	1	1
	Moderate	1	2
	Low	2	3

If the problem is determined to be a "no action problem," it is assigned a value of 99. Otherwise, it is assigned a value of 1, 2, 3.

Regardless of the techniques used to assign a severity rating to each usability problem, severity ratings are used to classify usability problems detected on a given software project. Developers use the ratings to identify which problems will be corrected as well as when those problems will be corrected (which development phase or product release).

Usability problems are also classified according to where their location in the user interface, i.e., the specific point in the dialogue at which the problem occurred. One such categorization distinguishes among problems found in a single location in the interface,

1 This table is taken from Table 1, page 401.

those found at two or more locations, problems with the overall structure, and those caused by omissions [Nielsen 93b]. This categorization has been used to contrast effectiveness of paper mock-ups with working prototypes and to determine if specific types of usability problems are easier to detect than others.

A Comparison of Perspectives

Some of the classification schemes used by cognitive psychologists have been applied to the user interface. Phenomenological taxonomies that are based on omissions, substitutions, or unnecessary repetitions have been modified to examine the way in which the user completes a task on a software system. Issues such as user perception, memory, and attention (in a cognitive taxonomy) are important aspects of usability and are currently the focus on much research in human-computer interaction. Error classification based on biases or deep-rooted tendencies of the subjects have not been investigated by human-computer interaction researchers and may not be as applicable to usability. Schemes which classify errors as either endogenous or exogenous can be useful as developers choose between training (endogenous errors) or system redesign (exogenous errors).

Although the levels of analysis (what happened, how it happened, and why it happened) have been used to guide problem reporting, many of the taxonomies developed by psychologists have not been incorporated into the life cycle. In general, software developers do not have the background or expertise to use these taxonomies efficiently or effectively. Some of the classification techniques, such as slips versus mistakes, have not provided developers with the information needed for problem correction [Vora 95]. In addition, the taxonomic classes discussed above have not been linked to product improvement activities or process improvement methodology. More research is needed to determine if the approaches espoused by cognitive psychologists can be modified, extended, or interpreted for use during product development.

Some of the classification techniques used by software engineers have been applied to user interface development. Usability specialists also associate cost-to-fix to with when a problem is detected during development. This type of analysis is then used to encourage the introduction of user interface development activities early in the product life cycle.

Other taxonomies used by software engineers have not been applied to the user interface. These include module size, module name, number of modules affected, origin, and cause. Interviews with software developers indicated that sometimes usability problems are classified according to system function and corrected when the associated functional enhancement is implemented. More work is needed to determine if the classification schemes used by software engineers can contribute to user interface development process.

While the taxonomies currently used by usability engineers compare individual evaluation techniques, they fail to provide sufficient information to guide both process and product improvement. Research is needed to determine if techniques for classifying software defects can be applied to usability problems, if new taxonomies for classifying data can be discovered, and which classified data are relevant to process and product improvement.

2.3 Data Analysis

While the classification of defects does provide interactive software developers with a better understanding of system problems and potential solutions, further analysis of classified product and process data can provide even greater understanding. This section outlines approaches used by software engineers, usability engineers, and concludes with a comparison of the approaches.

Software Engineering

Software engineers have developed many techniques for analyzing product and process data. Many analysis scenarios focus on the data and classification schemes outlined in sections 2.1 and 2.2. Two other important techniques are described here: causal analysis and clustering analysis.

To develop a causal theory, relationships among the following types of information need to be identified:

- explanations for its introduction,
- a description of the actual event (e.g., software defect),
- when the problem was identified,
- when it was introduced, and
- justifications for actions taken.

The four levels of inquiry used by psychologists to analyze human error can contribute significantly to a causal theory [Senders 91]. The first level of inquiry (why the error occurred) is directly linked to the explanation for its introduction. The second level (what error occurred) corresponds to a description of the actual event. The third level (which object was involved) may also be included in the description of the event. The fourth level (to whom, where, and when it occurred) corresponds, in part, to when the problem was identified and when it was introduced.

As causal relationships are identified, the results can be used to create techniques and activities designed to eliminate error (process improvement), determine error rates and identify sources of system unreliability, and provide a comprehensive context for the error if the user's environment is included in the problem description.

If no causal mechanisms can be determined, then a software developer can do little except study the statistics and examine circumstances under which error rates vary. This cursory type of analysis will reveal consistent relationships between the frequency of errors and individual development circumstances; however, it does not provide a rationale on which to base long-term decisions spanning multiple development projects.

Causal analysis is a technique for identifying the cause of a software defect. As the causal chain is tracked, a practical cause (the point(s) in the chain at which some action can be taken) surfaces. Software engineers have used defect causal analysis to determine when, and what, corrective actions are needed [Giblin 92] and have reported that defect causal analysis reduced certain types of errors over subsequent releases of a given project [Card 93].

Clustering analysis has also been used by software engineers to analyze problems by investigating groups of problems. The classification schemes discussed in the previous section provide the framework for this type of analysis. One important result discovered during analysis was that software defects are generally found in close proximity to other defects, i.e., that the probability that more errors exist in a program section is proportional to the number of defects already detected in that section [Myers 79]. This observation has enabled developers to focus testing efforts on sections of code (particular modules or functions) which appear to be more error prone. This particular result has encouraged software engineers to look for other ways in which defects might cluster, e.g., development phase discovered and development phase introduced [Giblin 92].

Usability Engineering

Some work in user interface development has focused on identifying clusters of usability problems. Two of the classification schemes discussed in section 2.2 provide the basis for this analysis: heuristic analysis [Nielsen 93b] and location in the dialogue. Although usability problems have been analyzed according to heuristic, little research addresses how the analysis is to be used in either product or process improvement. The heuristic categories have not been linked to activities, methods, or techniques. Identifying clusters of usability problems according to their location in the dialogue is similar to other research efforts that have focused on classifying user errors according to tasks and subtasks. This type of analysis has enabled developers to suggest appropriate corrections that eliminate the errors in each cluster, i.e., for each task [Cox 94].

A Comparison of Perspectives

Research is needed to apply both causal and clustering analysis to usability. To determine whether or not causal analysis will be an effective tool for user interface developers, much data needs to be collected and analyzed. In particular, data and analysis techniques are needed to determine if relationships identified by software developers hold true for the user interface. For example, no evidence suggests that the probability that usability problems exist in a given dialogue element is proportional to the number of usability problems already detected in that element or that usability problems predicted by experts cluster differently from problems observed during a user test [Mack 94].

Although usability problems have been analyzed for clustering according to heuristic and location in the dialogue, more work is need to determine if clustering or grouping can provide a more thorough understanding of usability problem characteristics [Jeffries 94]. Statistical analyses can be used to determine if relationships exist between clusters of usability defects or if relationships among clusters exist across projects and application domains.

2.4 Strategies Based On Data Classification And Analysis

This section focuses on process and product improvement strategies used by software engineers and those used by usability engineers. The section concludes with brief comparison of strategies used by each community.

Software Engineering

Error data (quantitative and/or qualitative) are crucial to the success of both process and product improvement. These data are often used to guide future development projects [Ostrand 84] [Weiss 79] and can be used to determine:

- the effectiveness of current software development techniques,
- the success of new development methodologies,
- whether or not design goals were met,
- the effectiveness of the error detection techniques,
- the major sources of error,
- error rates, and
- trends in error statistics.

Tracking software error data contributes significantly to strategies which include project estimation and progress monitoring. In addition, removing causes of major defects offers the best short-term potential for guiding product improvement.

Error prediction is also an effective product improvement strategy [Senders 91]. Predictions can be made about types of errors, point probabilities (probability that something will happen on a particular occasion), and changes in error rates.

Sound error prediction requires that both the timing of an error (when it will occur), and its form be identified. Taxonomies chosen for this type of information are critical, because a high correlation has been found between causality and taxonomic categories.

Taxonomies must be selected carefully as they influence the ease with which errors are predicted, e.g., one that is too specific can interfere with prediction.

Software engineers have used several strategies based on error prediction to reduce errors. One strategy is to predict the form an error will take so that the system design can be improved to "absorb" the errors. A second strategy assumes that certain types of errors can not be eliminated. It focuses design so that undesirable consequences of error (such as frequency or severity) are reduced or minimized. A third strategy is not correct certain classes of errors, but handle them through improved training sequences or tutorial chapters in the user manual.

Cost/benefit analysis also can be used to develop and justify a product and process improvement plan. Early work in evaluating development effort assumed that effort was apportioned in the same way no matter what kind of project was being developed [Brooks 75]. Project development effort appeared to be distributed as follows: one-third on planning, one-sixth on coding, one-quarter on unit and integration testing, and one-quarter on system testing. Since observations from individual case studies differed with these assumptions, development efforts across projects were compared. The results indicated that the distribution of effort not only varied but was dependent on the level of difficulty (project complexity).

Investigation into the variation in development effort resulted in several models in which costs and productivity are compared. Two such models are COCOMO (which actually consists of three separate models, one for each level of detail used in the estimates) [Boehm 81] and the Pfleeger model (which allows the user to define any factor that affects development cost) [Pfleeger 91].

Other approaches to cost/benefit analysis include models for short-and long-term cost justification [Bias 94]. These approaches include a basic framework [Mayhew 94], a business case approach [Karat 94b]; and approaches from three perspectives (vendor, internal development project, and contractor) [Dray 94] [Ehrlich 94] [Mauro 94].

Usability Engineering

Most experts agree that usability-related process and product improvement can be achieved by performing *any* user interface development activity [Hix 93] [Nielsen 93b]. Some mappings between specific user interface development activities and individual life cycle phases and the associated software engineering activities have been suggested [Curtis 92] [Mantei 88]. However, little guidance focuses on the selection of individual user interface development activities or team member roles and skills appropriate for a specific project. Current recommendations for developers beginning a usability engineering program focus in three areas: discount usability engineering, comparison of user interface development activities, and cost/importance analysis.

The "discount usability engineering" method utilizes a small set of less expensive user interface development activities to reduce costs associated with user interface development [Nielsen 89] [Nielsen 93b] [Nielsen 94a]. It is suggested that the activities listed below be used during early stages of development when prototypes are available:

- user and task observation,
- scenarios,
- simplified thinking aloud, and

- heuristic evaluation.

Observing users performing work-related tasks at customer sites [Holtzblatt 93], and the use of scenarios [Carroll 90] [Carroll 92] are techniques used prior to and during system design. Simplified thinking aloud and heuristic evaluation are used to evaluate the level of usability in a system's user interface. Although heuristic evaluation has been found to be useful as an inspection method, greater success is achieved when usability experts and/or domain experts perform the evaluation [Nielsen 94b].

Some recent research has focused on comparing and contrasting user interface development activities in an effort to provide developers with sufficient information to guide the selection and performance of individual activities. These efforts have examined individual activities, types of activities (e.g., inspection methods versus user testing methods), and the way in which those activities can be performed. Development activities were examined from the following perspectives: types of usability problems, individual evaluator versus groups of evaluators, and evaluator expertise. The results of these studies are outlined below.

Inspection methods uncovered types of usability problems different than those discovered during user testing [Desurvire 94] [Karat 94a] [Nielsen 93b] [Nielsen 94]. Analysis indicates that serious usability problems can be overlooked when inspection methods are used. These problems often are classified as minor problems, but result in a high level of frustration for the user. In addition, inspection methods do not enable developers to predict final system acceptance or user preferences. However, these methods do provide developers with an inexpensive way to detect some problems prior to implementation and permit quick (and relatively inexpensive) evaluation of alternative designs or alternative products for in-house use [Brooks 94].

Some research has focused on comparing usability problems detected by groups of evaluators with usability problems detected by individual evaluators. Groups of evaluators often obtain more reliable results than individual efforts at evaluation and tend to catch false problem reports [Karat 92]. However, groups do not always catch as many unique problems as a set of independent evaluators [Nielsen 94]. More work is needed to determine if specific activities are more effectively performed by individuals or groups.

Several studies examined the impact of evaluator expertise on detected/predicted usability problems [Desurvire 94] [Nielsen 93b]. Although non usability experts identified some problems, better results were achieved by usability specialists. The best results were achieved by individuals who were experts in both the application domain and usability.

One study compared various usability methods by asking thirteen usability specialists to rate the impact of individual methods on the level of usability that could be achieved in the final product [Nielsen 93b]. The top five methods were:

- iterative design,
- task analysis,
- empirical tests with real users,
- participatory design, and
- visit customer sites prior to design.

Although this study provides general guidance for developers by indicating which activities were most useful, the conclusions that can be drawn are limited. First, various developmental factors, e.g., organizational culture, were not taken into account. Second, only usability specialists were surveyed and no indication of how much the level of expertise impacted the conclusions was provided. Third, the survey did not differentiate between participatory design activities or task analysis activities. Although the survey results do provide guidance as to the category of activity, developers would still have to decide which specific activity, within a category, would be most appropriate. Fourth, a high statistical correlation existed between the ratings and the extent to which the methods were actually used during development.

Cost/importance analysis is a product improvement technique which is used to prioritize usability problems prior to correction [Hix 93]. This technique allows developers to distinguish problems which must be corrected regardless of the cost from those which can be corrected based on other criteria such as associated cost or impact on usability.

Developers using this method rate each problem according to the cost-to-fix and relative importance to system usability. The cost-to-fix is given by three values: low, medium, and high (which could be derived based on numerical estimates). These values reflect the level of difficulty that will be encountered by developers when the problem is corrected. The importance of the correction is evaluated similarly. A problem's importance can be assessed by a thorough analysis of videotaped user testing sessions using metrics such as the number of times a problem occurred, the delay for the user, or how widespread a problem is. If inspection methods are used, the accuracy with which the importance is estimated depends on the expertise of the developers/evaluators.

Usability problems detected on a given project are labeled and entered into a table such as the one given in Table 2.2. Based on their placement in the table, problems are selected for correction.

Table 2.2. Cost/importance table.

		Problem Importance		
		Low	Medium	High
Cost To Fix	Low			
	Medium			
	High			

Although cost/importance analysis has been used successfully, further investigation is needed to increase the benefits. In particular, studies are needed to determine if assigning costs to classes of usability problems will help developers as problems are selected for correction.

Cost/benefit analysis is also used by usability engineers to justify and develop a usability engineering program. One software engineering model discussed above (COCOMO) was extended to include user interface development activities. It is used to assess the costs and benefits of incorporating human factors in the software life cycle, and outlines both tangible (and intangible costs as well as benefits [Mantei 88].

An alternative approach to cost/benefit analysis incorporates comparison of costs and benefits associated with various sets of development activities. When the cost of discount usability engineering, discussed in Section 2.4.2, was compared with costs summarized above, it was found that the discount method substantially reduced the cost without sacrificing the overall quality of the results [Nielsen 93b] [Nielsen 94a].

A Comparison of Perspectives

Software engineers have developed strategies for product and process improvement based on defect causal analysis and defect clustering. Research is needed to determine if causal analysis can be applied successfully to the user interface. Although current research in usability problem clusters provides a good first step, more work is needed to determine appropriate categories of usability problems.

Error prediction techniques used by software engineers also need to be extended to include usability. Although usability specialists are beginning to use inspection methods to predict the presence of individual usability problems, this type of data collection is not widespread (see Section 2.4.2).

Obtaining a reliable cost/benefit model for the incorporation of human factors in the software life cycle requires large amounts of historical project data collected over a variety of projects [Mantei 88] [Thomas 95]. Thus far, information of this type has not been collected, tracked, or analyzed. Much data are needed to examine the relative costs and benefits associated with development individual activities.

2.5 Summary

Two bodies of work (software engineering and usability engineering) are relevant to this research project. Four research topics were examined in each body of work: data gathering, problem classification, data analysis, and improvement strategies. Each topic was discussed first from a software engineering perspective and then from a usability engineering perspective. Methods, techniques, and approaches used in each research community were compared. Research that is needed to extend current results to include usability were also presented.

3 CONSTRUCTION OF THE USABILITY PROBLEM TAXONOMY

Chapter 3 describes how the Usability Problems Taxonomy (UPT) was developed. Section 3.1 outlines initial data collection that occurred early in the research effort. Analysis of that data is presented in section 3.2. Section 3.3 outlines how the data and analysis were used to build the UPT. Section 3.4 examines four strengths of the UPT. The findings presented in this chapter are summarized in section 3.5.

3.1 Initial Data Collection

Four software-development organizations agreed to provide data for this research. The organizations, located in government and industry, are designated only as A, B, C, and D to protect their identities and the proprietary nature of the data. The size of the organizations ranged from large to small, as did the size of the development teams. Developers at organizations A and B had access to personnel with usability-related expertise and performed various user interface development activities early in the development process. One developer at organization C had limited experience in user interface development and attempted some user interface development activities during development. Developers at organization D focused on usability by performing user tests to identify usability problems and determine the level of usability achieved in the software system.

Five development projects were surveyed: one at A, one at B, two at C, and one at D. The systems had been under development for at least one year. Each system has a graphical user interface that allows the user to enter data or commands using the keyboard and to use the mouse to directly manipulate objects on the screen. The systems spanned several application domains. Projects B and C1 included hyperlinks for navigation. This information is summarized in Table 3.1.

Table 3.1. Participant development projects.

Project	**Type**	**Application domain**	**Hyperlinks**	**Usability expertise**
A	Government Lab	Computer-aided software engineering (CASE) tool	No	yes
B	Commercial	Operating system with on-line help	Yes	yes
C1	Contract	On-line help	Yes	limited
C2	Contract	Banking system	No	limited
D	Commercial	Network access system	No	evaluation only

The projects were designed for users with different levels of expertise. Two projects (A and D) were intended for users who are expert in the respective application domains. Projects B, C1, and C2 were developed for users with varying levels of application domain expertise. The level of computer expertise likewise varied from project to project. Three systems (B, C1, and C2) were designed for computer novices as well as computer experts, while A and D were developed for a more restricted user group. See Table 3.2.

Table 3.2. Levels of expertise of the intended user groups.

Project	**Domain expertise**	**Computer expertise**
A	Expert	Moderate to expert
B	Novice to expert	Novice to expert
C1	Novice to expert	Novice to expert
C2	Novice to expert	Novice to expert
D	Expert	Novice to moderate

Usability problems were identified (observed) on each of the five projects during user interface evaluation. The usability problems for projects A, B, C2 and D were identified on site at the development organization, those on project C1 were identified at Virginia Tech using a copy of the system under development. Developers identified the problems on projects B and D. The author and a colleague identified usability problems on projects A and C1. The author, a colleague, and developers identified problems on project C2. See Table 3.3.

Table 3.3. Location, evaluators, and methods used.

Project	Location	Evaluators	Method
A	On-site	Author and colleague	Expert evaluation
B	On-site	Developers	User test
C1	Virginia Tech	Author and colleague	Expert evaluation
C2	On-site	Author, colleague, and developers	User test
D	On-site	Developers	User test

Usability problems were identified during user testing with one user at a time, user testing with two users at a time (co-discovery learning), and expert evaluation by the author and a colleague. During the expert evaluation, three techniques were used: heuristic evaluation [Nielsen 90], task-based evaluation, and object-based evaluation (a new technique developed by the author). The steps used in the performance of each of these techniques are outlined briefly below.

Heuristic evaluation

Evaluators identified a small number of core (primary) tasks for the system under evaluation. Using the set of heuristics in Table 3.4 as a guide, evaluators identified usability problems as each core task was performed.

Table 3.4. Usability Heuristics.

Heuristic
Simple and natural dialogue
Speak the users' language
Minimize the users' memory load
Consistency
Feedback
Clearly marked exits
Shortcuts
Precise and constructive error messages
Prevent errors
Help and documentation

Task-based evaluation

For the portion of the system under evaluation, the evaluators listed all user tasks, various paths through those tasks, and possible user errors (deviations from the task). The evaluators identified usability problems encountered as they attempted each path through a task as well as possible deviations from those paths.

Object-based evaluation

Evaluators listed each type of object (e.g., buttons, title bars, scroll bars, fields, dialogue boxes, menus, icons, hyperlinks) present in the user interface. Taking one type of object at a time, evaluators examined each individual object of that type throughout the interface and identified usability problems associated with those objects.

A total of 645 usability problems were identified using the evaluation methods listed in Table 3.3. Written problem descriptions were gathered from lists of usability problems available in problem reports, notes taken during evaluation activities, a database of usability problems, and reports that contained recommendations, solutions, and user preferences in addition to problem descriptions. Since usability problems were identified using evaluation activities performed on prototypes or specific system components, the number of usability problems collected on each project (given in Table 3.5) does not reflect either the level of usability achieved in the software system or the emphasis placed on usability at that organization.

Table 3.5. The reporting techniques and number of identified problems.

Project	Reporting techniques	# Problems
A	Notes containing list of problems	91
B	Database containing problem descriptions	336
C1	Notes containing list of problems	89
C2	Notes containing list of problems	54
D	Reports containing lists of problems	75

3.2 Initial Data Analysis

This section describes how heuristic analysis was used to classify the data described in section 3.1. The section concludes with a discussion of the weaknesses of heuristic analysis with respect to usability problem classification.

3.2.1 Usability Heuristic Analysis

Heuristic analysis was used to categorize (classify) two-thirds of the identified usability problems. To classify a given problem, the problem description was compared to each of the 10 heuristics listed in Table 3.4 above to determine if the presence of the problem violated the premise of the heuristic. For example, if the user has instructed the system to perform a lengthy operation and the system does not provide any feedback for the user (e.g., message box, status bar), the problem would be classified as a feedback problem, since needed feedback was missing. The violated heuristic, feedback , would be recorded for this problem.

The meaning of each heuristic was derived from the actual words in Table 3.4 in section 3.1. This differed slightly from the original interpretation for three of the 10 heuristics [Nielsen 94b]. A fourth heuristic, help and documentation, was judged to be very vague. The interpretation for the three heuristics (speak the users' language, shortcuts, and precise, constructive error messages) used in this research, as well as the procedure used to classify a "help" problem, are discussed briefly below.

The original interpretation of the speak the users' language heuristic included mental model and metaphor issues. Mental model and metaphor problems were not included in this category for two reasons. First, these problems are not always due to the language used in the user interface. Including them in a category about language obscures much of their meaning. Second, mental model and metaphor issues are so important to usability that they need to be considered separately.

The shortcuts heuristic included hypertext links and system-provided default values. Since hypertext links are a recent innovation, much can be learned about usability problems that occur with this particular navigational technique. Therefore, the decision was made to separate these issues from other shortcut problems. Similarly, system default values were distinguished from other shortcuts such as button access to menu functionality and aliases. Problems with defaults were not adequately captured by the word "shortcuts."

The original interpretation of the precise, constructive error messages heuristic included error recovery. The interpretation used in this research did not. Although a good quality error message can help a user recover from a mistake, error recovery is about much more than reading and interpreting a message. Error recovery includes such issues as easy access to the part of the user interface in which the correction must be made. As in the two cases discussed above, error recovery was not adequately captured by the words "precise, constructive error messages."

Usability problems that occur when help or documentation was used would be marked as help and documentation problems. They would then be categorized using the remaining nine heuristics and the "other" category described below.

When a heuristic category could not be identified for a given usability problem, the problem was labeled "other." This occurred when a problem corresponded to the excluded meanings discussed above. It also occurred when the a problem could not be classified in any heuristic category (e.g., naming conventions that are not meaningful enough, the user's inability to discover direct manipulation, poorly displayed on-line help information, and the picture on an icon is not meaningful to the user).

Usability problems from four of the five projects were categorized using heuristic analysis. All of the problems identified on projects A, C1, and C2, and roughly half of the problems identified on project B, were categorized using this method. None of the problems identified on project D were categorized at this time. The problems were categorized (analyzed) by either the author or the author and a colleague as outlined in Table 3.6 below.

Table 3.6. Number of problems categorized according to heuristic.

Project	# Analyzed	Analyzed by
A	91 of 91	Author and colleague
B	172 of 336	Author
C1	89 of 89	Author and colleague
C2	54 of 54	Author and colleague
D	0 of 75	None categorized
Total	406 of 645	

The decision not to categorize all of the problems was made after 406 problems had been classified. At that point, the author assessed the progress that had been made. It was apparent that categorizing additional problems would not move the research forward and that heuristic analysis was only an interim step in the research effort (specific reasons for this determination are given in section 3.2.2). As a result, the decision was made not to examine the remainder of the problems on project B nor any of the problems on project D. This was a good decision, as usability problems that had not been previously examined were needed for the UPT reliability study that is presented in Chapter 4.

In spite of the attempt to define the heuristics clearly and precisely, heuristic analysis revealed weaknesses in the following areas: distinguishability, mutual exclusiveness, completeness, and specificity. The 10 heuristic categories did not adequately distinguish among different types of usability problems, i.e., different types of problems were classified according to the same heuristic. In addition, the heuristic categories were not mutually exclusive, i.e., individual problems were classified according to more than one heuristic. The categories were not complete, i.e., a large number of problems could not be classified according to any heuristic. And, the heuristics were not specific enough, i.e., a large number of problems were classified according to various heuristics, but were not adequately captured by those heuristics.

Each of the four weaknesses is examined below. Example problem descriptions are used to illustrate each point. The descriptions, delineated by bullets, have been sanitized to protect the proprietary nature of the data.

Distinguishability

The following two problems are both categorized as feedback problems; however, they are very different problems (the first is visual, the second is about language). The feedback heuristic does not distinguish between these two problems.

- During the Save operation, the mouse pointer shape does change from an arrow to an hourglass, however, the hourglass is too small to see.
- The feedback message "Record added successfully to database. Transaction # 1243257978783323e"."

The first feedback problem is related to non-message feedback and occurs because the hourglass is not large enough to provide feedback information. The second feedback problem occurs because the message contains unnecessary information (transaction identification number).

The next two problems are both categorized as simple and natural dialogue problems. Although they are very different problems (the first is a visual, layout problem, the second is about task mapping), the simple and natural dialogue heuristic does not distinguish between these two problems.

- The user must enter data in 10 fields in a dialogue box. Although the fields do not need to be completed in a specific sequence, several users have complained that they are not arranged on the screen in the order in which they would prefer to complete them.
- In adding a variable to a mathematics equation, the user progressed through many dialogue boxes, each of which asked the user to enter only one piece of information.

The first problem is related to the visual layout of objects on the screen and occurs because the arrangement of those objects is not natural to the user. The second problem occurs because the user task was not mapped properly to the system, i.e., unnecessary steps were added to the user task.

<u>Mutual exclusiveness</u>

The following two problems illustrate that the heuristic categories are not mutually exclusive, i.e., a specific usability problem is correctly categorized in more than one heuristic. Each of the following problems is classified according to two heuristics.

- Users did not know what the "OK" button meant.

This problem is classified as a clearly marked exits problem as well as a simple and natural dialogue problem. The "OK" button is used to close a window and is considered to be an exit mechanism. It is also a simple and natural dialogue problem in that the user does not know how to complete the current step in the task. Consequently, the user cannot continue to either the next step in the current task or begin the next task.

- Users do not want to use a menu to restore a window to its normal size from its iconized state, i.e., users want window control buttons available on iconized windows.

This problem could be classified as a consistency problem (iconized window features are not consistent with features available in non iconized windows), and as a shortcut problem since shortcuts (control buttons) are not provided on the iconized windows.

Completeness

In addition to the problems that fit the excluded interpretations described earlier in this section, many problems could not be classified according to any heuristic. The following list provides examples of problems that cannot be classified using the heuristic categories.

- The user wants to delete tables and entries in those tables.
- Once a variable's name has been saved, it cannot be changed.
- The font used on the column headers is too small for users to read easily.
- Users did not know they could select a word in a text entry field and type over it. They backspaced over individual characters and retyped the word correctly.
- When a new row was added to a table too large to be entirely visible in a window, the user had to scroll to find the newly added row.

The first three problems are related to missing system functionality, the inability of the user to reverse an action, and the appearance of a text object, respectively. The last two are about the inability of users to extend their current knowledge of direct manipulation to a text object, and missing automation.

Recall that four types of problems normally classified in the speak the users' language, shortcuts, and precise, constructive error messages heuristics were placed in the "other" category because of the interpretation used in this research. While these four types of problems did contribute to the large number of problems in this category, many other types of problems were also included. As will be discussed in section 3.3.1, the "other" category contains a total of 22 types of problems (18 in addition to these four). This observation is made to provide perspective for the discussion of completeness, i.e., regardless of the interpretation of the heuristic categories (original or modified), the categories are not complete.

Specificity

Many usability problems can be classified according to one or more heuristic category; however, those heuristics do not adequately capture the actual usability problem. The first three problems illustrate this weakness with respect to the consistency heuristic, the fourth demonstrates this weakness with respect to the feedback heuristic.

- Most of the on-screen instructions in the Save window were formatted in 12 point, bolded font. The user had trouble reading the last paragraph that was formatted in 7 point font, plain text.
- Default values in data entry fields are available in some windows but not others.
- The XXX window has a white background, but is not editable (the user interface is not consistent with specifications which states that a white background implies editability).

These three problems are all classified as consistency problems. However, this heuristic does not capture or describe the actual problems. In the first case, while the 12 point font size is not consistently applied to all on-screen text, this problem is actually about a font size that is too small to be read easily. The second problem indicates that default values are not consistently placed in all data entry fields; however, within a specific window, it is also a missing default values problem. The third problem describes an inconsistency between the specifications and the user interface. Given the white background, the problem is better described as a misleading visual cue.

- Although some feedback was present to indicate tabbing from field to field, the user lost the sense of where she was on the screen.

This problem is classified as a feedback problem (the feedback is present but inadequate since it is not noticeable enough). The feedback heuristic does not capture the entire problem; the problem is also about user's ability to navigate within a window.

3.2.2 Conclusions Regarding Heuristic Analysis

Heuristic analysis was investigated to determine if it could provide the basis for a taxonomic model of usability problems that could be used in both product and process improvement efforts. It was concluded that heuristic analysis would not contribute directly to the goal of this research (see section 1.3). This section outlines the reasons for this decision.

The heuristic categories are not an effective approach to building a taxonomy of usability problems. The categories do not have four properties: distinguishability, mutual exclusiveness, completeness, and specificity (see section 3.2.1). Distinguishability is critical for developers to who want to use a taxonomy to characterize, and differentiate among, various usability problems. Mutual exclusiveness is important for statistical analyses that can be used to analyze a set of usability problems using the taxonomic categories. Completeness is necessary so that developers can categorize *all* identified problems. Specificity is equally important for developers who use a taxonomy to expose the actual problem contained in a description.

In addition, the heuristics are not a cost-effective classification tool. Heuristic analysis is not only time consuming but arduous. This is primarily due to the fact that the heuristics are prescriptive. Since problem statements are descriptive, attempting to classify them using a set of prescriptive categories requires a change of perspective for each problem. The heuristics were derived from a set of usability guidelines. The guidelines were developed to help user interface designers avoid usability problems or prevent them from appearing in the user interface. It is difficult to use strategies for preventing usability problems to describe those problems when they occur.

Although heuristic evaluation has been used to identify usability problems and improve product usability, the heuristic categories do not support other aspects of product improvement. The heuristics have not been shown to help developers write higher quality descriptions. Since the heuristics categories are not mutually exclusive, some types of statistical analysis of data categorized by heuristic are difficult and hard to interpret. The heuristics also do not provide an adequate framework in which to compare and contrast usability problems detected on a given project. As a result, they do not help developers design and compare both local and global solutions.

In addition, the heuristics do not provide an effective approach to usability-related process improvement. The heuristic categories are hard to interpret, i.e., the categories cannot be linked to development context. Recall the two simple and natural dialogue problems described under distinguishability in section 3.2.1. The simple and natural dialogue heuristic does not provide any guidance for addressing those problems. The first problem would be best corrected by a graphic designer, the second problem by a developer who is either proficient in task analysis or requirements analysis, or is an expert in the application domain. If, for example, 25% of the problems detected on a given project are classified as simple and natural dialogue problems, this heuristic does not provide any guidance with respect to the correction or prevention of those problems. Similarly, no clear association exists between any of the individual heuristic categories and either team roles and skills or development activities, methods, and techniques.

3.3 Development of the Usability Problem Taxonomy

This section describes how the heuristic analysis described in section 3.2 was used to build the Usability Problem Taxonomy (UPT). Section 3.3.1 describes the process used to examine problems categorized within each heuristic category. Section 3.3.2 outlines how clusters of problem types were incorporated directly into the UPT.

3.3.1 Revisiting Usability Problem Data: Subtypes

Although the weaknesses of heuristic analysis described in Section 3.2 precluded its use as the basis for a taxonomic model of usability problems, the classification proved to be a productive first step in the construction of the Usability Problem Taxonomy. The problems categorized within nine of the 10 heuristic categories were examined closely to identify if commonalities and similarities existed among the problem descriptions (recall that the tenth heuristic, help and documentation, was used only to mark a problem). The problems placed in the "other" category (that could not be classified according to any heuristic) were also examined.

Commonalities and similarities among problem descriptions were identified, and the problems within each category were divided into various subtypes (subtype refers to a group or cluster of usability problems within a category). A total of 74 subtypes were identified. The subtypes for each heuristic are given in Tables 3.7 - 3.15. Table 3.16 contains subtypes identified for the "other" category.

The simple and natural dialogue category contains seven types of problems, listed in Table 3.7. The subtypes vary from screen clutter (1), screen layout (5) and unnecessary buttons (7), to wording (2) and task mapping (3, 4, 6).

Table 3.7. Subtypes for simple and natural dialogue.

Subtype	Simple and natural dialogue
1	screen clutter (dialogue too complex)
2	wording affects task completion, e.g., not knowing what the OK button does
3	poor overall mapping to users' tasks
4	poor mapping to sequence of user subtasks
5	inadequate screen layout (grouping related features, use of color, location of icons)
6	extra, unnecessary steps in task on system
7	unnecessary features and extra buttons

Eight subtypes were observed for speak the user's language. These subtypes are given in Table 3.8 and were concerned with language used in the user interface. The subtypes focused on problems that occurred when clear, precise, application domain terms were not used.

Table 3.8. Subtypes for speak the users' language.

Subtype	Speak the users' language
1	system terms not users' terms
2	nonstandard meanings for words
3	enforced naming conventions (limited user-defined word length)
4	laymen's terms, not domain terms
5	threatening terms
6	bossy terms
7	terms not meaningful
8	confusing terms

Table 3.9 contains the five subtypes for minimize users' memory load. These subtypes described problems that were associated with the system's inability to help the user complete the task without complications. For example, users progress more easily through a sequence of subtasks if dialogue elements are displayed rather than hidden, if the system describes the requirement format for input, makes generic commands available, and ensures that object names are unique.

Table 3.9. Subtypes for minimize users' memory load.

Subtype	Minimize users' memory load
1	dialogue elements not displayed to user
2	system does not describe required format of user input or provide example
3	need generic commands (can be applied in multiple circumstances with similar results)
4	more than 1 item named too similarly, names on buttons and title bars do not provide enough information
5	non displayed items (user must remember pathname, which fields are required and optional)

Consistency issues arise frequently in the user interface, resulting in eleven subtypes (see Table 3.10). The subtypes vary from the way the objects in the user interface look (1) and where they are placed (2), to system responses (3 and 4) and naming conventions (5). In addition, consistency issues arise with respect to object affordances

(6), interpretation of similar types of feedback (7), and the types of information presented in an on-line help document (8). Subtypes 9, 10, and 11 have to do with the a consistent approach to task structure (both within the given system and as it is compared with other application software).

Table 3.10. Subtypes for consistency.

Subtype	Consistency
1	object look
2	object placement
3	mouse click results
4	system responses (to menu item selection, tab key, OK button)
5	naming (e.g., menu item and associated dialogue box title bar)
6	object affordances (and actions on those objects)
7	meanings of similar feedback
8	type of information presented (as in a glossary or page pane)
9	task structure
10	users' task expectations and system
11	system with user interfaces of other applications

Seven feedback subtypes are presented in Table 3.11. This heuristic contains problems that occur when needed feedback is missing (1), when feedback is present but inadequate in some way (2, 3, 4, 6, and 7), or feedback is present but unnecessary (5).

Table 3.11. Subtypes for feedback.

Subtype	Feedback
1	needed feedback missing
2	feedback present but slows user down
3	feedback present but obscure (difficult to understand), insufficient
4	feedback present but misleading (tells user the wrong thing)
5	feedback present but unnecessary feedback
6	feedback present but not adequately persistent
7	feedback present but disturbing, threatening

Three subtypes were observed for clearly marked exits. These subtypes, listed in Table 3.12, focused on whether or not the system had an undo feature (1), provided clear screen exits (2), and allowed the user to exit without an action (e.g., when a window does not have a cancel button the user is forced to take an action to exit the window).

Table 3.12. Subtypes for clearly marked exits.

Subtype	Clearly marked exits
1	missing undo
2	unclear screen exit
3	no exit without action or without prompting for no action (no cancel button)

Shortcut problems resulted when needed shortcuts were missing (1) or when the shortcuts were present but inadequate (2). An example of a missing shortcuts problem would be not having button access to a frequently used function embedded in a menu. An example of problem with a shortcut in the user interface would be a key sequences for menu item that is not intuitive, e.g., control p for copy. See Table 3.13.

Table 3.13. Subtypes for shortcuts.

Subtype	Shortcuts
1	shortcuts missing (abbreviations, function keys, double-clicking, button access to system functions, macro facilities, aliasing capabilities, reusing interaction history)
2	shortcuts present but not good

The subtypes identified for precise, constructive error messages are listed in Table 3.14. These subtypes describe problems with language (1, 2, 3, and 4), those that may impede the user's progress through a task (5), and those that do not facilitate error recovery (6). Note that the first subtype below is similar to speak the users' language (7) which was about language that was not meaningful.

Table 3.14. Subtypes for precise, constructive error messages.

Subtype	Precise, constructive error messages
1	unclear language
2	imprecise, rather than general or vague
3	does not help user solve the problem
4	impolite, blames user
5	multi-level messages (user can click for more information)
6	no link from error message to location in help system

Three subtypes were noted for prevent errors (see Table 3.15). Two very specific subtypes (1 and 3) are about the lack of confirmation for user actions with consequences and preventing errors that occur when the user clicks a mouse button. Subtype 2 is a more general category of problems in which the system did not help guide the user through the sequence of subtasks.

Table 3.15. Subtypes for prevent errors.

Subtype	Prevent errors
1	no user confirmation for actions with consequences
2	system does not guide user during task (shift user focus)
3	closeness of mouse clicks, extraneous clicks on buttons, mouse movement off menus

Commonalities among problems that could not be classified according to the heuristics (other) yielded the 22 subtypes given in Table 3.16. These subtypes varied from issues with available system functionality (1 and 6) and naming conventions that were not a speak the users' language problem (3) to navigation (7) and direct manipulation problems (8). Several subtypes had to do with on-line help information (14, 15, 16, 17, 18, and 19). As mentioned above, error recovery was included in this list (10). The remaining subtypes (2, 4, 5, 9, 11, 12, 13, and 22) are concerned with whether or not the system helps the user complete the task in an efficient manner. Note that subtypes 4, 10, 13, and

15 correspond to the four types of problems excluded from three heuristic categories according to the interpretation for this research (see section 3.2.1).

Table 3.16. Subtypes for other.

Subtype	Other
1	flexibility - missing functionality
2	system slows user down, impedes performance (not directly involved in a specific, core user task
3	naming conventions (not a speak the user's language problem, not required for task completion, wording is just not meaningful enough)
4	reasonable defaults and reasonable system responses to an action, visibility
5	look and feel of the interface - placement, arena
6	flexibility - some functionality present, perhaps inadequate number of choices, ignoring opportunity to make button responses appropriately context sensitive
7	navigation within a screen (field to field)
8	direct manipulation (user was unaware of, did not know about, discoverability)
9	menu organization (grouping features in a menu)
10	error recovery
11	organization of search topics in help search screen
12	visualness (aesthetic quality), e.g., font size, appearance, tab appearance
13	mental model and metaphors
14	missing information
15	missing hypertext link
16	organization and structure of information in on-line help text
17	wording (misleading on-screen instructions)
18	duplicate information
19	unnecessary, extraneous information displayed
20	icon (picture) design, non word names which are not meaningful enough to the user
21	users have trouble with new ideas - context menus
22	locus of control with user

3.3.2 Clusters Of Problem Subtypes: The Usability Problem Taxonomy

In the second step of an iterative process, the 74 subtypes were likewise examined for commonalities and similarities. Commonalities were found to span multiple heuristic categories. For example, speak the users' language problems could also be found in feedback messages, error messages, on-screen text (instructions), and problems with object names and labels. Likewise, the appearance of screen objects, the layout of those objects, and non-message feedback were all focused on the appearance of the user interface. Shortcuts and the user desiring an alternate path through a task as noted in "other" (7) were also determined to be similar subtypes.

In addition, divisions between subtypes within a category were also investigated. For example, consider feedback and speak the users' language (Tables 3.11 and 3.8, respectively). Feedback issues focus on two areas: problems with non message feedback and problems with feedback messages. Speak the users' language problems are divided into problems that occurred with names and labels and problems with words used in prose or text. The decision was made to keep such divisions separate.

Twenty-one problem types were identified. These problem types, given in Table 3.17 below, emerged as clusters or groupings of the 74 subtypes. An important observation was that consistency issues occur in many different contexts. As a result, consistency was included in *each* problem type.

The 21 problem types were again examined for clustering and were grouped into five primary categories (Visualness, Language, Manipulation, Task-mapping, and Task-facilitation). These primary categories are given in the "5 primary categories" column in Table 3.17 below.

Note that each primary category contains specific problem types. Visualness usability problems are about the user's ability to see objects in the user interface. Language problems focus on the user's ability to understand the words (text objects) that are used in the interface. Manipulation is concerned with the user's ability to understand visual cues and directly manipulate user interface objects. Task-mapping problems focus on the structure (mapping) of the user task on the system. Task-facilitation refers to the systems' ability to help the user follow the task structure and complete the task.

Table 3.17. The five primary categories with associated problem types.

5 primary categories	Problem types
Visualness	Object (screen) layout
	Object appearance
	Object movement
	Presentation of information/results
	Non-message feedback
Language	Naming/labeling
	Feedback messages
	Error messages
	Other system messages
	On-screen text
	User-requested information/results
Manipulation	Visual cues
	Direct manipulation
	Physical aspects
Task-mapping	Interaction
	Navigation
	Functionality
Task-facilitation	Alternatives
	Task/function automation
	User action reversal
	Keeping the user task on track

The five primary categories were again grouped into two components: the artifact component and the task component. The artifact component contains three categories: visualness, language, manipulation. The categories in the artifact component focus on usability problems users experience with artifacts (objects) in the user interface. The task component contains two categories: task-mapping and task-facilitation. The task component focuses on usability problems users experience as they perform tasks on the system (i.e., problems that occur as the user moves *through* a task). The two components correspond to the two dimensions of a usability problem. Together, the two components comprise the Usability Problem Taxonomy (UPT). Complete descriptions of each component and the five primary categories can be found in Appendix A. These descriptions, and additional definitions, are provided at a web site (contact the author for the URL).

The hierarchical structure of the Usability Problem Taxonomy is illustrated in Figure 3.1. The figure has four levels. The first level corresponds to the starting point and contains the two component names. From the starting point, usability problem categories branch to the right. Level 2 contains the five primary categories (this is an important level in the UPT reliability study presented in Chapter 4 and in the association with development context discussed in Chapter 5). Levels 3 and 4 contain the 21 problem types and two additional subcategories. The two subcategories are other wording and cognitive aspects (recognition). Other wording was added to distinguish prose and text problems from those that occur with names and labels. Cognitive aspects (recognition) was added to separate problems that were best described as physical aspects of direct manipulation from those that pertained to visual cues and a general knowledge of direct manipulation techniques.

The problem types in levels 3 and 4 of Figure 3.1 actually contain lists of specific examples of usability problems in that category. These detailed lists provide feedback to the classifier that a problem has been classified correctly. These lists are not included in the figure, but can be viewed at the URL given above.

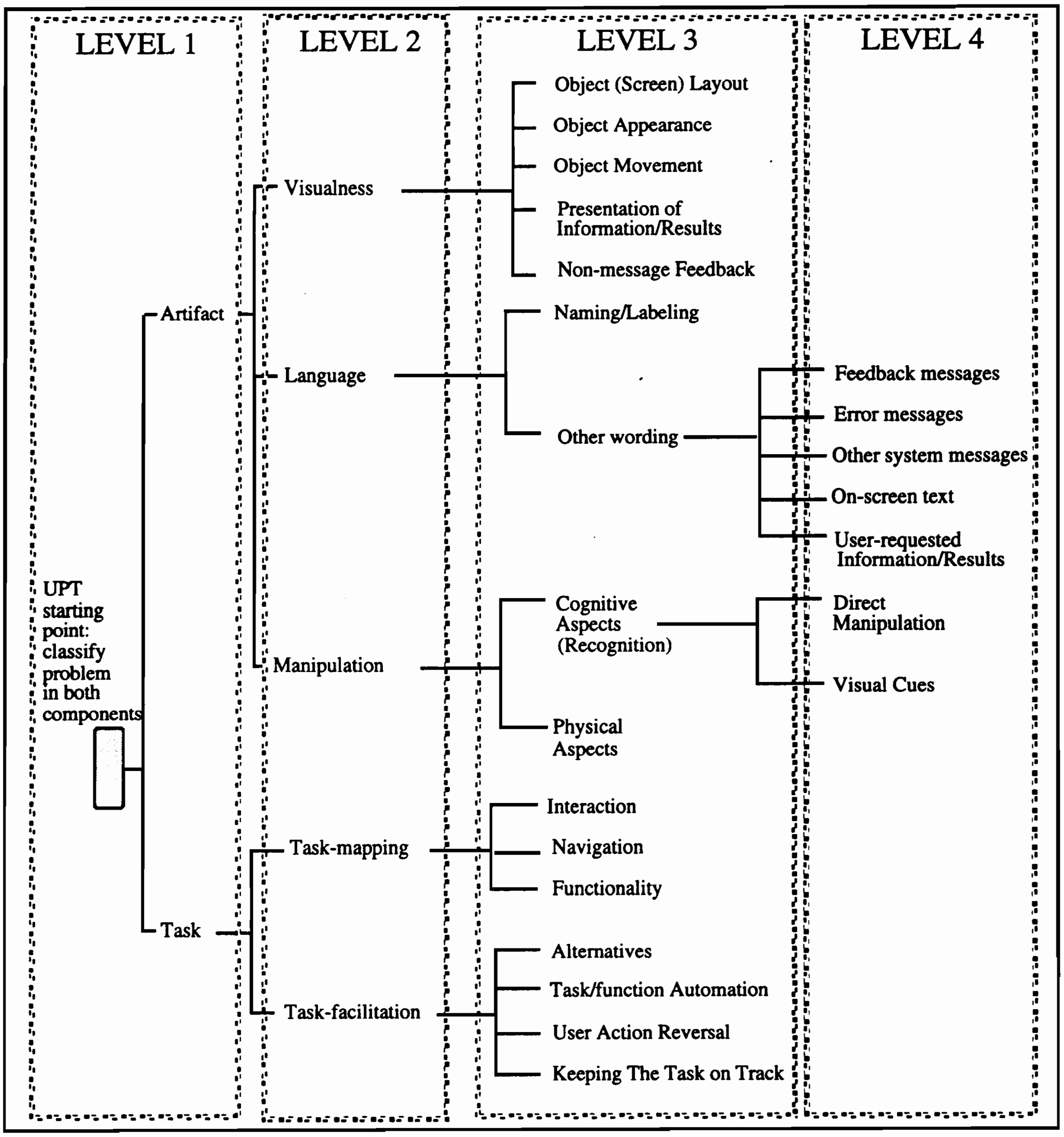


Figure 3.1. Hierarchical structure of the Usability Problem Taxonomy.

The Usability Problem Taxonomy (UPT) is a taxonomic model, based on problem characteristics, in which usability problems can be classified. Although some words in the UPT reflect user interface guidelines and heuristics, the taxonomy is not based on these guidelines; it is based on problem descriptions. In addition, the UPT is not a cause-and-effect model. While usability problems do have an effect on the user and task performance, this model categorizes problem descriptions, not the impact on task performance.

Each usability problem is classified in the artifact component as well as the task component. This concept is based on the premise that usability problems have two dimensions (an artifact dimension and a task dimension). Hence, the artifact component and the task component are not intended to be mutually exclusive but are used together to describe two different aspects of an individual problem. Although the task-artifact approach has been used during design [Carroll 91] [Carroll 92] [Preece 94], it has not been applied to usability problems.

Within each component, however, the categories *are* mutually exclusive, i.e., a usability problem is classified either as a visualness, language, or manipulation problem in the artifact component, and a task-mapping or task-facilitation problem in the task component. Extending that idea to levels 3 and 4, once a problem has been classified in a primary category, it can only be classified in one subcategory or problem type. For example, if a problem is classified in the artifact component as a visualness problem, then it is classified as either an object layout, object appearance, object movement, presentation of information/results, or non message feedback problem. Similarly, if the same problem is classified as a task-mapping problem, then it can only be classified as an interaction, navigation, or functionality problem.

Various factors affect the level within each component at which a problem can be classified. These factors as well as instructions for problem classification are discussed in section 4.1.

3.4 Strengths Of The Usability Problem Taxonomy

This section examines the UPT with respect to the four issues described in section 3.2.1: distinguishability, mutual exclusiveness, completeness, and specificity.

Distinguishability

The UPT categories do distinguish among different types of usability problems, i.e., different types of problems are not classified in the same category (level 2) or in the same problem type (levels 3 and 4). However, the associated definitions may be elaborated to ensure that classifiers with minimal training or expertise will be able to use the UPT reliably (see Chapter 4).

Mutual exclusiveness

The UPT categories within each component are defined to be mutually exclusive, hence the difficulties caused during statistical analysis of the heuristic categories do not surface with the UPT-based analysis. Individual problems are classified only in one artifact category and in one task category.

Completeness

The categories in level 1 of the Usability Problem Taxonomy (UPT) are arguably complete. The user interface is comprised of artifacts which are used to complete user tasks. Users experience problems as artifacts (user interface objects) are viewed, read, and manipulated. Similarly, task-related problems occur because the mapping was inadequate or because the system was not programmed to help the user complete the task.

The categories in the UPT were developed empirically and are based on problem characteristics noted in the 406 problem descriptions that were examined during the heuristic analysis. All problems could be classified. No new categories were needed in level 2 to classify the full set of 645 usability problems (recall that 239 problems were not examined prior to building the taxonomy). It is possible, however, that since only five application domains were surveyed, analysis of usability problems identified in very different application domains (e.g., virtual reality or software systems designed for the physically challenged) may require the introduction of new problem types (levels 3 and 4).

Specificity

Since each UPT problem type contains many examples of usability problems in that type, the categories are very specific. However, additional examples may be added to further clarify the type of problems contained in each category.

3.5 Summary

Five development projects at four software-development organizations were surveyed during this research project. A total of 645 usability problem descriptions were collected from projects that spanned five different application domains. The projects were intended for users with varying levels of application domain and computer related expertise.

A subset (406) of the 645 usability problems were classified initially according to the heuristic(s) violated. During this process, it was concluded that heuristic classification was not an effective approach to building a taxonomy of usability problems. This conclusion was reached for several reasons:

- four weaknesses related to using the heuristics as a classification scheme were identified,
- the heuristic categories were determined to be prescriptive not descriptive, and
- no clear associations exist between the heuristic categories and development context.

An iterative process was used to reexamine the usability problems that had been classified according to the heuristics. Seventy-four heuristic subtypes were identified and again combined into 21 problem types. These problem types clustered in five primary categories. The five primary categories were grouped into two components. The two components were merged to form the Usability Problem Taxonomy. The UPT is descriptive rather than prescriptive and does not exhibit the four weaknesses discussed for heuristic analysis.

4 THE UPT AND PRODUCT IMPROVEMENT

This chapter examines how the Usability Problem Taxonomy can be used in various product improvement activities. Section 4.1 presents a study that showed that the UPT could be used to classify usability problems reliably. Section 4.2 examines UPT classification and illustrates how the UPT can be used during problem reporting to improve problem description and, during problem analysis, to organize usability problems and suggest local and global solutions. A brief summary is given in section 4.3.

4.1 The Reliability Of The UPT And Its Use In Problem Classification

This section discusses a study that was undertaken to assess the reliability with which the UPT can be used to classify usability problems. Section 4.1.1 describes how the study was conducted. Section 4.1.2 presents the results of the study. Section 4.1.3 contains remarks about the results of the reliability study. Section 4.1.4 discusses the responses obtained on a UPT user satisfaction survey completed by each study participant.

4.1.1 The UPT Reliability Study

The purpose of this study was to show that the UPT can be used reliably by interactive software developers and evaluators to classify usability problems. For the UPT to be considered a reliable classification tool, different classifiers must classify a given usability problem (or set of problems) similarly. In this context, reliability implies repeatability.

Two decisions impacted the design of the UPT reliability study. The first decision was to seek individual participants with a variety of backgrounds (industry, government, or academic) as well as substantial experience in software and/or user interface engineering. The second was to have the participants classify real-world usability problems selected randomly from the surveyed projects.

Each classifier was given a packet of materials that contained UPT documentation, instructions, and the list of problems. Each was asked to classify the problems in the artifact component and the task component of the UPT and then return the responses to the author. Classifiers participated at a location and time(s) of their choosing.

Classifiers

Seven classifiers from industry and academic development environments were selected to participate. They had varying levels of experience in industry, government, and academic environments. Classifier experience (in number of years) is summarized in Table 4.1.

Table 4.1. Classifier experience: number of years in industry, government, and academic environments.

Classifier	Industry	Government	Academic	Total
1	5	-	20	25
2	10	3	15	28
3	11	-	6	17
4	10	-	8	18
5	1	5	12	18
6	-	-	20	20
7	5	10	-	15

With the exception of classifier 6 (who had academic experience), each participant had experience in both software and user interface development (see Table 4.2). In addition, five classifiers use guidelines and heuristics regularly; the remaining two never use guidelines nor heuristics. At the onset of the study, three classifiers had limited exposure to the UPT; four had no prior experience with the UPT.

Table 4.2. Classifier experience: number of years in software and user interface development.

Classifier	Software Development	User interface Development
1	5	10 - 15
2	5	12
3	11	3
4	9	1
5	12	4
6	5	-
7	7	6

Classification Materials

The materials sent to each classifier contained the following items: a tutorial, sample problem classifications, Figure 3.1, a glossary of terms, and a list of 20 usability problems. A brief description of the user task and type of system was included with each usability problem description. Six of the seven classifiers used the World Wide Web version of the UPT (contact the author for the URL). One classifier was unable to access the World Wide Web and used a paper copy of the UPT. No time limit was placed on the classification. The classifiers could classify the problems in any order, revisiting any given problem as often as needed (potentially to modify an original classification). The goal was to obtain the best classification possible (in the judgment of each classifier).

The List of Usability Problems

A set of 20 usability problems was randomly selected from the 645 problems that had been collected from the five surveyed projects. The decision to use a set of size 20 was a compromise between two factors. The sample had to be large enough to satisfy criteria for statistical significance, yet small enough to limit the time spent by each classifier so that their participation in, and completion of, the reliability study could be assured.

The number of problems randomly selected from each project for this study approximated the percentage each project contributed to the total number of problems as shown in Table 4.3. For example, consider project A. Three problems were randomly selected from the 91 problems collected on project A (see columns "# for sample" and "# collected"). The three randomly selected problems represent 15% of the 20 sample problems (see column "% of sample"). This closely approximates the contribution of project A to the total number of collected problems, i.e., 91 of 645 or 14% given in column "% collected."

Table 4.3. Distribution of problems over the five projects.

Project	# For Sample	% Of Sample	# Collected	% Collected
A	3	15	91	14
B	11	55	336	52
C1	1	5	89	14
C2	2	10	54	8
D	3	15	75	12
Total	**20**	**100**	**645**	**100**

The exception to this selection rule occurred on project C1. Since project C1 was an on-line help system and project B contained an on-line help system (recall the system descriptions in section 3.1), the decision was made to limit the number of problems selected from project C1. This ensured that the on-line help application domain was not over-represented in the sample. Although project C1 contributed 14% of the total number of collected problems, only one problem was randomly selected from project C1 which corresponded to 5% of the sample. This decision impacted the number of problems randomly selected from projects B, C2, and D. Note that the percentage of sample problems selected from projects B, C2, and D in column "% of sample" are somewhat higher than the respective values in the "% collected" column.

The 20 sample problems were divided into two sets of size 10. The first set of ten problems was selected from the group of problems examined during heuristic analysis prior to building the UPT (see section 3.2.1). These problems were selected from projects A, B, C1, and C2 and their portions of the sample problems are given in the "old" column in Table 4.4 below. The remaining 10 problems had not been previously analyzed and were selected from projects B and D (see the "new" column in Table 4.4).

Table 4.4. Distribution of problems over the five projects.

Project	Old	New	# Of Problems
A	3	0	3
B	4	7	11
C1	1	0	1
C2	2	0	2
D	0	3	3
Total	**10**	**10**	**20**

Two factors impacted the decision to select problems from both groups. Problems were selected from the "old" group to ensure that a variety of application domains were represented. Problems were selected from the "new" group to demonstrate that UPT reliability did not depend on the whether or not the problems had been previously examined.

The same selection rule was applied to each set of 10 problems, i.e., the number selected from each project for the old group (and the new group) should approximate the contribution of that project to the total. Recall from the discussion above that the one exception to this rule was project C1 (the on-line help system). The number of problems selected from each project is given in the "# old (sample)" column in Table 4.5 below. These numbers approximate the contribution of each project to the total number of previously analyzed problems (see the percentages given in columns "% old sample" and "% old collected").

Table 4.5. Distribution of previously examined problems.

Project	**# Old (Sample)**	**% Old (Sample)**	**# Old Collected**	**% Old Collected**
A	3	30	91	22
B	4	40	172	42
C1	1	10	89	22
C2	2	20	54	13
Total	**10**	**100**	**406**	**99**

Similarly, the 10 new problems were randomly selected from projects B and D. Seven problems were selected from project B. Three were selected from project D. Note that the respective percentages given in columns "% new (sample)" and "% new collected" in Table 4.6 are approximately equal.

Table 4.6. Distribution of new problems.

Project	**# New For Sample**	**% New (Sample)**	**# New Collected**	**% New Collected**
B	7	70	164	69
D	3	30	75	31
Total	**10**	**100**	**239**	**100**

The 20 problem descriptions were used as they were reported, i.e., they were modified only to ensure the confidentiality of the surveyed organizations and products. Some contextual information was also provided such as a brief description of the user task that was being performed when the problem occurred, the type of user interface (e.g., GUI, command line, form, menu), and the type of software system (e.g., word processor, database, on-line help). Dr. J. A. N. Lee verified that the changes required to protect confidentiality did not modify the meaning, intent, amount, or quality of the information available in the descriptions.

4.1.2 Results Of The UPT Reliability Study

The kappa statistic, κ, was used in this study to assess the level of agreement among classifiers in the artifact component as well as in the task component Figure 3.1 [Cohen 60] [Fleiss 71]. The kappa statistic was selected due to the categorical nature of the data. Although the chi-square statistic measures association, for this work, the kappa statistic was more appropriate as it measures agreement. The level of agreement was computed only at level 2 of Figure 3.1 The decision to restrict the analysis to this level was due to the limited sample size (20 problems). Although the number of observations in this study were insufficient to calculate kappa at levels 3 and 4, showing UPT reliability at these levels is planned as a future project (see section 7.1).

Summary of UPT Classifications

To compute κ for the data collected in this study, two tables were constructed (one for the categories in the artifact component and one for the categories in the task). The tables contain the counts of the number of problems classified in each category, i.e., each numeric entry in Tables 4.7 and 4.8 represents the number of times an individual problem was classified in a specific category by the seven classifiers. Using problem 1 as an example, first consider the artifact classifications in Table 4.7. One of seven classifiers classified problem 1 as a visualness (V) problem, five classified it as a language (L) problem, zero classified it as a manipulation problem (M), and one was unable to classify it in any of the three artifact categories (Other). In Table 4.8, one classifier placed problem 1 in the task-mapping (TM) category, five placed it in the task-facilitation (TF) category, and one was unable to classify it in any task category (Other).

Table 4.7. Counts for the artifact component.

Counts:	Artifact				
Problem #	V	L	M	Other	Agrmt
1	1	5	0	1	+
2	0	0	6	1	+
3	6	0	1	0	+
4	3	0	1	3	
5	1	4	0	2	
6	3	0	3	1	
7	1	6	0	0	+
8	0	0	5	2	+
9	1	4	1	1	
10	0	7	0	0	++
11	0	7	0	0	++
12	6	0	1	0	+
13	0	7	0	0	++
14	2	3	2	0	
15	0	0	7	0	++
16	1	5	1	0	+
17	2	4	0	1	
18	1	6	0	0	+
19	5	2	0	0	+
20	4	0	3	0	
Total	37	60	31	12	
p_j	.264	.429	.221	.086	

There are 140 classifications in the artifact table and 140 in the task table (seven classifiers, 20 problems). The entries in the total line in Tables 4.7 and 4.8 represent the total number of classifications in each UPT category. For example, in Table 4.8, 43 of the 140 task classifications were in the task-mapping category; 62 were in task-facilitation, and 35 problems were classified in the "other" category classifications by classifiers who were unable to categorize those problems in the task component of the UPT. The p_j line in Tables 4.7 and 4.8 represents the proportion of problems classified in a given UPT category (j varies over the categories and takes the following values: v, l, m, tm, tf, and other). For example, in Table 4.8, the proportion of problems classified as task-mapping problems is 43 of 140 ($p_{tm} = .307$). Similarly, the proportion of problems classified as task-facilitation problems is 62 of 140 or ($p_{tf} = .443$). The proportion of problems that could not be classified in any task category is 35 of 140 or ($p_{other} = .25$).

The entries in the "Agrmt" column in each table indicate those problems having good agreement ("+") as well as complete agreement ("++"). Good agreement occurred when five or six classifiers placed a problem in one category. Complete agreement occurred when all seven classifiers categorized an individual problem the same way. For the artifact classifications, classifiers had complete agreement on four problems (10, 11, 13, and 15). They had good agreement on nine problems (1, 2, 3, 7, 8, 12, 16, 18, and 19). Classifiers had good agreement on six of the 20 task classifications (problems 1, 5, 8, 16, 17, and 20). Note that no problems had complete agreement in the task classifications.

Table 4.8. Counts for the task component.

Counts: Problem #	Task TM	TF	Other	Agrmt
1	1	5	1	+
2	3	4	0	
3	2	1	4	
4	3	3	1	
5	0	6	1	+
6	1	4	2	
7	0	3	4	
8	5	0	2	+
9	4	1	2	
10	3	1	3	
11	2	2	3	
12	4	2	1	
13	1	2	4	
14	4	3	0	
15	3	3	1	
16	1	6	0	+
17	1	5	1	+
18	0	3	4	
19	4	3	0	
20	1	5	1	+
Total	43	62	35	
p_j	.307	.443	.25	

In both tables, problem 1 through 10 are the "old" problems (problems that had been previously examined), 11 through 20 are the "new" problems (problems that had not been previously examined). Casual inspection of the two tables indicates that problems having good or complete agreement were located in both the old group as well as the new group of problems. For the artifact classifications (Table 4.7), five problems having good agreement were in the old group, four were in the new group of problems. One problem having complete agreement in the artifact component was in the old group, three were in the new group. For the task classifications (Table 4.8), three of six problems having good agreement were in the old group, three were in the new group. The fact that problems having good or complete agreement were spread through both groups demonstrates that classifiers were able to use the UPT reliably to classify problems in both the "old" and "new" groups.

Table 4.9 shows the number of problems having good or complete agreement in each UPT category. Note that 7 of the 13 problems with good or complete agreement in the artifact categories were classified as language problems. It is possible that the classifiers recognized language problems more easily than either visualness or manipulation problems. It is also likely that more language problems were contained in the sample than other types of problems. Similarly, five problems having good agreement in the task categories were classified as task-facilitation problems. This could be due to the classifiers' recognizing task-facilitation problems more easily. However, based on the classifiers' comments, it is more likely that the task-facilitation category was imbued with additional meaning which resulted in the number of problems classified in that category.

Table 4.9. Counts for good and complete agreement.

Counts	Artifact			Task	
Agreement	V	L	M	TM	TF
good (+)	3	4	2	1	5
complete (++)	0	3	1	0	0
Total	3	7	3	1	5

Classifiers agreed on only three of the 16 problems in both the artifact and the task classification (problems 1, 8, and 16). This results in a proportion of .19 which indicates that, on a problem-by-problem basis, good agreement in the artifact classification does not necessarily result in good agreement in the task classification. It should also be noted that the agreement on problem 1 was in the language and task-facilitation categories, agreement on problem 8 was in the manipulation and task-mapping categories, and agreement on problem 16 was in the language and task-facilitation categories. The fact that language and task-facilitation categories were involved is not surprising given the values in Table 4.9 above.

Various factors influence the level of agreement among classifiers. These factors include whether or not a problem description was vague or incomplete, classifier experience with various application domains, and familiarity with the UPT (minimal UPT training materials were made available to each classifier). Equally important is the fact that the classifiers did not observe each problem as it occurred. These factors, discussed further in section 4.1.2, can lead individual classifiers to interpret and classify problems differently.

Computing Kappa

The kappa statistic, κ, is the proportion of agreement after chance agreement is removed from consideration. The kappa statistic was applied independently to both the artifact component and the task component of the UPT. The following assumptions were made [Cohen 60]:

- the 20 usability problems are independent,
- in the artifact component, the categories within each level are independent, mutually exclusive, and exhaustive,
- in the task component, the categories within each level are independent, mutually exclusive, and exhaustive, and
- the classifiers operate independently and are a priori judged equally competent to perform the classifications.

To compute κ, let

$\bar{P}$ = the proportion of problems in which the classifiers agreed, and

$\bar{P}_c$ = the proportion of problems for which agreement is expected by chance.

Then $\bar{P} - \bar{P}_c$ represents the proportion of agreement beyond chance (the degree of agreement actually attained in excess of chance). $\bar{P}$ will be greater than $\bar{P}c$ when nonchance factors are operating in the direction of agreement. The difference, $\bar{P} - \bar{P}_c$ is positive when the proportion of agreement is greater than chance, and is negative when there is less than chance agreement.

The quantity $1 - \bar{P}c$ measures the maximum possible degree of agreement over and above what would be predicted by chance. Dividing the difference $\bar{P} - \bar{P}_c$ by $1 - \bar{P}_c$ yields the kappa statistic:

$$\kappa = \frac{\bar{P} - \bar{P}_c}{1 - \bar{P}c}.$$

Note that the upper limit of κ is 1 and occurs when $\bar{P}$ is 1. Kappa is 0 when the proportion of agreement equals chance.

<u>The Hypothesis Tests</u>

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 (standard deviation of κ): $\kappa/SE(\kappa)$. Under the null hypothesis, $\kappa/SE(\kappa)$, will be approximately distributed as a standard normal variate Z (by the Central Limit Theorem).

The following six hypothesis tests were performed.

1. Test the null hypothesis that the classifications in the artifact component are random versus the alternative hypothesis that there is more agreement than would be obtained by random classification using all 20 problems.
2. Test the null hypothesis that the classifications in the task component are random versus the alternative hypothesis that there is more agreement than would be obtained by random classification using all 20 problems.
3. Test the null hypothesis that the classifications in the artifact component are random versus the alternative hypothesis that there is more agreement than would be obtained by random classification using the 10 "old" problems.
4. Test the null hypothesis that the classifications in the task component are random versus the alternative hypothesis that there is more agreement than would be obtained by random classification using the 10 "old" problems.
5. Test the null hypothesis that the classifications in the artifact component are random versus the alternative hypothesis that there is more agreement than would be obtained by random classification using the 10 "new" problems.
6. Test the null hypothesis that the classifications in the task component are random versus the alternative hypothesis that there is more agreement than would be obtained by random classification using the 10 "new" problems.

Tests 1 and 2 focus on UPT reliability over the entire set of 20 problems. Tests 3 and 4 focus on UPT reliability for the 10 problems that had been examined previously. Tests 5 and 6 focus on UPT reliability for the 10 problems that had not been examined previously. As each test was performed, the following values were computed: $\overline{P}$, $\overline{P}_C$, κ, Z, and p (the level of statistical significance).

The results of tests 1 and 2 are summarized in Table 4.10.

Table 4.10. Statistics for all 20 usability problems.

All 20 problems	$\overline{P}$	$\overline{P}_C$	$\overline{P} - \overline{P}_C$	κ	Z	p
1. Artifact	.588	.31	.278	.403	9.776	.000
2. Task	.414	.353	.061	.095	2.306	.011

For test 1, the null hypothesis is rejected with p essentially 0. There is sufficient evidence to conclude that there is agreement greater than chance in the artifact component (for all 20 problems). For test 2, the null hypothesis is rejected with $p = .011$. There is sufficient evidence to conclude that there is agreement greater than chance in the task component (for all 20 problems).

The results of tests 3 and 4 are summarized in Table 4.11.

Table 4.11. Statistics for 10 usability problems examined previously (OLD).

10 old problems	$\overline{P}$	$\overline{P}_C$	$\overline{P} - \overline{P}_C$	κ	Z	p
3. Artifact	.533	.274	.259	.357	7.336	.000
4. Task	.414	.34	.074	.112	2.144	.016

For test 3, the null hypothesis is rejected with *p* essentially 0. There is sufficient evidence to conclude that there is agreement greater than chance in the artifact component (for the "old" 10 problems). For test 4, the null hypothesis is rejected with $p = .016$. There is sufficient evidence to conclude that there is agreement greater than chance in the task component (for the "old" 10 problems).

The results of tests 5 and 6 are summarized in Table 4.12.

Table 4.12. Statistics for 10 usability problems not examined previously (NEW).

10 new problems	$\overline{P}$	$\overline{P}_c$	$\overline{P} - \overline{P}_c$	κ	Z	*p*
5. Artifact	.643	.366	.229	.437	6.442	.000
6. Task	.414	.372	.042	.068	1.018	.154

For test 5, the null hypothesis is rejected with *p* essentially 0. There is sufficient evidence to conclude that there is agreement greater than chance in the artifact component (for the "new" 10 problems). For test 6, while the null hypothesis would not be rejected at the .05 level of significance (with $p = .154$), it is important to note that κ is positive. There is some evidence to suggest that there may be agreement greater than chance in the task component (for the "new" 10 problems).

4.1.3 Remarks About The UPT Reliability Study

This section discusses the results presented in section 4.1.2 and examines several reasons for differences noted between UPT components. Four factors, outlined in section 4.1.2, that influence the level of agreement among classifiers using the UPT are examined more closely. These factors were the lack of classifier experience with the specific application domains, lack of familiarity with the UPT, the fact that the classifiers did not observe each problem as it occurred, and vague and incomplete problem descriptions. This section concludes with a brief discussion of the novelty of the UPT and how its newness affected classification, in particular, how classifiers interpreted the structure of the UPT as they classified usability problems.

The first factor, lack of classifier experience with the specific application domains, can certainly affect the classifier's ability to understand and classify usability problems noted on a given project. For example, consider an evaluator observing a user testing session with a complex software package designed for a highly trained, expert user group. Although the evaluator may understand artifact problems when they occur, the evaluator may not understand usability problems that occur because of the way the tasks are mapped to the system if he/she is not sufficiently familiar with the application domain. It is also likely that this lack of understanding could result in an inappropriate task classification. Since the demographic information collected on each classifier did not include expertise in the five application domains from which the sample problems were selected, no conclusions can be reached at this time. Investigating this possibility is a future project and is described in section 7.1.

The second factor, lack of familiarity with the UPT, can also affect classification. During a pilot study that preceded the reliability study described in section 4.1.2, it was noted that classifiers became more comfortable with the taxonomy as the study progressed. In the reliability study, the classifiers had minimal training on the UPT (training took the form of a brief, 20 minute, written tutorial. It is assumed that with additional training, classifiers will have an increased understanding of the categories and of problem classification using the UPT. It is believed that additional training will improve the reliability with which the UPT is used.

The third and fourth factors, that the classifiers did not observe each problem as it occurred and vague and incomplete problem descriptions, also affect the classifiers' ability to classify a problem appropriately. It is difficult to fully understand a usability problem when the amount of contextual information available to the classifier is inadequate. As discussed in section 1.2, many difficulties arise when developers attempt to use vague and incomplete problem descriptions to raise the level of usability achieved in the user interface. Some difficulties experienced by classifiers during the study that were attributed to poor problem descriptions are outlined below.

Several classifiers commented on the quality of the 20 problem descriptions (recall that although the real-world problem descriptions used in the study were sanitized to ensure confidentiality, they were not modified in any other way). Comments from several classifiers during, and after, the study indicated that they felt many of the problem descriptions were vague, incomplete, and provided inadequate contextual information while containing irrelevant information. Others noted that multiple problems were contained within a single problem description. Individual classifiers chose to ignore certain phrases within the descriptions 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.

One classifier indicated that the vagueness of the problem descriptions prevented a thorough examination of the task component. This classifier commented, "I did not dive into the task side too deeply. I think that this is because I did not have a full understanding of the user interface and possible tasks that the user may perform. I think that if I used this taxonomy in my own work, I would have a richer population within the task hierarchical structure."

Several other reasons potentially impact the differences between the results for the artifact and task components presented in section 4.1.2. These include the novelty of the UPT and the sensitivity of each UPT component to the application domain.

The UPT is a new approach to thinking about usability problems, their description, classification, and analysis. Traditionally, usability problems have been examined with respect to guidelines, system functionality, and potential solutions. And, although a theoretical foundation has been used to view systems within a task-artifact framework [Carroll 91] [Carroll 92], this framework has not been applied to usability problem classification or analysis. By capturing the two dimensions of a usability problem in the artifact and task components, the UPT enables developers to focus on problem characteristics instead of guidelines, functionality, and solutions.

Although the UPT captures both dimensions of a usability problem, its novel approach (especially to the task dimension) affected classification. Two classifiers remarked that they found classification in the artifact component easier than in the task component. This response could have resulted from problem descriptions with a stronger artifact component than task component (in the judgment of individual classifiers). However, it is also likely that the classifiers found the characteristics of the visualness, language, and manipulation categories in the artifact component intuitive and easy to understand, but had difficulty in distinguishing between the task-mapping and task-facilitation categories. Depending on their background and experience, it is possible that the classifiers interpreted the task categories somewhat differently and, in particular, imbued the task-facilitation category with additional meaning.

The two hypotheses mentioned above are supported by the values in the "Agrmt" column in Tables 4.7 and 4.8. Recall from these two tables that the classifiers had good agreement on 13 of 20 problems in the artifact classification, but in the task classification had good agreement on only 6 of 20 problems.

In addition, the differences noted in the results for each component could be due to the sensitivity of each component to the application domain. It is possible that the task component is more, or less, sensitive to the application domain than the artifact component. This hypothesis is based on the fact that the 10 old problems were selected from four projects and the 10 new problems were drawn from two projects. Recall the *p*-values of .016 for the 10 old problems (test 4) and .154 for the 10 new problems (test 6) in Tables 4.11 and 4.12, respectively. To determine if the task component is sensitive to application domain, more research is needed which may result in expansion and refinement of the task component so that the various aspects of problems due to tasks in differing application domains are included. This type of investigation is a possible future project and is outlined in section 7.1.

Regardless of the differences noted for the two components, usability problem classification using the UPT was shown to be reliable (see section 4.1.2). The reliability with which problems were classified in the artifact component did not vary regardless of which problem set was tested. More variability existed among the task classifications (recall that the level of agreement among the task classifications for the 10 old problems was significant, whereas the level of agreement among the task classifications for the 10 new problems was not). However, since the kappa statistic remained positive for the set of 10 new problems, it is concluded that even in this case, there was some agreement after correction for chance.

4.1.4 UPT User Satisfaction Survey

The seven classifiers were asked to rate the UPT in five areas by completing the usefulness and satisfaction questionnaire. The classifiers were asked to evaluate how strongly they agreed with each of the following five statements:

1. The UPT was easy to use.
2. The UPT was easy to understand.
3. The UPT was easy to learn.
4. The UPT was comprehensive (exhaustive).
5. The UPT is a useful tool for interactive software developers.

A four-point scale was used:

- 1, definitely do not agree,
- 2, somewhat agree,
- 3, agree for the most part, and
- 4, definitely agree.

The mean responses to the five questions are given in Table 4.13. As indicated in the table, for the most part, classifiers agreed that the UPT was easy to learn, was comprehensive (complete) and was considered to be a useful tool for interactive software developers. One classifier commented that while the UPT was complete for the 20 problem descriptions, that the UPT might not provide adequate coverage for other application domains.

Table 4.13. Mean response per question.

Question	Mean
1. The UPT was easy to use.	2.786
2. The UPT was easy to understand.	2.786
3. The UPT was easy to learn.	3.5
4. The UPT was comprehensive (exhaustive).	3
5. The UPT is a useful tool for interactive software developers.	3.786

The mean response for each classifier is given in Table 4.14. Note that, with the exception of classifier 6, the means vary from 3 to 3.8. These values indicate that six of the seven classifiers were satisfied with the UPT. This is further corroborated by the overall mean of 3.171. Individual comments, such as "I liked the breakdown into artifact and task," and "This was educational" supported these findings.

Table 4.14. Mean response per classifier.

Classifier	Mean
1	3.2
2	3.8
3	3.2
4	3
5	3
6	2.4
7	3.6

During the analysis of the responses to the usefulness and satisfaction questionnaire, it was concluded that a different approach was needed to assess the UPT. In particular, questions were needed that focused on the UPT as a conceptual model, as well as questions that concentrated on the instantiation of the UPT on the World Wide Web. Two classifiers did provide separate responses for the conceptual model and for the web instantiation (in these cases, the two values were averaged).

The classifiers may have interpreted each of the five questions differently depending on whether or not an individual question seemed to be more appropriate to the web version or the conceptual model. For example, it is possible that classifiers interpreted question 2 with respect to the conceptual model and question 3 with respect to the UPT instantiation. This hypothesis could explain the why the mean response for question 2 (the UPT is easy to understand) is lower than the mean response for question 3 (the UPT is easy to learn). Classifiers noted that the web document was well constructed; however, some difficulty was experienced as they tried to understand the new approach to problem classification embodied in the conceptual model.

4.2 Product Improvement And The UPT

This section outlines ways the UPT can be used to improve the level of usability achieved in the user interface of a software product. The techniques illustrated in this section were developed using the experiences of the author and several colleagues. Much of the discussion throughout this chapter is based on two critical incidents (presented in section 4.2.1), the problem reports associated with those incidents (presented in section 4.2.2), and solutions to those problems (also presented in section 4.2.2). Section 4.2.2 serves as a benchmark for the remainder of the chapter by examining problem reports and solutions without using the UPT. Section 4.2.3 re-examines the usability problems for each critical incident and discusses the benefits that could have been derived by using the

UPT during problem reporting. Section 4.2.4 outlines the advantages of using the UPT after problem reporting to analyze usability data and identify patterns of usability problems.

4.2.1 Two Critical Incidents

This section presents two critical incidents. The two incidents occur on different systems during specific tasks that are common tasks on each system. The first critical incident occurs as a user is formatting a removable diskette using an operating system with a graphical user interface. The second critical incident occurs when a user is attempting to add a row to a table in a spreadsheet-like application.

DISK NAME CRITICAL INCIDENT

A user is attempting to format a removable diskette using an operating system with a graphical user interface. After inserting the removable diskette in the floppy drive, the dialogue box shown in Figure 4.1 is displayed.

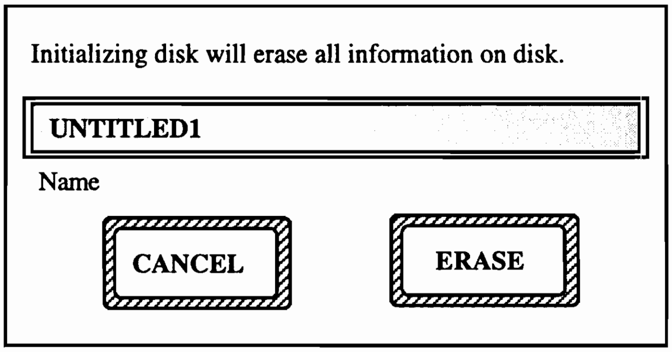
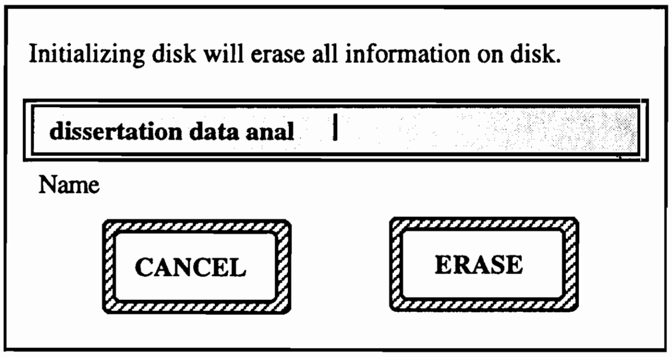


Figure 4.1. Dialogue box for disk initialization.

The user notes that the name field is fully highlighted and decides to enter a name for the disk. She selects the disk name "dissertation data analysis" and attempts to type that name in the name field. After the user types the "l" in "analysis," the computer beeps and does not accept any additional characters for the name. The cursor (insertion point) remains visible after the "l". The dialogue box seen by the user at this point is shown in Figure 4.2.


Initializing disk will erase all information on disk.

dissertation data anal

Name

Figure 4.2. Usability problem with the name field occurs during disk initialization.

The user attempts to enter the "ysis" in "analysis" three more times. The computer beeps after each try. After the fourth beep, the message box shown in Figure 4.3 pops up on the screen.

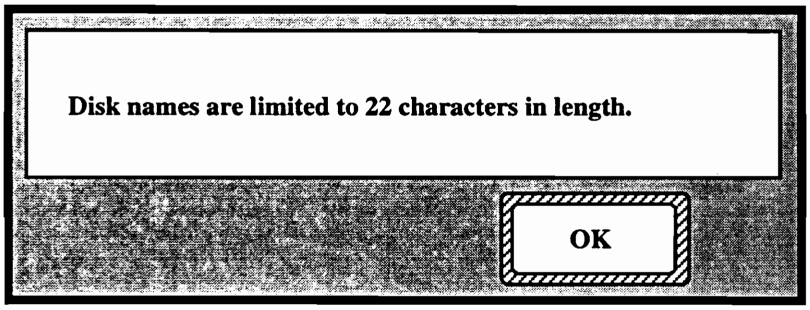

Disk names are limited to 22 characters in length.

Figure 4.3. Error message for disk initialization.

The user remarks that she does not like the beep and is frustrated to find that she must start again with a new, abbreviated name.

DATA ENTRY CRITICAL INCIDENT

A user is attempting to entering information for a new student in a table containing student records (one student per record, one record per row). Although the system has a graphical user interface, the data cannot be entered directly in the table. The user must select the "Add Row" menu item to display the data entry box illustrated in Figure 4.4. The information entered in the data entry box corresponds to one row in the table.

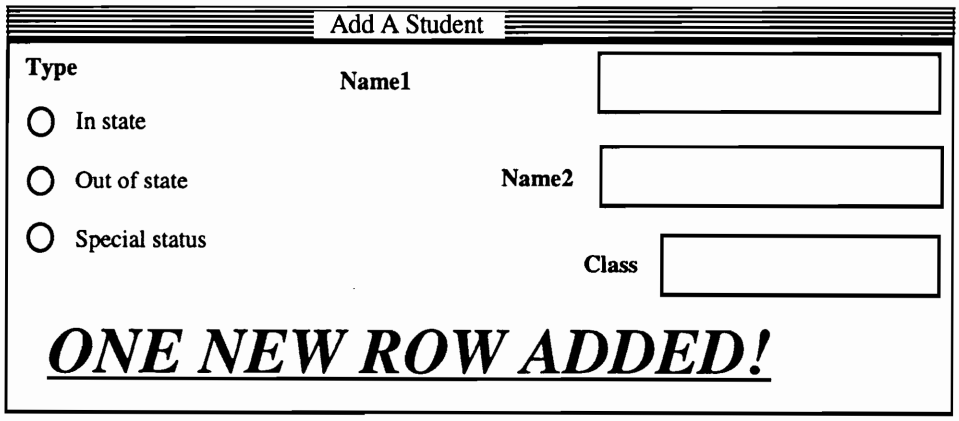


Figure 4.4. Data entry box.

The user thinks about entering the following data: student name, student address, class level (freshman, sophomore, junior, senior, graduate), and their residency status (in state, out of state, special student status). The user notices that the label on the title bar did not match the menu item that was selected ("Add Row"). He remarks that the size of the labels is small and difficult to see. The user comments that the names "Type", "Name1", and "Name2" are not meaningful to him. He is unable to distinguish the name field from the address field. He also notices that the "Name1" label is much farther away from it's associated field than the other labels. He wonders why the on-screen text says "One new record added!" when he hasn't done anything yet and comments that it dominates the dialogue box. He wants to enter the student's class but cannot recall the permissible values.

4.2.2 Problem Identification, Reporting, Analysis, And Correction Without The UPT

In current practice, developers identify and report problems that arise in critical incidents such as the two presented above. Sometimes usability problem data is analyzed prior to correction, other times it is not. Some current analysis techniques are informal and result in a brief, cursory examination of the data. Other techniques are more structured such as grouping by system function, grouping by heuristic violated, or ranking problems according to cost-to-fix and importance. This section takes a real-world approach to the two critical incidents in that the set of usability problems identified during the Disk Name critical incident is not analyzed, the set is usability problems associated with the Data Entry critical incident is analyzed. Note that to provide a benchmark for the remainder of this chapter, the UPT is *not* used in this section.

DISK NAME CORRECTION SCENARIO

Depending on their level of expertise, evaluators observing the Disk Name critical incident could report any or all of the following four problems.

1. The user did not like the beep.
2. The message box finally appeared after several beeps.
3. To correct the error, the user was forced to count the number of characters entered.
4. Since the name field was fully highlighted and longer than the allowable length, the user thought that disk names could be at least as long as the field.

An inexperienced evaluator may focus on problems 1 or 2, since they are the easiest to observe and report. A more experienced evaluator may note problems 1 and 2, but also report problems 3 or 4.

For this discussion, assume that all four problems have been reported. The evaluators give the four prose problem descriptions to developers who design solutions for each problem. The solutions are listed below.

1. Developers receiving problem description 1 (the user did not like the beep) decide to solve the problem by including a user preference (option) that would allow the user to set the volume of the beep.
2. Problem 2 (the message box finally appeared after several beeps) is addressed by removing the delay in the appearance of the message.
3. A solution to problem 3 (to correct the error, the user must count the number of characters entered) would involve adding new user interface functionality, e.g., having the system count the characters as the user enters text in the name field.
4. Developers decide to address problem 4 (the name field was longer than the allowable length, the user thought that disk names could be at least the field length) by providing a better visual cue for the length of a disk name. One slot (underscore) will appear for each allowable character position.

The four problem descriptions listed above do describe problems that should be addressed. Implementing the four solutions above will correct those problems and raise the level of usability achieved in this part of the user interface (the Disk Name dialogue box). However, the developers did not recognize the real, underlying usability problem: the system is not flexible enough to allow disk names of any length. Consequently, the descriptions do not capture the real problem and it remains in the user interface.

DATA ENTRY CORRECTION SCENARIO

Many usability problems are noted during the Data Entry critical incident. Eight possible problem reports are listed below.

1. The label on the title bar ("Add A Student") is not consistent with the name of the menu item that was selected ("Add Row") to display the data entry box.
2. The size of the text used in the labels is small and difficult to see.
3. The user comments that the names "Type", "Name1", and "Name2" are not meaningful.
4. The user is unable to distinguish the name field ("Name1") from the address field ("Name2").
5. The user had trouble understanding the words in the data entry box.
6. The field label "Name1" is much farther away from it's associated field than the other labels.
7. In addition to the distraction caused by the size of the line "One new record added!", the feedback provided by this on-screen text is premature as the record has not been added at this point in the task.

8. The user cannot recall the permissible values for the class field.

Developers on this team perform some analysis prior to suggesting solutions for each of the eight usability problems above. They look for commonalities and notice that problems 3 and 4 are different reports of the same basic problem: the field names were not meaningful enough. Problem description 5 is vague, the developers cannot decide whether or not it refers to the same problem as in descriptions 2 and 3. After further consideration, the developers also count the number of problems associated with this data entry box and observe that if an alternate form of data entry were designed and implemented (e.g., direct manipulation of the rows in the table), that the box could be eliminated. As a result, this global solution would address all eight problems. Rather than correcting each individual problem, they opt for implementing direct manipulation.

4.2.3 Using The UPT During Problem Reporting

As reported by the author and several colleagues in both academic and industry development environments, using the UPT during an evaluation activity can help developers write more accurate and complete problem descriptions. In current practice, problem descriptions are frequently vague and incomplete, often containing multiple problems and information about user reactions. This section examines how using the UPT during problem reporting can improve problem descriptions. Each problem description will be examined with respect to the following characteristics:

- clear (no ambiguity nor vagueness, no additional information is needed),
- precise (one description contains only one problem, multiple problems are not contained in the same description),

- comprehensive (a description contains artifact and/or task information), and
- problem-centered (information about the user is clearly distinguished from information about the problem).

In this section, the UPT is used to classify, and improve, the problem descriptions for the two critical incidents described above.

USING THE UPT DURING THE DISK NAME CRITICAL INCIDENT

Assume that an evaluator is observing the Disk Name critical incident. Also assume that the evaluator notes problem 1 first. He uses the UPT to focus on the artifact and task aspects of each usability problem and to classify those problems during problem reporting.

Problem description 1 (the user did not like the beep) would be classified in the artifact component as a visualness, non-message feedback problem since the beep is feedback that the user was unable to enter a character. Although the problem description focuses on the user's reaction to the beep, the non-message feedback category includes problems that are due to distracting, annoying, and confusing feedback. The developer realizes more detail was need in the problem description and recalls that the user was confused about the beep. He modifies the problem description accordingly. The developer attempts to classify problem description 1 in the task component, however, this problem does not have a task dimension. The problem is not about the structure of the task on the system or the ability of the user to follow that structure. Problem 1 is classified in the deepest level of the artifact component, and therefore, has a strong artifact dimension.

The first version of the problem description was precise, i.e., only one problem was contained in the description). However, it was not clear (additional information was needed). The new version of the description is both clear and comprehensive (both the artifact and task dimensions have been explored by the evaluator). In addition, it is now problem-centered as the user reaction is distinguished from the actual problem.

The evaluator continues with problem description 2 (the message box finally appeared after several beeps). He determines that this problem is a language, error message problem that is due to the delay in the appearance of the message. As with problem 1, the problem is classified in the deepest level in the artifact component (the problem has a strong artifact dimension); there is no task component. Since no new information was needed to classify this problem, the problem description was clear, precise, problem-centered, and comprehensive prior to classification.

He examines problem description 3 (to correct the error, the user was forced count the number of characters entered). Since the error message is clear and does help the user solve the problem, there is no artifact component. However, it is classified as a task-facilitation, user action reversal problem because the system places the burden for error correction on the user and does not provide a mechanism for easy error recovery. It is classified at the deepest level in the task component, and therefore, has a strong task dimension.

Prior to classification, the problem description was clear, precise, and problem-centered. However, as the description is classified in the UPT, both dimensions of the problem are explored. The task classification produces a comprehensive description as information is identified that supplements the original description.

Having thought about the task component of problem description 3, the evaluator recognizes that he has focused on what the system did *after* the problem occurred, not on the problem itself. He realizes that the usability problem occurred because the highlighted name field was longer than the allowable length (problem description 4). He classifies this as a manipulation, cognitive aspects, visual cues problem in the artifact component. In the task component, the problem is classified as task-facilitation, keeping the task on track problem since the system did not prevent user errors. This problem description is classified at the deepest level in both components.

No additional information was needed to classify problem description 3 in either component. Prior to classification, the description was clear, precise, and problem-centered. After classification, the description is comprehensive in that it has been examined from both an artifact perspective (visual cues) and from a task perspective (error prevention).

The classifications for problem descriptions 1 through 4 are summarized in Table 4.14. A value of "artifact" in the "strong" column indicates that an individual problem has a strong artifact dimension and was classified in the deepest level of the artifact component of the UPT (problem descriptions 1 and 2). A value of "task" in the "strong" column indicates that an individual problem has a strong task dimension and was classified in the deepest level of the task component of the UPT (problem description 3). When both values are listed, the problem was fully classified in each component (problem description 4).

Table 4.14. Problem classification for Disk Name critical incident.

Description	Artifact	Task	Strong
1	Non-message feedback	No task component	Artifact
2	Error messages	No task component	Artifact
3	No artifact component	User action reversal	Task
4	Visual cues	Keeping the task on track	Artifact & Task

This scenario demonstrates how the problem reporting process becomes more focused, and problem descriptions more clear, precise, problem-centered, and comprehensive when the UPT is used during problem reporting. Although problem descriptions 1 and 2 (user preferences regarding beeps and messages that do not appear immediately) are problems that should be corrected, they do not begin to capture the real, underlying problem in the critical incident: the user must know ahead of time that disk names can only be 22 characters in length and must count characters as they are typed into the field. Problem description 3 touches on the real problem; however, it is written from the perspective of error recovery. Problem description 4 does capture more information about the real problem. By using the UPT to classify each of the four problems, the developer increases his level of understanding of the underlying problem (this is illustrated as he progresses from description 1 to description 4).

USING THE UPT DURING THE DATA ENTRY CRITICAL INCIDENT

Assume that an evaluator is observing the Data Entry critical incident. She uses the UPT to focus on the artifact and task aspects of each usability problem and to classify those problems during problem reporting. For this example, the order in which the classifications below are presented is *not* relevant to the discussion.

The evaluator notes problem 1 (the label on the title bar is not consistent with the name of the menu item) and identifies it as a consistency problem. She then uses the UPT to classify the problem. She notes that the UPT does not contain a consistency category and examines the artifact component first. She recognizes that it is a language, naming/labeling problem and finds that consistency issues related to naming and labeling are included in this category. The problem is classified to the deepest level, and therefore, has a strong artifact component. She then examines the task component. The problem is not about task structure nor facilitation; there is no task component. Note that in this case, prior to classification, the problem description was clear, precise, problem-centered, and comprehensive as no additional information is needed to classify the problem.

Problem description 2 (the size of the text used in the labels is too small and is difficult to see) is classified as a visualness, object appearance problem. The UPT category is about problems with appearance and include the size of user interface objects (in this case, a text object). Thus, the last part of the description, "is difficult to see", is distinguished from the real problem since it describes the effect on the user. Problem description 2 has a strong artifact component, but no task component. Prior to classification, the problem description was clear, precise, and comprehensive. After classification, it is also problem-centered as the effect on the user is distinguished from the actual problem.

The evaluator attempts to classify problem description 3 (the names "Type", "Name1", and "Name2" are not meaningful) in the artifact component. She recognizes that the problem is about field labels that are not meaningful enough to the user. However, she notices that problems in the naming/labeling category that are about meaningful labels refer to one label *at a time*. The evaluator modifies description 3 by separating it into 3 separate descriptions: one for each label name (3a, 3b, and 3c). Each new description is classified as a language, naming/labeling problem. Each of these three problems have a strong artifact component, but no task component.

The original problem description was clear and problem-centered; however, multiple problems were contained in the one description. By using the UPT to examine both dimensions of the problem, the resulting problem descriptions are precise and comprehensive.

Problem description 4 (the user is unable to distinguish the name field, "Name1", from the address field, "Name2") is also about labels. In this case, the labeling problem is due to the similarity of names used for different fields. It is classified as a language, naming/labeling problem and has a strong artifact component. There is no task component. Prior to classification, problem description 4 was clear, precise, and problem-centered. UPT classification affirms that it is also comprehensive.

Using the UPT to classify the fifth problem description (the user had trouble understanding the words in the data entry box) emphasizes how vague and imprecise that description is. The evaluator knows that the problem is classified as a language problem in the artifact component. However, she is unable to determine whether the problem belongs in the naming/labeling category or the other wording category (which contains problems with on-screen text). To eliminate the vagueness from the problem description, she uses the UPT do identify the necessary distinguishing information. This information would specify whether the problem occurred because of a name(s) or label(s) or whether it occurred because of other words used in the interface. If it is due to other wording, she would specify further whether the problem occurred with a feedback message, an error message, other types of system message, on-screen text, or user-requested information/results.

Since description 5 is so vague and imprecise, she is not comfortable classifying the problem in the task component. After adding the new information, the description would be clear. If the new information reveals that multiple problems are contained within that description, those problems would be separated into different descriptions. Then each description would be precise and problem-centered. By using the UPT to explore both the artifact and task components a higher quality, comprehensive problem description is produced. The new description(s) would then classified in the task component.

Problem description 6 (the field label "Name1" is much farther away from it's associated field than the other labels) can be considered a consistency problem, i.e., the space between fields and labels is not the same throughout the dialogue box. Using this as a guide, the problem is classified as a visualness, object (screen) layout problem. The problem has a strong artifact component, and although layout problems are sometimes indicative of poor task structure, there is no task component for this particular problem because, in this case, placement of the field label is not related to task structure. Prior to classification, the description was clear, precise, and problem-centered. UPT classification affirms that it is also comprehensive.

The evaluator attempts to classify problem description 7 (in addition to the distraction caused by the size of the line "One new record added!", the feedback provided by this on-screen text is premature). She takes the first half of the description (the distraction caused by the size of the line "One new record added!") and classifies it in the artifact component as a visualness, object appearance problem. The first half of the problem has a strong artifact component, but no task component. The second half of this description (the feedback provided by this on-screen text is premature) is classified in the artifact component as a language, feedback message problem. This half also has a strong artifact component, but no task component. She realizes that the assumption of mutual exclusion among artifact categories has been violated. She re-examines the problem description and determines that the description contains two problems. She divides the two problems into separate descriptions (7a and 7b).

Description 7a is clear and precise. By classifying this description in the object appearance category, the effect on the user is distinguished from the actual problem and the description is problem-centered. Description 7b is also clear, precise, and problem-centered. Both descriptions contain sufficient information to be classified appropriately and are comprehensive.

To classify problem description 8 (the user cannot recall the permissible values for the class field), the evaluator examines the artifact component. The problem is not about visualness, language, nor manipulation. She concludes that there is no artifact dimension to this problem. The problem is about preventing user errors and minimizing the user's memory load by providing a list of allowable values from which the user can select. Therefore, she classifies in the task component as a task-facilitation, keeping the task on track problem. Prior to classification, the problem description is clear, precise, problem-centered, and comprehensive.

The classifications for problem descriptions 1 through 8 are summarized in Table 4.15. Note that each of the three problem descriptions that resulted from number 3 are included (3a, 3b, and 3c). Problem description 5 has been classified as written initially, prior to improving the description. As a result, the "strong" column contains the value "vague artifact". This indicates that the vagueness of the problem description prevented classification to the deepest level in the artifact component. The classification for problem description 7 is based on the improved descriptions that resulted when the two problems in the initial description were divided into two descriptions (7a and 7b).

Table 4.15. Problem classification for Data Entry critical incident.

Description	Artifact	Task	Strong
1	Naming/labeling	No task component	Artifact
2	Object appearance	No task component	Artifact
3a	Naming/labeling	No task component	Artifact
3b	Naming/labeling	No task component	Artifact
3c	Naming/labeling	No task component	Artifact
4	Naming/labeling	No task component	Artifact
5	Language	No task component	Vague Artifact
6	Object (screen) layout	No task component	Artifact
7a	Object appearance	No task component	Artifact
7b	Feedback message	No task component	Artifact
8	Keeping the task on track	No artifact component	Task

This scenario demonstrates how the UPT can be used during problem reporting to improve the quality of individual problem descriptions. Although several of the original problem descriptions were clear, precise, problem-centered, and comprehensive prior to classification (1, 4, 6, and 8), the UPT enabled the evaluator to improve problem descriptions 2, 3, 5, and 7.

4.2.4 Using The UPT After Problem Reporting

The UPT can also be used after problems have been reported to classify and improve problem descriptions. In addition, the UPT can provide the foundation for usability problem analysis with specific emphasis on identifying patterns of usability problems. Identifying patterns of usability problems help developers design global solutions that address multiple problems.

4.2.4.1 Using The UPT To Classify Problems After Reporting

As discussed in section 4.2.3, the UPT can help developers improve problem descriptions by classifying problems during problem reporting. In particular, the examples illustrated how the UPT can be used to focus problem identification (Disk Name critical incident), to improve the clarity of a given description (Data Entry critical incident, description 5), and, as illustrated by problem description 7 of the Data Entry critical incident, to capture individual problems in separate descriptions. Other examples showed how the UPT can help evaluators distinguish between information about the actual problem and information about the user.

Although it is more effective to use the UPT during problem reporting, the UPT can also be used after the reporting process has concluded to help developers improve problem descriptions. In the example discussed in section 4.2.4.1, the improvements to problem descriptions 3 and 7 could have been made after the evaluation as well as during problem reporting. Recall that for problem 3, the categories that are about names and labels that are not meaningful enough occur with one label at a time. The problem 7 example illustrated how the mutual exclusiveness of the categories within one component can be used to distinguish among problems. One additional example, discussed below, illustrates how the two components can likewise be used to distinguish among multiple problems contained in one description.

Assume that the problem reporting phase is complete. A developer has been handed the following problem description for the Disk Name critical incident: "the error message finally appeared after several beeps" with a message that indicated that "a maximum of 22 characters are allowed in a disk name." The developer classifies the problem description in the artifact component as a language, error message problem. In the task component, the classification is task-facilitation, user action reversal. When these two classifications are considered further, it is concluded that there are actually two problems in the one description, i.e., in this case, the two categories do not describe two dimensions *of the same problem*. That is, a problem that is due only to the delay in the appearance of the message box is a very different from a problem that is about the absence of a mechanism for easy error recovery. The developer would separate the problems into two different descriptions and classify each along the other component. The first description has a strong artifact component and would be classified in the task component. The second description has a strong task component and would be classified in the artifact component. The original problem description was clear problem-centered. The two new descriptions are precise and comprehensive.

4.2.4.2 The UPT As A Foundation For Analysis

This section describes how the UPT can provide the foundation for identifying patterns of usability problems. Patterns of usability problems can be used to guide the identification of global solutions that address multiple problems. In addition, relating the UPT to other aspects of the user interface, such as tasks and user interface objects, can result in extensions to UPT analysis. Usability problem groupings within one UPT category are examined first. Groupings across UPT categories are investigated next.

Groupings that require additional information are then presented. In each case, the pattern of usability problems is used to identify global solutions.

GROUPINGS WITHIN ONE CATEGORY

Assume that evaluators identify 350 usability problems. The problems are classified in the UPT and a distribution of problems over UPT categories is derived. Of the artifact classifications, 35% are visualness, 40% are language, 5% are manipulation, and 20% do not have an artifact component. Of the task classifications, 28% are task-mapping, 30% are task-facilitation, and 42% do not have a task component. The distribution is summarized in Table 4.16.

Table 4.16. Distribution of problems across UPT categories.

Category	Percent
Visualness	35
Language	40
Manipulation	5
Other (Artifact)	20
Task-mapping	28
Task-facilitation	30
Other (Task)	42

The evaluators examine the problems in each category more closely. They find that half of the visualness problems are about text that is too small. In addition, the developers note that most of the task-mapping problems occur when the users are performing arithmetic calculations using post-fix notation.

The developers identify two global solutions based on individual categories. The first is to enlarge the size of all text in the user interface (this addresses roughly 17.5% of the problems). Second, the developers decide to change the sequence of all calculation subtasks from post-fix to in-fix. This will address the associated task-mapping problems.

If the developers had not analyzed the problems according to categories, they might have addressed each problem (or just a subset of those problems) individually. If only a subset of the visualness problems were corrected, some "too small" text would have remained in the interface. In addition, depending on the programming language used implement the user interface, it might be possible to address all font size problems with one (or a few) corrections. If the problems have not been grouped, this type of approach could not be used. Had the task-mapping problems not been grouped, they might have been addressed individually. This could result in inconsistent corrections, i.e., some corrections could have focused on in-fix notation, others might have focused on superficial changes to help the user with post-fix notation.

GROUPINGS ACROSS CATEGORIES

Developers can also identify groupings of problems across categories. Consider the 40% in the language category in Table 4.16 above. Developers examine the problems in this category and find that they have the following distribution: 25% are feedback message problems, 20% are error message problems, 5% are system message problems, 30% are on-screen text problems, and 20% are user-requested information/results problems. This distribution is presented in Table 4.17.

Table 4.17. Distribution of problems across language, other wording categories.

Other wording	Percent
Feedback messages	25
Error messages	20
System messages	5
On-screen text	30
User-requested information/results	20

Developers examine problems in the feedback, error, and system message problem types. They determine that most of the problems in these categories have resulted from using system terms rather than user task domain terms. They identify a global solution that involves rewording the messages accordingly.

The example given above is based on a problem (using system terminology inappropriately) that is common to the three message categories contained in other wording. Because of the similarities among these three categories, the grouping was easy to detect. Other patterns may exist among very different categories that are not as easily detected. For example, possible groupings may be detected among an artifact and a task category (object appearance and alternatives). Further investigation is needed to identify these patterns. This is planned for a future project and is described in Chapter 7.

GROUPINGS THAT REQUIRE ADDITIONAL INFORMATION

To identify certain patterns of usability problems, additional information is needed. This information includes the user task being performed and the user interface object that was used when the problem occurred. The Disk Name critical incident illustrates how task information can used to group problems. The Data Entry critical incident illustrates how user interface object information is likewise used to identify patterns of usability problems.

Task information can be used with UPT classifications to identify global solutions. Consider the Disk Name critical incident. Assume that all four problems were reported, i.e., none were eliminated from the list. If the four problems were grouped according to the user task (disk initialization), two global solutions could be identified quickly that would address the entire set of problems (either correct problem 4 or allow disk names to be any length).

Although the two global solutions in the Disk Name critical incident outlined above could be identified without the UPT, the UPT can be used to supplement task grouping in several ways. First, lengthy or complex tasks may have a large number of associated usability problems. In these instances, the UPT can contribute significantly to organizing those problems. Second, UPT classification enables developers to more easily determine if the problems are artifact problems or task problems. If the problems associated with a given task are primarily artifact problems, the solution(s) may be very different than if those problems have a strong task component. Third, UPT classification and analysis can increase the level of understanding about tasks and the problems that occur during those tasks.

Usability problems can also be grouped according to user interface object. As in the task discussion above, the UPT can supplement grouping by object to identify global solutions. Consider the eight problems classified in the Data Entry critical incident. The problems occurred within the same dialogue box. If the problems had been grouped according to the user interface object (the dialogue box), developers could identify a global solution (use direct manipulation) that would address all eight problems.

When the UPT is used in conjunction with grouping by object, it is easier to determine if most of the problems associated with a specific user interface object are artifact problems or task problems. If they are artifact problems, developers must examine how that object was constructed. If they are task problems, developers must think about how that object was used. It is likely that very different solutions would be needed to correct the artifact (or task) problems associated with a particular object. In addition, UPT classification can help developers determine if the same type of problem has occurred many times with many different objects of the same type (e.g. the font size of many different field labels is too small or the status bar indicators used for several different, lengthy operations is too narrow). By investigating how often the same type of problem occurs with the same type of object, developers can identify more efficient solutions.

Additional research is needed to determine the nature of the relationships of UPT categories with user tasks and various user interface objects. This is discussed briefly in Chapter 7.

4.3 Summary

This chapter presented a reliability study which showed that the UPT can be used to classify usability problems reliably. Since the components of the UPT correspond to the two dimensions of a usability problem (artifact and task), each problem is classified in the artifact component as well as the task component. Recall that the UPT has four levels and that each category in levels 3 and 4 contains specific examples of usability problems in that category. See Figure 3.1.

Although the objective is to classify a problem as deeply as possible in each component of the hierarchical structure (levels 3 or 4), it is not always possible to classify a problem in levels 2, 3 or 4. When problems can be classified in categories in levels 3 or 4, they are matched with the specific examples of usability problems in those categories. When usability problems are classified in categories in level 2 or some of the categories in level 3, they are matched to definitions rather than specific examples. Recall problem description 5 of the Data Entry critical incident (the user had trouble understanding the words in the data entry box). This problem was matched to the definitions provided in the language category, but could not be classified further.

Some problems cannot be fully classified. Problem descriptions frequently are unclear, imprecise, and/or incomplete. Other problem descriptions may have either a strong artifact or a strong task component and are classified deeply in only one component.

In the experience of the author and several colleagues, thinking about the two dimensions of a usability problem helps developers isolate the major problem in a critical incident, write more complete problem descriptions, eliminate vagueness, and reduce the tendency to include multiple problems in one description. Evaluators are also able to distinguish between the problem and user reactions, the effect on the user, and contextual information. These conclusions were confirmed during a discussion with one usability professional who indicated that a working knowledge of the UPT categories helped her identify individual problems and report them clearly and precisely [Bowen 96].

While using the UPT during problem identification improves problem identification and description, it does not guarantee that the real, underlying problems will be identified. For example, consider the first problem description in the disk initialization critical incident (user didn't like the beep). It is possible that after classifying that description in the UPT, an inexperienced developer will still not recognize the underlying problem. Further research is needed to develop a method for improved problem identification.

After usability problems have been identified and reported, the UPT can be used to analyze those problems and identify patterns. Since patterns of usability problems are often hard to detect among a large group of usability problems, UPT contributions during this activity are substantial. Patterns of problems within a one category or across categories can be identified easily. If cost or severity data is collected for each problem, developers can investigate ways to prioritize the problems across categories as well as *within* individual categories. In addition, the UPT can be used in conjunction with other types of groupings (according to user task or user interface object) to characterize the problems associated with each group and identify multiple solutions to those problems.

5 THE UPT AND PROCESS IMPROVEMENT

Three important concepts in software process improvement are data collection, data analysis, and the association of that data to specific improvements. These three concepts also apply to user interface process improvement. Recall that Chapter 4 presented the UPT and discussed ways to use the UPT to classify, organize, and analyze usability problem data. This chapter shows that usability problem data is an important component of user interface process improvement, and illustrates how to relate types of usability problems to specific improvements. The two types of improvements that are explored are based on the two development context factors discussed in Chapter 1: developer factors (team member roles and skills) and activity factors (activities, methods, and techniques). In particular, this chapter relates types of usability problem data to improvements that can be made to the development team as well as improvements that can be made to the development process.

This chapter presents the findings of a study undertaken to identify associations of the five primary UPT categories with each of the two development context factors. The study consisted of two parts: the Roles Evaluation and the Activities Evaluation. The chapter is organized as follows. An overview of the study is discussed in section 5.1. Section 5.2 presents the results of the Roles Evaluation in which developer roles were associated with individual UPT categories. Section 5.3 presents the results of the Activities Evaluation in which development activities were associated with individual UPT categories. Section 5.4 compares and contrasts the results described in sections 5.2 and 5.3. Section 5.5 outlines a step-by-step procedure for process improvement in which the UPT is used as a diagnostic tool. The results presented in this chapter are summarized in section 5.6.

5.1 The UPT Association Study

The purpose of this study was to identify associations between developer roles and UPT individual categories, as well as between development activities, methods and techniques and individual UPT categories.

Six experts in user interface engineering participated in the study. Each expert had extensive experience (at least thirteen years) as a usability engineer, software engineer, and/or human-computer interaction researcher in industry, government, and academic environments. Their experience, in number of years, is summarized in Table 5.1.

Table 5.1. Classifier experience: number of years in industry, government, and academic environments.

Expert	Industry	Government	Academic
1	16	-	10
2	5	-	20
3	10	3	15
4	19	15	-
5	11	-	2.5
6	10	2	9

With the exception of expert 5, each participant had experience in both software and user interface development (see Table 5.2). Expert 5 had experience as a researcher in both industry and academia. In some cases, the number of years given in the columns in Table 5.1 overlapped. Five experts used guidelines and heuristics regularly; one used them infrequently. At the onset of the study, one expert had limited exposure to the UPT; five had no prior experience with the UPT.

Table 5.2. Classifier experience: number of years in software and user interface development.

Expert	Software Development	User interface Development
1	9	7
2	10	15
3	5	12
4	10	15
5	-	-
6	12	7

The materials sent to each expert contained the following items:

- instructions,
- UPT documentation,
- the hierarchical structure of the UPT as shown in Figure 3.1,
- Roles Form 1,
- Roles Form 2,
- Activities Form, and
- three glossaries (for Roles Form 1, Roles Form 2, and the Activities Form).

Roles Form 1 contains a list of both software engineering and usability engineering roles. Roles Form 2 contains only those software engineering present on Roles Form 1. The Activities Form contains software engineering and usability engineering development activities, methods, and techniques. Roles Form 1, Roles Form 2, and the associated glossaries are located in Appendix B, the Activities Form and associated glossary are in Appendices C.

The participants were asked to perform two assessments. Each participant was first asked to assess the degree to which individual roles and skills could address usability problems in each of the five primary UPT categories (the Roles Evaluation); and second, to assess the degree to which individual development activities, methods, and techniques (henceforth referred to as activities) could address usability problems in each of the five primary UPT categories (Activities Evaluation). The following scale was used:

0	No. This role, or activity, essentially does not address this category of usability problem.
1	Moderate/somewhat. This role, or activity, might address this category of usability problem.
2	Yes. This role, or activity, would address this category of usability problem.

For the purposes of this study, "address" meant prevent, think about, detect, predict, and/or correct. The forms are given in Appendices B and C.

The responses associated with each UPT category and combinations of those categories were analyzed using the mean and standard deviation. The mean is a measure of central tendency of a set of data. The standard deviation is a measure of the variability of that data (a small standard deviation indicates that the experts tended to agree). Recall that the data sets examined in this chapter contain discrete values comprised of only zeros, ones and twos. Due to the discrete data, the possible values contained in the sets of responses on the Roles Forms and the Activities Form, and the relatively small sample size, the standard deviations that were computed tended to specific, distinct values such as .204, .478, .574, .824, .917, and 1.417.

The means and standard deviations were interpreted in the following way. A large mean (greater than or equal to 1.5), combined with a small standard deviation (less than or equal to .6), is used to identify roles and activities that definitely address UPT categories. A small mean (less than or equal to .5), combined with a small standard deviation (less than or equal to .6), is used to identify roles and activities that definitely do not address UPT categories. Roles and activities with means between .5 and 1.5 are not notable in terms of this research since they are characterized by "Moderate/somewhat. This role, or activity, might address this category of usability problems." Likewise, roles and activities with large standard deviations indicate that the experts did not tend to agree on a rating.

The interpretation of each pair of means and standard deviations is indicated by one of the following five symbols: "+", "w+", a blank space, "w—", and "—". The meaning of each symbol is given below:

"+" indicates that agreement existed among the respondents that a particular role could be used to address usability problems in a given category (positive association, i.e., the mean is greater than or equal to 1.5 with a standard deviation less than or equal to .6),

"w+" indicates that weak agreement existed that a particular role could be used to address usability problems in a given category (weak positive association, i.e., the mean is less than, but very close to, 1.5 and a standard deviation not much larger than .6),

" " blanks indicate that there was either no agreement (i.e., any mean paired with a standard deviation greater than .6) or agreement that roles and activities only moderately addressed individual UPT categories (i.e., the mean is greater than .5 and less than 1.5, with a standard deviation less than .6),

"w—" indicates that weak agreement existed that a particular role could *not* be used to address usability problems in a given category (weak negative association, i.e., the mean is greater than, but very close to, .5 and a standard deviation not much larger than .6), and

"—" indicates that agreement existed that a particular role could *not* be used to address usability problems in a given category (negative association, i.e., the mean is less than or equal to .5 with a standard deviation less than or equal to .6).

While the remainder of this chapter focuses on positive associations, some interesting points will be noted about negative associations. Associations denoted by blanks will not be considered.

5.2 The Association Of Roles And Skills With UPT Categories

This section presents the two different kinds of results obtained during the Roles Evaluation. Section 5.1.1 examines the association of individual developer roles with UPT categories. Section 5.1.2 examines the results obtained by comparing expert responses on Roles Form 1 and Roles Form 2. This comparison is performed to determine whether or not software engineering skills are viewed differently depending on the availability of usability expertise (i.e., would a manager be more inclined to use a software engineering role to address usability issues if no usability experts available).

5.2.1 Association Of UPT Categories With Developer Roles

In this section, responses of all six participants on Roles Form 1 are examined to identify roles that definitely address usability problems in each UPT category as well as those that address multiple categories of usability problems. Roles that do not effectively address usability problems in the various categories are also identified.

A synopsis of the means and standard deviations computed from the responses on Roles Form 1 is given in Table 5.3 below. Each cell in the table contains the interpretation of the associated pair of means and standard deviations computed from the responses (zero, one, or two). The columns labeled "V", "L", "M", "TM", and "TF" contain the interpretation of the results for the responses in each of the five UPT categories (the means and standard deviations for each pair of roles and categories are given in Tables D.1 through D.5 in Appendix D). The columns labeled "AC", "TC", and "UPT" present the interpretation of results for the combined categories in the artifact component, the combined categories in the task component, and the entire UPT structure, respectively (the means and standard deviations are given in Tables D.6 through D.8 in Appendix D).

The original responses (raw data) were used to calculate the means and standard deviations for each column. For example, the means and standard deviations for the responses for the "V" column were based on the original responses for that category. Likewise, the means and standard deviations for the "AC" column were calculated using the *original responses* for the three artifact categories, i.e., the "AC" column was *not* based on the means and standard deviations obtained for each individual category. Similarly, the means and standard deviations for the "TC" column were calculated using the *original responses* for the two task categories. The means and standard deviations for the "UPT" column were based on the original responses for all five categories.

The information in the table is read as follows. Consider the roles of end user and quality assurance. The role of end user addresses task-mapping usability problems (+), but are only weakly associated with the task component (w+). Similarly, the role of quality assurance does not address task-mapping problems (—) and is only weakly associated with the task component (w—).

An examination of the rows in Table 5.3 reveals that three roles are strongly associated with each of the individual categories, the two components, and the entire UPT: human factors expert, user interface interaction designer, and user interface evaluator. Similarly, two roles were identified for which a strong negative association exists for each of the individual categories, the two components, and the entire UPT: software documentor and software librarian.

Further examination reveals some surprising results. Consider the role of system analyst. A strong positive association exists between this role and the task-mapping and task-facilitation categories. This leads to a strong positive association in the task component. Notice, however, that although a strong positive association is not noted in the artifact categories, a strong positive association *is noted* for the entire UPT. This result is attributed to the very high means (and small standard deviations) in the two task categories, and the moderate mean and small standard deviation calculated for the artifact component (the mean of 1.222 and standard deviation of .444 can be found in Appendix D, Table D.6). Similarly, consider the role of marketer. A moderate association is observed for the visualness category, a weak negative association is observed for the language category, a strong negative association is observed for the manipulation category. However, the strong negative association in the artifact component is due to the very low mean of .167 (and standard deviation of .408) in the manipulation category and the moderately low mean of .833 (and standard deviation of .753) in the visualness category.

Table 5.3. Interpretation of roles and skills results for Roles Form 1.

Role	V	L	M	TM	TF	AC	TC	UPT
cognitive psychologist		+	+			w+		
customer								
end user				+			w+	
graphic designer	+							
human factors specialist	+	+	+	+	+	+	+	+
market analyst		—	—		w—	w—		
marketer		w—	—	w—	—	—	—	—
moderator (focus group)								
problem domain expert				+	+		+	
quality assurance				—			w—	
software specifier								
software designer	—	—		w—		w—	w—	w—
software documentor	—	—	—	—	—	—	—	—
software implementor	—	—		—		w—	w—	w—
software librarian	—	—	—	—	—	—	—	—
software tester				—	—		—	
systems analyst				+	+		+	+
technical writer	+	+				+		w+
user-interface interaction designer	+	+	+	+	+	+	+	+
user-interface software designer								
user-interface software implementor								
user-interface evaluator	+	+	+	+	+	+	+	+
user-interface documentor								

Strong positive associations for the three artifact categories (visualness, language, and manipulation) are captured in the Venn diagram in Figure 5.1 below. A strong positive association exists between three roles (human factors specialist, user interface interaction designer, and user interface evaluator) and each of the three artifact categories (these roles are found in the intersection of the three ovals). Only one role is associated with just an individual category: graphic designer with visualness. This association was expected, since the skills of a graphic designer can be applied directly to visualness problems such as object layout and appearance. Two roles are each associated with two categories: technical writer with visualness and language, and cognitive psychologist with language and manipulation. It was not unexpected that the role of technical writer was associated with the language category (language issues are very important and the skills of a technical writer can be directly applied to those issues). However, the association of technical writer with visualness was surprising. The experts may have assumed that a technical writer would have some expertise in textual layout and presentation. A clear link also exists between the cognitive psychologist and language and manipulation problems. The skills of cognitive psychologist include a knowledge of human memory limitations and the ability of humans to recognize visual cues in the interface.

Note that in all Venn diagrams in this work, a region with no element indicates an empty region. In Figure 5.1, three regions are empty: language "only", manipulation "only", and the intersection of visualness and manipulation.

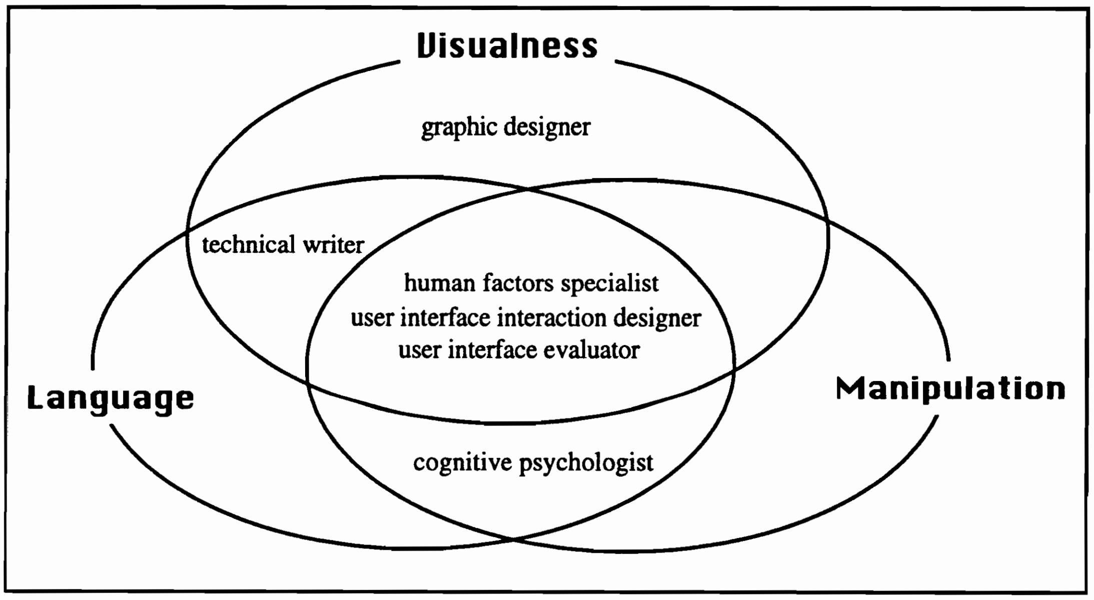


Figure 5.1. The overlap among roles for visualness, language, and manipulation.

Strong positive associations observed for the two task categories (task-mapping and task-facilitation) are illustrated in Figure 5.2. A strong positive association exists between five roles (human factors specialist, user interface interaction designer, user interface evaluator, systems analyst, and problem domain expert) and both of the two task categories (these roles are found in the intersection of the two ovals). The first three roles are the same as the three that addressed all artifact categories in Figure 5.1 above. The remaining two (systems analyst and problem domain expert) complement the original set of three roles. The systems analyst has the skills to perform various systems analysis activities which focus on necessary functionality, and, less directly, on user tasks. Similarly, the problem domain expert has knowledge of the application domain (which is applied to user tasks). The only role that is specifically applicable to one category is that of end user. The end user can be very helpful in determining if tasks have been structured appropriately on the system. No roles address only task-facilitation problems.

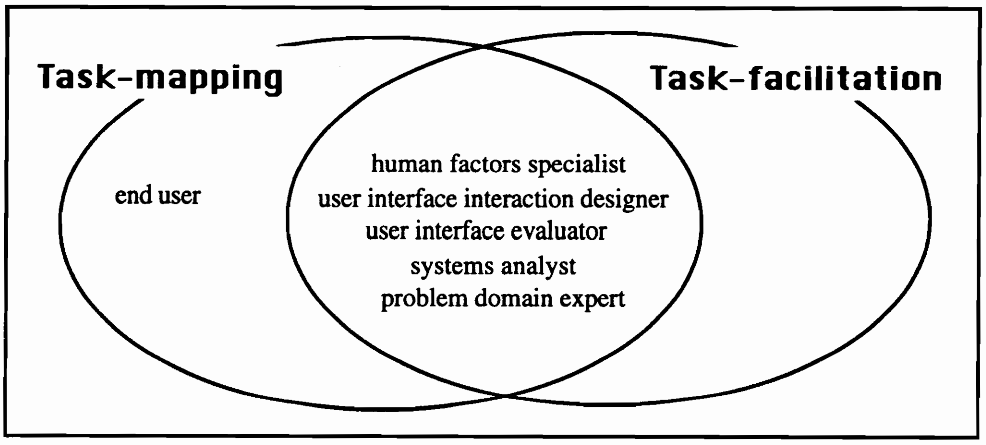


Figure 5.2. The overlap among roles for task-mapping and task-facilitation.

It is interesting to note that the role of cognitive psychologist was not associated with either task-mapping or task-facilitation. The skills that a cognitive psychologist bring to a development team can certainly address cognitive aspects of tasks, yet the blanks in the TM and TF columns in Table 5.3 indicate a lack of agreement that these skills would contribute directly to improving the task structure.

The strong positive associations observed for the artifact component as a whole and task component as a whole are illustrated in Figure 5.3. A strong positive association exists between three roles (human factors specialist, user interface interaction designer, and user interface evaluator) and each of the two components (note the intersection of the two ovals). At this level, the role of technical writer was associated only with the artifact component and the roles of systems analyst and problem domain expert were associated only with the task component.

Some interesting observations can be made as Table 5.3, Figure 5.1, and Figure 5.2 are compared. Notice that although there was a strong positive association of the role of cognitive psychologist with the language and manipulation categories (as shown in Figure 5.1 above), it was only weakly associated with the artifact component (note the "w+" in the UPT column of Table 5.3). Since the association was weak, it is not included in Figure 5.3 below. The role of graphic designer is associated with visualness but not associated with the artifact component because the skills of the graphic designer do not extend to language issues and may be only indirectly related to the appearance of visual cues (manipulation). Similarly, although the role of end user was strongly associated with the task-mapping category, it is only weakly associated with the task component and likewise is not included in the Venn diagram below.

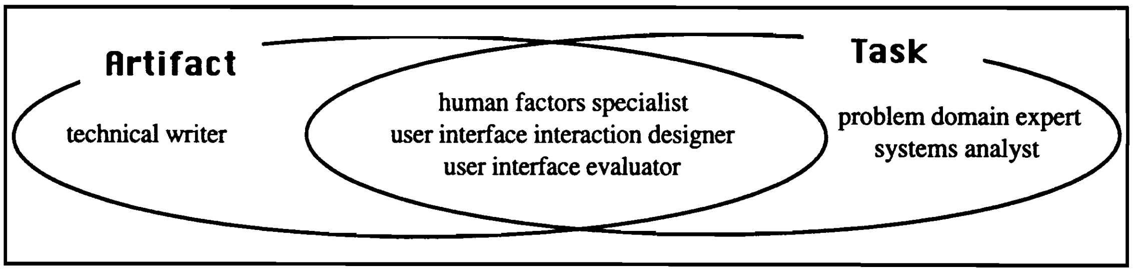


Figure 5.3. The overlap among roles for the artifact and task components.

Roles associated strongly with the overall UPT are given in the "Overall UPT" column in Table 5.4. Table 5.4 compares the "Overall" roles with roles that are associated with either the artifact component or task component *only*. The group of roles in the "Only AC" column is the set difference between the roles in the artifact oval in Figure 5.3 and the roles in the "Overall UPT" column in Table 5.4. Similarly, the group of roles in the "Only TC" column is the set difference between the roles in the task oval in Figure 5.3 and the roles in the "Overall UPT" column in Table 5.4.

For example, the role of technical writer is in the artifact oval in Figure 5.3, but it is not in the "Overall UPT" column in Table 5.4. Therefore, it is placed in the "Only AC" column in Table 5.4. Note that the role of human factors specialist is in the artifact oval in Figure 5.3. However, since it is included in the "Overall UPT" column in Table 5.4, it is *not* in the "Only AC" column in Table 5.4. Similarly, the role of problem domain expert is in the task oval in Figure 5.3, but it is not in the "Overall UPT" column in Table 5.4. Therefore, it is placed in the "Only TC" column in Table 5.4. Note that the role of systems analyst is in the task oval in Figure 5.3. However, since it is included in the "Overall UPT" column in Table 5.4, it is *not* in the "Only TC" column in Table 5.4.

The four roles that are considered the most "broadly" useful by the study participants are given in the "Overall UPT" column in Table 5.4. Note that the primary difference between Figure 5.3 and Table 5.4 is the movement of systems analyst from the task component to the overall UPT column. Also note that the role of technical writer is only weakly associated with the overall UPT, and therefore remains in the "Only AC" column below. The role of cognitive psychologist, weakly associated with the artifact component, is not associated with the overall UPT and does not appear in Table 5.4.

Table 5.4. A comparison of roles for the artifact component, task component, and overall UPT.

Only AC	Overall UPT	Only TC
technical writer	systems analyst human factors specialist user interface interaction designer user interface evaluator	problem domain expert

It is not surprising that three roles (human factors specialist, user interface interaction designer, and user interface evaluator) were perceived to be useful in addressing usability problems in each category since team members performing these roles have been trained in human-computer interaction. While it is often more productive for the role of user interface interaction designer to be performed by a different individual than the role of user interface evaluator (a fresh view of the system can result in the identification of additional usability problems), it is not absolutely necessary for these roles to be performed by different people. Regardless of the number of people that assume the four roles listed above, the roles would contribute significantly to a well rounded, multi disciplinary development team. This is an important conclusion for development organizations not yet sure of the value of usability engineering since this result indicates that the single, most valuable, "all-around" addition to a development team would be a person trained in these related areas.

The roles that were negatively associated with UPT categories were also identified. A Venn diagram approach was used that was similar to that used for positively associated roles. However, this approach did not provide any additional insight in distinguishing among roles that were either weakly or strongly negatively associated with UPT categories.

The roles that were negatively associated with UPT categories are traditional software engineering roles. Three roles were identified that were perceived to not be useful in addressing any UPT category (marketer, software documentor, and software librarian). The roles of quality assurance, software implementor, and software tester were negatively associated with either an individual category or with the entire task component. Market analyst and software designer were negatively associated with either with an individual category or the with the entire artifact component.

Several roles received an interpretation of blank in Table 5.3. These are the roles of customer, moderator (focus group), user interface software designer, user interface software implementor, and user interface documentor. Recall that a blank interpretation means that these roles were not associated (negatively or positively) with any category. Contrast two of these roles (user interface software designer, user interface software implementor) with software designer and software implementor. The two software roles were negatively associated with most of the individual categories (with a weak negative association with the overall UPT. This difference may be due to the perception that user interface software designers and user interface software implementors have some knowledge of the user interface and some background in usability whereas the individual performing the two software roles do not. This result justifies the recommendation that user interaction development be separated from software development [Hix 93].

5.2.2 Comparing Responses On Roles Form 1 And Roles Form 2

A second roles form (Roles Form 2) was designed to determine if the experts would view the applicability of software engineering roles and skills to UPT categories differently when usability specialists were not available. Roles Form 2 contains only the software engineering roles and skills present on Roles Form 1 (recall that Roles Form 1 contains both software engineering and usability engineering roles and skills). This section compares the responses on the two forms in an effort to determine if managers would use specific software engineering roles to address usability problems in a given UPT category if no usability specialists were available. The compared responses are for roles that are present in both forms. Both forms can be found in Appendix B.

Five of six participants completed the two roles forms: Roles Form 1 and Roles Form 2 (the sixth participant completed only Roles Form 1). To determine if there was a significant difference between the responses on Roles Form 1 (sample 1) and the responses on Roles Form 2 (sample 2), a paired t hypothesis test was performed. The paired t test was used since the two samples are not independent (the same people completed each form). A difference score, D, was calculated for each pair of responses for all five participants ($D = \text{response}_{\text{form2}} - \text{response}_{\text{form1}}$).

The null hypothesis, that the mean of the difference scores is not significantly different from 0, was tested against the alternative hypothesis that the mean of the difference scores is significantly different from 0. To compute the test statistic the sample mean and standard deviation are needed. The sample mean of the difference scores is $\overline{D} = .2092$. The sample standard deviation is $S_D = .58207$. Computing the test statistic, t, yields 6.4803 which results in a probability of $p = .000$. Since a two-tailed test was used, a p value less than .025 is needed to reject the null hypothesis. Since $.000 < .025$, the null hypothesis is rejected, i.e., it is concluded that at the $\alpha = .05$ level of significance, the difference scores are significantly different than 0. Since $\overline{D}$ is greater than 0, the responses on Roles Form 2 are higher than those on Roles Form 1.

Five additional paired t hypothesis tests were performed to determine if an individual participant's responses were significantly different from 0. In each case, the null hypothesis, that the mean of the difference scores for an individual participant is not significantly different from 0, was tested against the alternative hypothesis that the mean of the difference scores for that participant is significantly different from 0. The computed values for each of the five tests are given in Table 5.5. The values for the overall paired t test described above are also included.

Table 5.5. Paired t test values for each participant.

Participant	$\overline{D}$	S_D	**t**	*p*	**Roles Form 2 higher**
1	—	—	—	—	—
2	.48	.615	6.25	.000	yes
3	.14	.464	2.41	.008	yes
4	0	.586	0	.5	no
5	.12	.625	1.59	.056	no
6	.31	.498	4.99	.000	yes
Overall	**.2092**	**.58207**	**6.4803**	**.000**	**yes**

Participants 2, 3, and 6 did have significantly different responses on Roles Form 2, i.e., for participants 2, 3, and 6, the responses on Roles Form 2 are higher than those on Roles Form 1. Participants 4 and 5 did not have significantly different responses on Roles Form 2, i.e., for participants 4 and 5, the responses on Roles Form 2 are not higher than those on Roles Form 1.

Recall that participant 1 completed Roles Form 1 but did not complete Roles Form 2. The following comment was made: "I would not change any of my responses from Form 1. What is the purpose of two nearly identical forms?" Although this comment indicates that the responses on Roles Form 2 would not have been different than those on

Roles Form 1, they cannot be included since the form was not filled in. The observations about each individual participant as well as the overall result will be discussed further at the end of this section.

Recall that the overall paired t test for all pairs of observations resulted in the conclusion that the mean of the difference scores for all participants was significantly different from 0. To further investigate the relationship between the responses on Roles Form 1 and Roles Form 2, the mean responses for each role and each of the five primary UPT categories were calculated. The mean responses for how well each role in Roles Form 1 (that is also present in Roles Form 2) addresses usability problems in each of the five primary UPT categories are given in Table 5.6. The mean responses for how well each role in Roles Form 2 addresses usability problems in each of the five primary categories are presented in Table 5.7. Since only five participants completed the two forms, the means in both tables are based on those five opinions.

Table 5.6. Mean responses for roles in Roles Form 1 also present in Roles Form 2 (5 participants).

ROLES FORM 1	V	L	M	TM	TF
customer	0.6	0.6	0.6	0.8	0.6
market analyst	0.8	0.4	0.2	1.2	0.6
marketer	0.8	0.6	0	0.6	0
problem domain expert	1.2	1.6	0.8	1.8	1.6
quality assurance	0.8	0.6	0.4	0.4	0.8
software specifier	0.8	0.8	0.8	1.4	1
software designer	0.4	0.4	0.6	0.8	0.6
software documentor	0.2	0.6	0.2	0.4	0.2
software implementor	0.4	0.4	0.6	0.2	0.4
software librarian	0	0	0.2	0.2	0.2
software tester	0.6	0.6	0.6	0.6	0.6
systems analyst	1.2	1.4	1.2	2	1.6
technical writer	1.6	1.8	1.2	1.2	1.2

Table 5.7. Mean responses for Roles Form 2 (5 participants).

ROLES FORM 2	V	L	M	TM	TF
customer	0.8	1	0.8	1.2	0.8
market analyst	1.2	1	0.8	1.2	0.8
marketer	1	0.8	0.6	0.8	0.4
problem domain expert	1.8	2	1.4	2	1.8
quality assurance	0.8	0.8	0.6	0.4	0.4
software specifier	1	1	1	1.4	1
software designer	0.8	0.8	0.6	1	0.8
software documentor	0.6	0.6	0.4	0.6	0.4
software implementor	1	0.8	1	0.6	0.6
software librarian	0.2	0	0	0.4	0
software tester	0.6	0.6	0.8	0.8	0.4
systems analyst	1.8	1.8	1.4	2	1.6
technical writer	1.4	1.8	1.4	1.6	1.4

Casual inspection of the entries in the two tables shows that the mean responses in Table 5.7 for Roles Form 2 are somewhat larger than the mean responses in Table 5.6 for Roles Form 1. This is not surprising given the results of the overall paired t test described above. To determine if a linear relationship exists between the entries in each table, Pearson's coefficient of correlation, r, was computed. The calculation yielded r = .9. As r varies between -1 and 1, a value of .9 indicates that there is a strong positive linear relationship between the entries in Table 5.6 and the entries in Table 5.7. In view of this large positive correlation, it can be concluded that knowledge of a participant's response to a category on Roles Form 1 would give a strong indication as to their response to the corresponding category on Roles Form 2. When this result is examined in the context of the overall paired t test, it can be concluded also that the responses on Roles Form 1 differed from the responses on Roles Form 2 by a consistent amount.

The results of the overall paired t test and the strong positive linear relationship indicated by Pearson's coefficient of correlation seem to indicate that experts believe that specific software engineering roles would be used to address usability problems in a given

UPT category if there were no usability specialists available. However, other factors may also have influenced these results. Some possible factors are given below.

1. In retrospect, the amount of demographic information collected on each participant was too limited and not sufficiently specific. Although the participants' background and experience qualified them as experts in the field, it is possible that their backgrounds influenced their responses. For example, it is possible that participants with a stronger software engineering background responded differently than those with a stronger usability engineering background. Given the limitations of the demographic data, it was not possible to analyze the responses with respect to background or experience.

2. The instructions in the packet given to the six participants indicated that Roles Form 1 was to be completed just prior to Roles Form 2. Since the participants were not monitored as they completed the forms, there is no guarantee that the forms were completed in the same order. The order of completion could affect the responses.

3. Even if Roles Form 1 was completed just prior to Roles Form 2, it was the same type of form with the same instructions. Higher responses on the second form may be due to the participants tiring and possibly becoming more generous as they repeat the same process on the same questions.

4. Although the overall t test indicated that the mean of the difference scores was significantly different from 0, two of the five individual tests indicated that there was no substantial difference in the mean of the difference scores for those participants (4 and 5). When this observation is combined with participant 1's

comments regarding the blank Roles Form 2, it appears that individual expert opinions did vary.

5. The size of the sample can affect the significance of the results from a hypothesis test. For the paired t test, the test statistic is $t = (\overline{D} - \mu_D) / (S_D / \sqrt{n})$. This equation can be rewritten as $t = [(\sqrt{n})(\overline{D} - \mu_D)] / (S_D)$. Since the square root of the sample size, n, is a factor in the numerator of the test statistic, the larger the sample size, the larger the value for t. When t becomes large, the results of a hypothesis test can often be considered statistically significant; however, there may be no practical significance that can be attached to those results [Schulman 92].

The results of the overall hypothesis test indicate that the responses on Roles Form 2 are higher than those on Roles Form 1, i.e., that managers view software engineering roles differently if usability engineers were unavailable. However, for the reasons discussed above, this conclusion cannot be put forward with confidence.

5.3 The Association Of Activities, Methods, And Techniques With UPT Categories

This section presents the results from the Activities Evaluation. Responses of all six participants on the Activities Form are examined to identify activities that definitely address usability problems in each UPT category as well as those that address multiple categories of usability problems. Activities that do not effectively address usability problems in the various categories are also identified. The means and standard deviations computed from the responses on the Activities Form are given in Tables D.9 through D.16 in Appendix D.

A total of 38 development activities were included on the Activities Form which is given in Appendix C. The 38 activities were grouped by type as shown in Table 5.8. Note that the number of activities in each group varied.

Table 5.8. Types of activities on the Activities Form.

Type of Activity	Abbreviation	Number
system analysis	SA	11
guidelines	G	3
design	D	4
design representation techniques	DRT	5
inspection methods	I	7
user testing activities	UT	8

A synopsis of the means and standard deviations computed from the responses on the Activities Form is given in Table 5.9 below (see Tables D. 9 through D.16 in Appendix D for the means and standard deviations). Each cell contains the interpretation of the associated pair of means and standard deviations. Recall that the categories in the artifact component are labeled "V", "L", and "M", those in the task component are labeled "TM" and "TF", and the combined artifact categories, the combined task categories, and the entire UPT structure are labeled "AC", "TC", and "UPT", respectively. Similar to the discussion in section 5.2.1, the original responses were used to calculate the means and standard deviations for each column in Table 5.9.

An examination of the rows in Table 5.9 reveals some surprising results. Consider the three guidelines activities: use commercial style guides, use customized style guides, and use general interface guidelines. Note that a strong positive association exists for each guidelines activity and the visualness and manipulation categories as well as for the artifact component. However, a strong mean of 2 and standard deviation of 0 for visualness, a mean of 1.667 and a standard deviation of .816 for language, and a mean of 2 and standard deviation of 0 for manipulation result in the strong association of use customized style guides with the entire UPT. Also note that although a strong positive association exists for use general interface guidelines and each of the artifact categories, that this activity is not associated with the UPT (note the small mean of 1.167 and standard deviation of .983, and the small mean of 1 and standard deviation of .894 for the task-mapping and task-facilitation categories, respectively).

Strong positive associations between activities and the three artifact categories (visualness, language, and manipulation) are captured in the Venn diagram in Figure 5.4 below. A strong positive association exists between nine activities and each of the three artifact categories (these activities are found in the intersection of the three ovals). These nine activities include one guideline activity, two design activities, three inspection activities, and three user testing activities. No systems analysis activities (SA) were noted that address problems across the artifact categories, possibly because many system analysis activities are focused on tasks. In addition, no design representation techniques (DRT) are associated with the three artifact categories.

Table 5.9. Interpretation of results from the Activities Form.

Activity	V	L	M	TM	TF	AC	TC	UPT
System Analysis (SA)								
Contextual Inquiry				+	+		+	
Focus Groups			w—	+				
Functional Analysis			—					
Learn Application Domain		+		+			+	
Learn Competing Systems								
Systems Analysis	—		w—					
System Requirements & Specifications			w—	+	w+		+	w+
Task/Function Allocation	—			+	+	—	+	
Usability Requirements & Specifications	+		+			+		
User Analysis								
User Site Visits		+	w—	+			w+	
Guidelines (G)								
Use Commercial Style Guides	+		+			+		
Use Customized Style Guides	+		+			+		+
Use General Interface Guidelines	+	+	+			+		
Design Activities (D)								
High-fidelity Prototyping	+	+	+	+	+	+	+	+
Low-fidelity Prototyping	+	+	+	+	+	+	+	+
Participatory Design		+		+	+	w+	+	+
Usage Scenarios				+			+	
Design Representation Techniques(DRT)								
Behavioral Design Representation	—							
Knowledge & Model Based								
Object Orientation		w—						
State Transition Diagrams	—	—				w—		
Usage Scenarios, Use Cases			—	+	+	w—	+	
Inspection Methods (I)								
Cognitive Walkthrough								
Consistency Inspections	+	+	+	—		+	w—	
Feature Inspection	—	—	—	+		—		
GOMS Analysis	—	—				w—		
Guideline Reviews, Heuristic Evaluation	+	+	+			+		+
Pluralistic Walkthroughs	+	+	+	+	+	+	+	+
Standards Inspections				—	—		—	
User Testing Activities (UT)								
Alpha Tests								
Beta Tests								
Co-discovery	+	+	+	+	+	+	+	+
Critical Incident Taking	+	+	+	+	+	+	+	+
Structured Interviews		+		+				
Testing User Performance					+		w+	
User Preference								
Verbal Protocol Taking	+	+	+	+	+	+	+	+

Three activities address both visualness and manipulation problems: usability requirements and specifications (SA), use commercial style guides (G), use customized style guides (G). Four activities are associated with only the language category. Of those four, three are system analysis activities (learn application domain, user site visits, and participatory design), one is a user testing activity (structured interviews). It is not unexpected that these four activities were associated with the language category. Each activity focuses on either the application domain, the user's environment, or the user (all of which are related to the appropriate use of language in the user interface). In addition, note that three different systems analysis activities are associated with the visualness, language, and manipulation categories; however, there are no system analysis activities in the intersection.

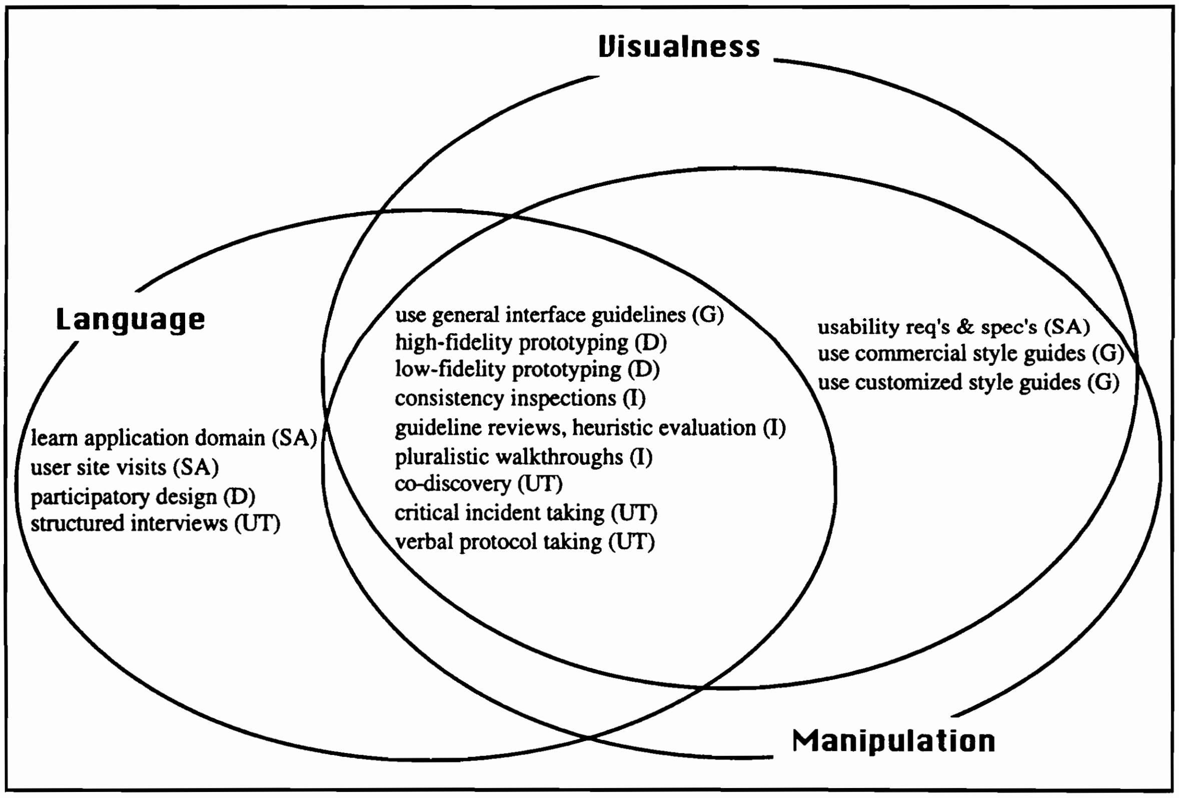

Visualness
Language
Manipulation
learn application domain (SA)
user site visits (SA)
participatory design (D)
structured interviews (UT)
use general interface guidelines (G)
high-fidelity prototyping (D)
low-fidelity prototyping (D)
consistency inspections (I)
guideline reviews, heuristic evaluation (I)
pluralistic walkthroughs (I)
co-discovery (UT)
critical incident taking (UT)
verbal protocol taking (UT)
usability req's & spec's (SA)
use commercial style guides (G)
use customized style guides (G)

Figure 5.4. The overlap among activities for visualness, language, and manipulation.

Strong associations were noted between 10 activities and the two categories in the task component. These activities are illustrated in the intersection of the two ovals in Figure 5.5 below. The 10 activities included two system analysis, three design, one design representation technique, one inspection method, and three user testing activities. It is not unexpected that task/function allocation is associated with both task-mapping and task-facilitation. Developers performing this activity examine the user's role as well as the system's role in a given task. Note that no guidelines activities were associated with the two categories in the task component. This is due to the very few guidelines that address user tasks.

Seven activities were strongly associated only with task-mapping, possibly the most difficult aspect of user interface design. Three were system analysis activities (focus groups, learn application domain, and system requirements and specifications). Only one design method (usage scenarios) was associated with task-mapping. It is surprising that usage scenarios (D) was not also associated with task-facilitation since scenarios can be used to explore the user's ability to follow the task structure and return to the task path when a deviant path has been taken. The one inspection method associated only with task-mapping, feature inspections, only checks individual system features.

It is not surprising that testing user performance is associated directly with task-facilitation because performance issues are directly related to the user's ability to follow the task structure through to completion. Although system requirements and specifications (SA) are weakly associated with task-facilitation (see Table 5.9), no design techniques focus specifically on the system's ability to facilitate task completion.

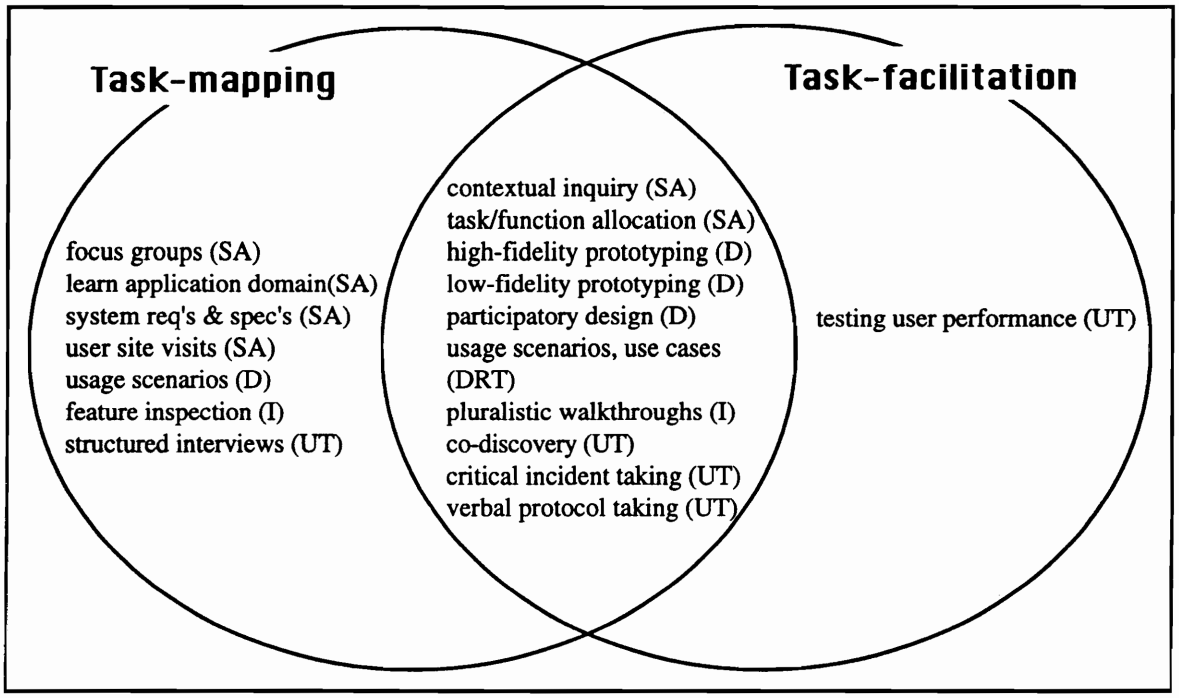

Task-mapping
Task-facilitation
focus groups (SA)
learn application domain(SA)
system req's & spec's (SA)
user site visits (SA)
usage scenarios (D)
feature inspection (I)
structured interviews (UT)
contextual inquiry (SA)
task/function allocation (SA)
high-fidelity prototyping (D)
low-fidelity prototyping (D)
participatory design (D)
usage scenarios, use cases (DRT)
pluralistic walkthroughs (I)
co-discovery (UT)
critical incident taking (UT)
verbal protocol taking (UT)
testing user performance (UT)

Figure 5.5. The overlap among activities for task-mapping and task-facilitation.

Activities that have been associated strongly with each component are illustrated in the Venn diagram in Figure 5.6. Six activities are strongly associated with both components. Of those six, two are design, one is an inspection method, and three are user testing. Surprisingly, no systems analysis activities nor design representation techniques are illustrated in the intersection. And, since guidelines activities were not associated with either task category in Figure 5.5 above, it is not unexpected that they are present only in the artifact side of Figure 5.6. The one inspection method included only for the artifact component is consistency inspections. No inspection methods are associated with the task component. Current consistency inspections do not often address task consistency issues.

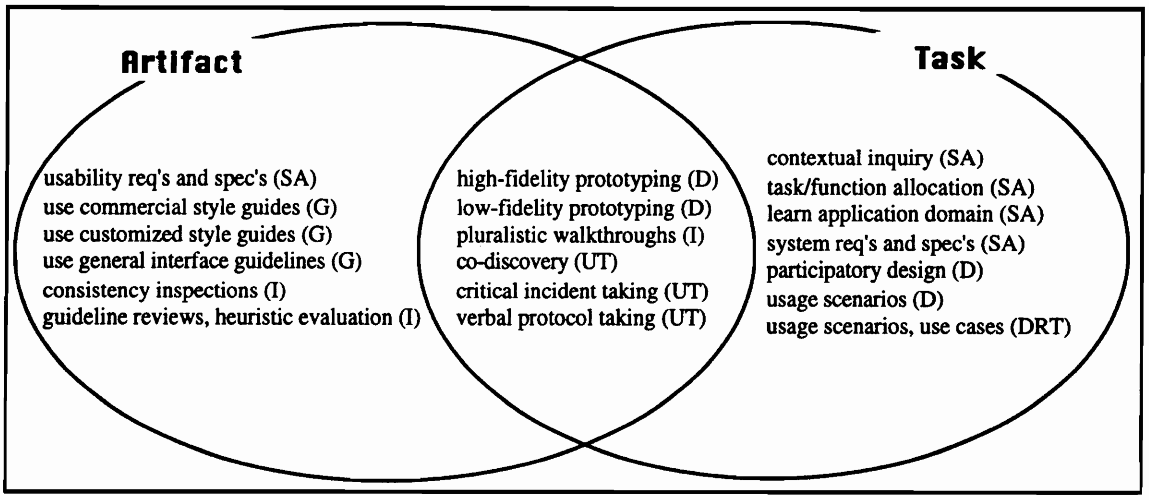

Artifact
Task
usability req's and spec's (SA)
use commercial style guides (G)
use customized style guides (G)
use general interface guidelines (G)
consistency inspections (I)
guideline reviews, heuristic evaluation (I)
high-fidelity prototyping (D)
low-fidelity prototyping (D)
pluralistic walkthroughs (I)
co-discovery (UT)
critical incident taking (UT)
verbal protocol taking (UT)
contextual inquiry (SA)
task/function allocation (SA)
learn application domain (SA)
system req's and spec's (SA)
participatory design (D)
usage scenarios (D)
usage scenarios, use cases (DRT)

Figure 5.6. The overlap among artifact-oriented activities and task-oriented activities.

Figure 5.6 illustrates that for the task component, developers have several activities that can be performed at specific points during development (analysis, design, and testing). However, few "intermediate" activities are noted that can be performed between analysis and design, and between design and testing. Only one systems analysis activity, several guidelines, design, inspection, and testing activities can be used to address problems in the artifact component.

Comparing Table 5.9, Figure 5.4 and Figure 5.5, it is observed that although task/function allocation was associated with visualness as well as with both categories in the task component, it is on the task "side" in Figure 5.6. Similarly, learning about the application domain and participatory design address language problems, but are on the task side of Figure 5.6 (participatory design is weakly associated with the artifact component). Although user site visits and testing user performance were not included in Figure 5.6, they were weakly associated with the task component (see Table 5.9). Structured interviews were associated with language problems as well as task-mapping problems; however, the values for the means and standard deviations did not place them in Figure 5.6.

Activities associated strongly with the overall UPT are given in the "Overall UPT" column in Table 5.10. Table 5.10 compares the "Overall" activities with activities that are associated with either the artifact component or task component *only*. The group of activities in the "Only AC" column is the set difference between the activities in the artifact oval in Figure 5.6 and the activities in the "Overall UPT" column in Table 5.19. Similarly, the group of activities in the "Only TC" column is the set difference between the activities in the task oval in Figure 5.6 and the activities in the "Overall UPT" column in Table 5.19.

For example, "usability requirements and specifications" is in the artifact oval in Figure 5.6, but it is not in the "Overall UPT" column in Table 5.19. Therefore, it is placed in the "Only AC" column in Table 5.10. Note that "use customized style guides" is in the artifact oval in Figure 5.6. However, since it is included in the "Overall UPT" column in Table 5.10, it is *not* in the "Only AC" column in Table 5.10. Similarly, "contextual inquiry" is in the task oval in Figure 5.6, but it is not in the "Overall UPT" column in Table 5.19. Therefore, it is placed in the "Only TC" column in Table 5.10. Note that "participatory design" is in the task oval in Figure 5.6. However, since it is included in the "Overall UPT" column in Table 5.19, it is *not* in the "Only TC" column in Table 5.10.

The nine activities that are considered the most "broadly" useful by the study participants are given in the "Overall UPT" column in Table 5.10. Note that although system requirements and specifications are weakly associated with the overall UPT, no other system analysis activities nor design representation techniques have been associated with usability problems in any UPT category. It is not unexpected that the activities that are associated only the task component are primarily system analysis activities. Those that address the artifact component are divided between one analysis activity, two guidelines activities, and one inspection method.

Table 5.10. A comparison of activities for the artifact component, task component, and overall UPT.

Only AC	Overall UPT	Only TC
usability req's & spec's (SA)	use customized style guides (G)	contextual inquiry (SA)
use commercial style guides (G)	high-fidelity prototyping (D)	learn application domain (SA)
use general interface guidelines (G)	low-fidelity prototyping (D)	system req's & spec's (SA)
consistency inspections (I)	participatory design (D)	task/function allocation (SA)
	guideline reviews, heuristic evaluation (I)	usage scenarios (D)
	pluralistic walkthroughs (I)	usage scenarios, use cases (DRT)
	co-discovery (UT)	
	critical incident taking (UT)	
	verbal protocol taking (UT)	

Seven activities are not associated (positively or negatively) with any category or combination of categories. These include two ~~are~~ system analysis activities (learn competing system and user analysis), one design representation technique (knowledge and model based), one inspection method (cognitive walkthroughs), and three user testing activities (alpha tests, beta tests, and user preferences). All guidelines and design activities are associated positively with some (or all) UPT categories. It is significant that learning about competing systems was not associated with any category since that activity can help developers learn about the way similar functionality has been presented to the user and can be used to anticipate usability problems with that functionality. Although user analysis can help developers focus the level of user interface complexity, at this time, user analysis is not related directly to types of usability problems.

Several activities were negatively associated with UPT categories. In the systems analysis group, recall that focus groups were positively associated with task-mapping, but have a weak negative association with manipulation problems. Systems analysis is negatively associated with visualness and manipulation (weak), and task/function allocation is negatively associated with visualness. It was not unexpected that behavioral design representation (including GOMS analysis) as well as state transition diagrams were negatively associated with the visualness category. In addition, feature inspections were negatively associated with each category in the artifact component. This is due to the fact that feature inspections do not focus on identifying artifact problems. Although behavioral design representation techniques and GOMS analysis (as an inspection method) have made an important contribution to the theory of human-computer interaction, these techniques did not fare well in this study, i.e., they were perceived not to be useful in addressing visualness problems and language problems and were not associated (either positively or negatively) with the manipulation, task-mapping, or task-facilitation categories. See Table 5.9.

5.4 A Comparison Of Roles And Activities

This section compares and contrasts the roles and activities that were positively associated with usability problems in each of the five primary UPT categories as well as roles and activities that were negatively associated with those categories. Roles and activities considered to be weakly associated with UPT categories are not included in the discussion. Roles and activities that are associated with problems in each component as well as those that address all types of problems are also examined.

Table 5.11 contains the list of roles and activities that can be used to address visualness usability problems. There are five roles and one systems analysis activity, three guidelines activities, two design activities, three inspection methods, and three testing methods. Human factors specialists, user interface interaction designers, and user interface evaluators each have training that would allow them to perform most of the identified activities. Two activities might require additional training: heuristic evaluation and pluralistic walkthroughs. The graphic designer can contribute during prototyping; however, there is no other activity that specifically draws on the skills of a graphic designer. In addition, technical writers are not trained in any of the activities. As mentioned in section 5.3, technical writers may have been included in this list for the purpose of layout text on the screen. Like the graphic designer, the technical writer could then contribute during prototyping.

Table 5.11. A comparison of roles and activities for the visualness category.

VISUALNESS		
Roles	**Activities**	**Type**
human factors specialist	usability requirements/specifications	SA
user interface interaction designer		
user interface evaluator	use commercial style guides	G
graphic designer	use customized style guides	G
technical writer	use general interface guidelines	G
	high-fidelity prototyping	D
	low-fidelity prototyping	D
	consistency inspections	I
	guideline reviews, heuristic evaluation	I
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	verbal protocol taking	UT

Table 5.12 contains the list of roles and activities that can be used to address language usability problems. There are two systems analysis activities, one guidelines activity, three design activities, no design representation techniques, three inspection methods, and four testing methods. The roles of human factors specialist, user interface interaction designer, and user interface evaluator were associated with the usability problems that are about the language used in the user interface. Team members performing these three roles can perform most of the activities without additional training.

The two remaining roles (technical writer and cognitive psychologist) are not well matched to the activities in Table 5.12. The role of technical writer is critical because language issues are a very important factor in system usability. Yet, there are no activities for which the technical writer is trained, and no activities in which the technical writer could be involved easily. Similarly, although cognitive psychologists are often trained in usability, they bring very different skills to the development team. Cognitive psychologists can also impact word usage; however, there are no activities that have been designed specifically to utilize this expertise.

Table 5.12. A comparison of roles and activities for the language category.

LANGUAGE		
Roles	**Activities**	**Type**
human factors specialist	learn application domain	SA
user interface interaction designer	user site visits	SA
user interface evaluator		
cognitive psychologist	use general interface guidelines	G
technical writer		
	high-fidelity prototyping	D
	low-fidelity prototyping	D
	participatory design	D
	consistency inspections	I
	guideline reviews, heuristic evaluation	I
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	structured interviews	UT
	verbal protocol taking	UT

Table 5.13 contains the list of roles and activities that can be used to address manipulation usability problems. It is not surprising that the roles of cognitive psychologist, human factors specialist, user interface interaction designer, and user interface evaluator were associated with the usability problems that are about the way the user manipulates objects in the user interface. Note that one systems analysis activity, three guidelines activities, two design activities, no design representation techniques, three inspection methods, and three testing methods were included. With the exception of the cognitive psychologist, the roles and activities listed below are well matched. While the skills of a cognitive psychologist are appropriate for addressing manipulation issues, such as visual cues, there are no activities for which this expertise is required.

Table 5.13. A comparison of roles and activities for the manipulation category.

MANIPULATION		
Roles	**Activities**	**Type**
human factors specialist user interface interaction designer user interface evaluator cognitive psychologist	usability requirements/specifications	SA
	use commercial style guides	G
	use customized style guides	G
	use general interface guidelines	G
	high-fidelity prototyping	D
	low-fidelity prototyping	D
	consistency inspections	I
	guideline reviews, heuristic evaluation	I
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	verbal protocol taking	UT

Table 5.14 contains the list of roles and activities that can be used to address task-mapping usability problems. The six roles were associated with the usability problems that occur because of the way the user tasks are mapped to the system. Three of the roles (end user, problem domain expert, systems analyst) were not associated with any artifact category, but could be used to address usability problems that are about user tasks. It is interesting that the role of moderator was not included, yet focus groups were associated with the task-mapping category. This may have occurred because the skills that a moderator possesses do not address task-mapping usability problems (the moderator's skills are focused on running a group discussion). However, since this activity is associated with this category, a role is needed that can be used to identify task-mapping problems while participating in the focus group.

There are six systems analysis activities, no guidelines activities, four design activities, one design representation technique, two inspection methods, and four testing methods. Human factors specialists, user interface interaction designers, and user interface evaluators are trained to perform the usability-related activities listed below. In addition to participating in the testing activities, the end user can contribute to contextual inquiry, focus groups, helping explain the application domain, system requirements and specifications, user site visits, and participatory design. The problem domain expert and systems analyst could likewise contribute to many of these same activities.

It is significant to note that the list of activities in Table 5.14 includes both usability-related systems analysis activities (that can be performed by usability experts) and traditional software engineering systems analysis activities (that can be performed by software engineers). The coverage in both roles and activities is indicative of the importance of building a system with the right functionality in which users' tasks are structured appropriately in the user interface.

Table 5.14. A comparison of roles and activities for the task-mapping category.

TASK-MAPPING		
Roles	**Activities**	**Type**
human factors specialist	contextual inquiry	SA
user interface interaction designer	focus groups	SA
user interface evaluator	learn application domain	SA
end user	system reqs./specs.	SA
problem domain expert	task/function allocation	SA
systems analyst	user site visits	SA
	high-fidelity prototyping	D
	low-fidelity prototyping	D
	participatory design	D
	usage scenarios	D
	usage scenarios, use cases	DRT
	feature inspection	I
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	structured interviews	UT
	verbal protocol taking	UT

Table 5.15 contains the list of roles and activities that can be used to address task-facilitation usability problems. Note that there are two systems analysis activities, no guidelines activities, three design activities, one design representation technique, one inspection method, and four testing methods. While the roles and activities for task-facilitation are matched similarly to those of task-mapping, it is significant to note that participatory design is included as an activity; however, the role of end user is not associated with this category. User involvement in the design process is the basis for participatory design. Of the roles listed, it is possible that the problem domain expert could perform the role of user during this activity.

Table 5.15. A comparison of roles and activities for the task-facilitation category.

TASK-FACILITATION		
Roles	**Activities**	**Type**
human factors specialist	contextual inquiry	SA
user interface interaction designer	task/function allocation	SA
user interface evaluator		
problem domain expert	high-fidelity prototyping	D
systems analyst	low-fidelity prototyping	D
	participatory design	D
	usage scenarios, use cases	DRT
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	testing user performance	UT
	verbal protocol taking	UT

Table 5.16 compares the roles and activities associated with the artifact component. With the exception of the role of technical writer (which is not associated with a specific activity that addresses artifact usability problems), note that a good mapping exists between roles and activities. Also note that only one systems analysis activity is present and no

design representation techniques have been associated with usability problems classified in the artifact component.

Table 5.16. A comparison of roles and activities for the artifact component.

ARTIFACT COMPONENT		
Roles	**Activities**	**Type**
human factors specialist	usability req's. & spec's.	SA
user interface interaction designer		
user interface evaluator	use commercial style guides	G
technical writer	use customized style guides	G
	use general interface guidelines	G
	high-fidelity prototyping	D
	low-fidelity prototyping	D
	consistency inspection	I
	guidelines review, heuristic evaluation	I
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	verbal protocol taking	UT

The roles and activities that address problems in the task component are given in Table 5.17 below. As mentioned above, the skills of the problem domain expert and systems analyst are only directly applicable to some of the activities in the list. However, there is good coverage between the roles associated with the task component and the activities that can be performed.

Table 5.17. A comparison of roles and activities for the task component.

TASK COMPONENT		
Roles	**Activities**	**Type**
human factors specialist	contextual inquiry	SA
user interface interaction designer	task/function allocation	SA
user interface evaluator	learn about application domain	SA
problem domain expert	system req's. & spec's.	SA
systems analyst		
	high-fidelity prototyping	D
	low-fidelity prototyping	D
	participatory design	D
	usage scenarios	D
	usage scenarios, use cases	DRT
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	verbal protocol taking	UT

It is significant that the role of cognitive psychologist was not associated with either UPT task category. This kind of expertise could be used to address task problems, especially task-facilitation problems that are about error prevention and the user's ability to recover from errors when they occur. Cognitive psychologists often contribute through research rather than practice. For cognitive psychologists to become an integral part of the development team, more research is needed to develop activities that utilize their expertise.

A final inspection of the roles and activities data reveals four critical roles and nine important activities for the entire UPT structure. These roles and activities are presented in Table 5.18 below. Three of these roles (human factors specialist, user interface interaction designer, and user interface evaluator) were associated with every UPT category. Note that the role of systems analyst was included primarily because of the importance of user tasks to overall system acceptance.

The nine activities focus on guidelines, design, and evaluation (inspection and user testing). Not only are design representation techniques absent from this list, but no specific systems analysis activities are listed that utilize the system analyst's skills. In addition, the activity "pluralistic walkthroughs" is associated with every UPT category; however, users participate in this activity, and the role of user is not associated with the UPT (in fact, the role of end user was only associated with the task-mapping category).

Table 5.18. A comparison of roles and activities for the entire UPT structure.

ACROSS UPT		
Roles	**Activities**	**Type**
human factors specialist	use customized style guides	G
user interface interaction designer		
user interface evaluator	high-fidelity prototyping	D
systems analyst	low-fidelity prototyping	D
	participatory design	D
	guideline reviews, heuristic evaluation	I
	pluralistic walkthroughs	I
	co-discovery	UT
	critical incident taking	UT
	verbal protocol taking	UT

In the above discussion, the number of activities (of each type) identified for each category was mentioned briefly. These results are summarized in Table 5.19 below. The row totals indicate how many times a specific type of activity is associated with individual UPT categories. For example, consider the two row totals that are lower than the others (guidelines and design representation techniques). A row total of 11 means that individual guidelines activities (there were three on the Activities Form) were associated 11 times with individual UPT categories and combinations of those categories. Similarly, a row total of three for design representation techniques means that individual techniques included on the Activities Form (there were five) were associated only three times with UPT categories.

A low row total can be interpreted as follows: individual activities of that type were associated minimally with several categories. The guidelines activities were associated minimally with language (1), task-mapping (0), task-facilitation (0), the task component (0), and the UPT (1). Design representation techniques were likewise minimally associated with all categories.

A count of zero or one (in a specific cell) indicates that developers do not have a choice of activities to apply to usability problems within a given category. For example, developers wanting to use a system analysis activity to address artifact problems have only one to choose from. Similarly, developers have no design representation techniques that address problems in the artifact component. To provide developers with a choice of activities in each type, new activities are needed.

Table 5.19. Number of each type of activity associated with UPT categories.

Activity	**V**	**L**	**M**	**TM**	**TF**	**AC**	**TC**	**UPT**	**Total**
systems analysis	1	2	1	6	2	1	4	0	17
guidelines activities	3	1	3	0	0	3	0	1	11
design activities	2	3	2	4	3	2	4	3	23
design representation techniques	0	0	0	1	1	0	1	0	3
inspection methods	3	3	3	2	1	3	1	2	18
testing methods	3	4	3	4	4	3	3	3	27
Total	12	13	12	17	11	12	13	9	99

Throughout this chapter, roles and activities that do not address usability problems in specific UPT categories (and combinations of those categories) were identified. These results are summarized in Table 5.20 below. The roles and activities listed in Table 5.20 received a strong negative rating by the experts ("—"). Roles and activities with a weak negative rating ("w—")were not included.

With the exception of market analyst, the roles that were negatively associated with UPT categories were traditional software engineering roles. Market analysts may also examine the prospective product from a usability perspective. Although some software engineering activities were included in the list below, several usability engineering activities were also negatively associated with specific categories[2]. For example, feature inspections and behavioral design representation techniques do not address visualness problems. Usage scenarios and use cases do not address manipulation problems.

5.5 Process Improvement Using The UPT

The UPT can be used as a diagnostic tool for usability-related process improvement. A step-by-step procedure that can be used by managers and developer is outlined below.

1. Make a list of activities that have been performed (or will be performed) on a given development project.
2. Make a list of the roles and skills of development team members (this list should also include skills and expertise that the development team has access to during development).
3. Classify each usability problem in the UPT. This yields an artifact classification and a task classification for each problem.
4. Obtain a distribution of problems over the three primary categories in the artifact component of the UPT. Obtain a distribution of problems over the two primary

2 It should be noted that the Activities Form in Appendix C contains more usability engineering activities than software engineering activities.

categories in the task component of the UPT. Note those categories that contain a large percentage of problems.

Table 5.20. Roles and activities that do not address UPT categories.

	NEGATIVE	ASSOCIATION
Category	**Roles**	**Activities**
visualness	software designer software documentor software implementor software librarian	systems analysis (SA) task/function allocation (SA) behavioral design reprs'n (DRT) state transition diagrams (DRT) feature inspection (I) GOMS analysis (I)
language	market analyst software designer software documentor software implementor software librarian	state transition diagrams (DRT) feature inspection (I) GOMS analysis (I)
manipulation	market analyst marketer software documentor software librarian	functional analysis (SA) usage scenarios, use cases (DRT) feature inspection (I)
task-mapping	quality assurance software documentor software implementor software librarian software tester	consistency inspection (I) standards inspection (I)
task-facilitation	marketer software documentor software librarian software tester	standards inspection (I)
artifact component	marketer software documentor software librarian	task/function allocation (SA) feature inspection (I)
task component	marketer software documentor software librarian software tester	standards inspection (I)
UPT	marketer software documentor software librarian	

5. Compare the skills available during development with the roles and skills needed to address problems in the categories that contain a large percentage of problems. Identify additional roles and skills that are needed to address those problems.

6. Compare the activities currently performed during development with the activities that can be used to address problems in the categories that contain a large percentage of problems. Identify activities that could be incorporated into the development process to address those problems.
7. Examine the roles, skills, and activities that, in the opinion of the experts, do not address specific UPT categories. If these roles, skills, and activities are currently used during development, do not expect them to address specific categories of usability problems.

This procedure can be performed during and after development. If it is performed during development, process improvement can be effected before the product is completed. If it is performed after development, process improvements can be incorporated on the next development effort.

The results presented in this chapter can be used to focus process improvement in two ways. First, a set of roles and a set of activities can be identified that will address usability problems in specific UPT categories or multiple categories. Second, roles and activities that do not address problems in specific categories can likewise be identified. These two results are important as they help managers and developers choose the *right* roles and activities for a given development effort.

5.6 Summary

This chapter presented the findings of a study designed to identify associations between developer roles and skills and the UPT, as well as associations of development activities, methods, and techniques with the UPT. Both positive and negative associations were identified.

Four roles were associated with the entire UPT structure: human factors specialist, user interface interaction designer, user interface evaluator, and systems analyst. Two additional roles (technical writer and problem domain expert) were identified for the artifact and task components, respectively.

Nine activities were found to address the entire UPT structure. Note that no systems analysis activities were included; however, there were one guidelines activity, three design activities, two inspection methods, and three user testing activities (see Table 5.10). The systems analysis activities were associated primarily with the task component, guidelines activities were associated primarily with the artifact component.

The roles and activities with strong positive associations were compared. Gaps and inconsistencies were identified, i.e., roles with no corresponding activity, activities with no associated role. For example, the role of systems analyst was not paired with any of the nine activities (see Table 5.18).

To determine if software engineering roles can be used to address usability problems if usability experts are not available, six experts were asked to assess the degree to which various roles and skills could address problems in individual UPT categories.

The experts completed two forms (Roles Form 1 and Roles Form 2). Roles Form 1 contained both software engineering and usability engineering roles. Roles Form 2 contained only software engineering roles found on Roles Form 1. The instructions for Roles Form 2 assumed that no usability experts were available. The responses on Roles Form 2 were found to be significantly higher than the corresponding responses on Roles Form 1. However, several factors that could have influenced the significance of this result were discovered and discussed. As a result, the conclusion was reached that although the there was a significant difference in the responses, there may not be any practical importance that can be attached to the result.

In addition, a step-by-step procedure for using the UPT as a diagnostic tool for usability-related process improvement was outlined. This procedure is based on the results obtained during the Roles Evaluation and the Activities Evaluation. It enables managers and developers to identify weaknesses in the user interface development process. The results presented in this chapter can then be used to formulate strategies for process improvement that are based on the addition of individual roles and skills to the development team and incorporating specific development activities into the software process.

6 SUMMARY AND CONCLUSIONS

This research developed a taxonomic model for understanding, describing, comparing, and analyzing usability problems, and a framework for understanding the relationship of usability problems to development context. The work was divided into two steps. Step one involved the development of an empirically-derived taxonomic model in which usability problems were classified, and evaluation of the taxonomy's reliability in classification. Step two focused on establishing associations between taxonomic categories and two aspects of development context: developer factors (roles and skills) and activity factors (development activities, methods, and techniques). This chapter summarizes the approach taken during each step, presents the findings of each step, and outlines the contributions from each step. Other discoveries are also discussed. The chapter concludes with a few brief remarks about this research effort.

6.1 Step One: The Usability Problem Taxonomy (UPT)

This section summarizes the development and evaluation of the Usability Problem Taxonomy and outlines the contributions of the UPT as well as additional discoveries that were made during the development of the UPT. The organization is as follows. Section 6.1.1. describes the approach used to develop the UPT. Section 6.1.2 outlines the findings of the UPT evaluation. Section 6.1.3 presents the contributions of the UPT. Section 6.1.4 describes other discoveries that were made during this step.

6.1.1 Summary Of UPT Development

Five projects were surveyed at four real-world, interactive software development organizations. A total of 645 usability problem descriptions were collected from those five projects. The problems were identified during usability evaluations conducted by the author, a colleague, and/or developers. Of the 645 descriptions, 406 were analyzed heuristically, i.e., classified according to the usability heuristic(s) violated. Heuristic analysis did not impact UPT development directly; however, it was used to organize the 406 descriptions for the next phase in taxonomy construction.

The UPT was developed using a bottom-up, iterative empirical approach. The problem descriptions were organized according to the heuristic(s) they violated, as well as one additional category for descriptions that could not be classified according to any heuristic. The problems were then reexamined and grouped according to commonalities. A hierarchical structure emerged among the groups of problems. The hierarchical structure, the UPT, contains two components (artifact and task), five primary categories, and 21 subcategories. The two components of the UPT correspond to two autonomous dimensions of a usability problem. During classification, each problem is classified in the artifact component and in the task component.

6.1.2 UPT Evaluation

A controlled study was undertaken that showed that the UPT can be used reliably to classify usability problems to level 2 of the taxonomy (recall that level 2 contains the five primary categories of visualness, language, manipulation, task-mapping, and task-facilitation). Seven classifiers from industry, government, and academic development environments with substantial interactive software development expertise participated in the study by classifying 20 usability problems that were randomly selected from the group of 645 problems. Ten of the 20 problems had been analyzed by the author prior to the onset of the study (these were selected from the 406 that had been examined previously). The remaining 10 had not been analyzed previously and were selected randomly from the 239 that had been examined previously.

To demonstrate UPT reliability, the kappa statistic, κ (the proportion of agreement after chance agreement is removed from consideration), was used to assess the level of agreement among classifiers. Three pairs of hypothesis tests were performed (six tests in all). One test in each pair was based on the artifact component and one was based on the task component. The first pair of tests was based on all 20 problems. The second pair of tests was based on the 10 problems that had been previously examined (referred to as "old"). The third pair was based on the 10 problems that had not been previously examined (referred to as "new").

The findings of the UPT reliability study showed that the UPT can be used to classify usability problems reliably to level 2 (the level of the five primary categories). The first pair of tests (all 20 problems) showed that there was agreement greater than chance in the artifact classifications ($\kappa = .403$) and the task classifications ($\kappa = .095$). Likewise, the second pair of tests (the 10 "old" problems) showed that there was agreement greater than chance in both the artifact classifications ($\kappa = .357$) and the task classifications ($\kappa = .112$). The third pair of tests (the 10 "new" problems) showed that there was agreement greater than chance in the artifact classifications ($\kappa = .437$), but for the task classifications, the test did not show that there was agreement greater than chance. Instead, the last test showed that there was some evidence to suggest that there may be agreement greater than chance in the task component ($\kappa = .068$). However, the level of agreement is not statistically significant at the .05 level of significance.

Several factors were identified that impacted the findings discussed above. These factors included lack of classifier experience with the application domains from which the problems were selected, lack of familiarity with the UPT, the fact that the classifiers did not observe each problem as it occurred, and vague and incomplete problem descriptions. In addition, the novelty of the UPT and its approach to problem classification (especially in the task component) also affected the classification. Therefore, in practical application, with training and increased familiarity, the reliability with which the UPT can be used to classify usability problems should be greater than that found in this study.

6.1.3 Contributions Of The UPT

The primary contribution of step one in this research project is the development of the UPT, a taxonomic model for classifying usability problems reliably. The UPT enables developers and evaluators to understand, describe, compare, and analyze usability problems. The model was built empirically and is comprised of descriptive categories. Each category contains explanations and descriptions of the type of problems it contains as well as descriptions of the type of problems it does not contain.

6.1.4 Additional Discoveries Made During UPT Development

Additional discoveries were made by the author during the development of the UPT. These additional discoveries relate to new evaluation methods, heuristic classification, four properties of a classification scheme, four characteristics of usability problem descriptions, using the UPT to generate higher quality problem descriptions, and data organization and analysis. Each discovery is described briefly below.

Object-based Evaluation

Object-based evaluation is a recent inspection method developed by the author that focuses on problems associated with specific user interface objects. Each type of user interface object is examined one at a time. Problems are noted and associated with a specific type of object. This method was developed during an initial data collection effort in which usability problems were identified on a real-world development project by the author and a colleague.

Using this inspection method is advantageous for several reasons. First, it is easy to learn. Second, it is less expensive than user testing activities. Third, it focuses on problems with the artifacts in the user interface. Additional research is needed to compare this method with other inspection methods and to determine if it can be used to detect task-related usability problems.

Heuristic Classification, The UPT, And Four Properties Of A Classification Scheme

Four important properties of a classification scheme were discovered empirically during heuristic analysis. These properties are distinguishability, mutual exclusiveness, specificity, and completeness. It was concluded that the usability heuristics were not an effective classification scheme since the heuristics did not satisfy these four properties.

To examine the UPT with respect to these four properties, the author classified all 645 usability problems on the five development projects. Distinguishability is satisfied for these problems, i.e., different types of problems were not classified in the same category or in the same problem type. The UPT categories are defined to be mutually exclusive, i.e., for the 645 problems, this property was never violated. The UPT also satisfies the specificity property. Each category contains many examples of specific usability problems that would be classified in that category as well as examples of problems that would not be classified in that category. And, the UPT categories are arguably complete (to level 2 containing the five primary categories). The five primary categories capture problems users have viewing, reading, and manipulating artifacts in the interface as well as problems related to task structure and the user's ability to complete the task. No new primary categories were needed to classify the 645 usability problems. It should be noted that additional subcategories may be needed for application domains very different from those surveyed, e.g., virtual reality software, or systems for the physically challenged. The conclusion that the UPT satisfies these properties was arrived at empirically by the author.

Four Characteristics Of Usability Problem Descriptions

Four important characteristics of a high quality problem description were identified empirically (by the author) during classification of the 645 usability problems. The four characteristics are clarity, precision, comprehensiveness, and problem-centeredness. Descriptions lacking these characteristics are not only vague and incomplete, but often contain information that describes the effect on the user and user reactions. Low quality descriptions are difficult to understand, think about, compare, and analyze. It is also difficult for developers to take low quality descriptions and consistently generate good solutions that raise the level of usability in the user interface.

Better Problem Descriptions

According to the experience of the author and several colleagues, a working knowledge of UPT categories helps developers write problem descriptions that are clear, precise, comprehensive and problem-centered. By thinking about the artifact and task dimensions of each usability problem as well as the UPT categories, necessary information that may be overlooked is captured in the problem description.

Several examples were presented in Chapter 4 that illustrate how the UPT can improve problem descriptions. Because using the UPT during problem identification and description helps developers and evaluators write higher quality problem descriptions, the loss of information described in Chapter 1 is minimized (see section 1.2).

Data Organization And Analysis

The UPT provides new ways to organize and identify patterns of usability problems. Classification results in groupings according to category. Groupings across categories can also identified (e.g., consistency issues across categories). The UPT can also be used with additional kinds of information (user tasks, user interface objects) to group usability problems. These new organizational strategies can be used to develop solutions for individual problems as well as solutions that address multiple problems simultaneously.

In current practice, usability problems are prioritized prior to correction. The groupings provided by the UPT can be used formulate new prioritization strategies.

Usability problems can be prioritized by category, within each category, and across categories.

6.2 Step Two: Associations Between UPT Categories And Development Context

This section summarizes a study in which associations between UPT categories and two development context factors were identified and outlines the findings and contributions of that study. This section is organized as follows. Section 6.2.1. describes the association study. Section 6.2.2 outlines the findings of the study. Section 6.2.3 presents the contributions made during the study.

6.2.1 Summary Of The Association Study

A study was undertaken in which associations between each of the five primary UPT categories and two aspects of development context (developer roles and skills, and development activities, methods, and techniques) were identified. Six experts with extensive usability expertise and industry experience participated in this phase of the research. Each assessed the degree to which individual roles or activities addressed problems in each of the five primary UPT categories (level 2 of Figure 3.1). For the purposes of this study, addressed meant prevent, think about, detect, predict, and/or correct.

The mean and standard deviation of the responses were analyzed. The means and standard deviations were interpreted as follows: strong positive association, weak positive association, no association, weak negative association, and strong negative association.

Pairs of means and standard deviations were grouped according to the interpretation so that roles and activities that do address usability problems in a given category or combination of categories are distinguished from those roles and activities that do not address usability problems in a given category.

6.2.2 Association Study Findings

The findings in this study are summarized in Table 6.1. The entries in the table are the roles and activities for which a strong positive association with UPT categories was noted. The categories and combinations of those categories are given in the leftmost column. UPT categories are abbreviated as follows: visualness (V), language (L), manipulation (M), artifact component (AC), overall UPT (UPT), task component (TC), task-mapping (TM), and task-facilitation (TF). The rightmost column (type) contains abbreviations for the type of activity on that line in the table. For example, learn application domain is a system analysis activity (SA).

The role and activity columns in the middle of Table 6.1 can be interpreted in the following way. The roles or activities listed for the five primary categories at level 2 are applied only to those categories (V, L, M, TM, TF), e.g., the role of graphic designer is related to visualness problems and the activity "learn application domain" is related only to language problems. The roles or activities listed for the each component (AC and TC) are related to the entire component, i.e., the role of technical writer is related to the artifact component while the activity of "contextual inquiry " is related to the task component. The roles or activities listed for the overall UPT address problems in both components, e.g., the role of a human factors specialist and the activity " use customized style guides" can be used to address both artifact and task related problems.

Table 6.1. Summary of roles and activities that address UPT categories.

Category	Role	Activity	Type
V	graphic designer		
L	cognitive psychologist	learn application domain	SA
		user site visits	SA
		participatory design	D
		structured interviews	UT
M	cognitive psychologist		
AC	technical writer	usability req's & spec's	SA
		use commercial style guides	G
		use general interface guidelines	G
		consistency inspections	I
UPT	systems analyst	use customized style guides	G
	human factors specialist	high-fidelity prototyping	D
	user interface interaction designer	low-fidelity prototyping	D
	user interface evaluator	participatory design	D
		guideline reviews, heuristic evaluation	I
		pluralistic walkthroughs	I
		co-discovery	UT
		critical incident taking	UT
		verbal protocol taking	UT
TC	problem domain expert	contextual inquiry	SA
		learn application domain	SA
		system req's & spec's	SA
		task/function allocation	SA
		usage scenarios	D
		usage scenarios, use cases	DRT
TM	end user	focus groups	SA
		user site visits	SA
		feature inspections	I
		structured interviews	UT
TF		testing user performance	UT

Approached from a slightly different perspective, if a manager wants to add one role to the development team to address problems in the language category, he/she can choose from either cognitive psychologist (in the language row), technical writer (in the artifact component row), or from human factors specialist, user interface interaction designer, user interface evaluator (in the UPT row). The manager would not choose the role of systems analyst since this role is only strongly associated with the task component (see Figures 5.1 and 5.2). If the manager wants to add one activity to the development process that addresses problems in the task-mapping category, he/she can choose from either focus groups, user site visits, feature inspections, or structured interviews (in the task-mapping row), any one of the six activities listed in the task component row, or any one of the nine activities in the UPT row.

Roles that do not address UPT categories were also identified. The roles for which a negative association was noted are traditional software engineering roles. Three roles were not associated with any UPT category (marketer, software documentor, software librarian). Additional roles were associated negatively with either an individual category or with an entire component. These roles were quality assurance, software implementor and software tester for the task component, and market analyst and software designer for the artifact component.

Similarly, activities that do not address UPT categories were noted. No guidelines activities were associated with the two categories in the task component. In addition, the activities for which a negative association was noted include both traditional software engineering and user interface engineering activities. These include systems analysis, behavioral design representation, feature inspections, and GOMS analysis (as an inspection method).

6.2.3 Contributions Of The Association Study

The association study contributes in three ways. First, this study provides the first direct link between specific types of usability problems and two development context factors. Second, a step-by-step procedure was developed that uses the UPT as a diagnostic tool to identify weaknesses in the current software process and outline process improvement strategies. Third, gaps and inconsistencies detected between roles and activities serve as a guide to future research. Each contribution is discussed briefly below.

This study provides the first direct link between specific types of usability problems and two development context factors: developer roles and skills, and development activities, methods, and techniques. This link shows that usability problem data is an important component of user interface process improvement. Developers can now use data available on every interactive software project to identify strategies for adjusting the composition of the development team and improving the interactive software development process.

A step-by-step procedure was developed that uses the UPT as a diagnostic tool to identify weaknesses in the current software process. Incremental process improvement strategies are then formulated that are based on selecting team members and development activities appropriate for a specific development project. To formulate a process improvement strategy, a manager or developer would first use the UPT to classify usability problems detected on a given development project. He/she would then identify the UPT categories that contain the largest number of problems. Based on the identified categories, the manager selects new roles and/or activities that can be used to address problems in those specific categories.

By comparing the roles and activities for which a strong positive association exists with UPT categories, gaps and inconsistencies were detected. The primary gaps and inconsistencies are outlined below:

- Three roles were associated with individual categories (graphic designer, cognitive psychologist, and technical writer), yet no activities have been designed specifically to take advantage of the skills and expertise of each role.

- Pluralistic walkthroughs were associated strongly with each of the five primary categories; however, the end user participates in this activity when it is performed during development, yet this role was associated only with the task-mapping category.

- Participatory design was associated with three categories: language, task-mapping, and task-facilitation. The role of end user is important in the performance of this particular activity, yet this role was not associated with either language or task-facilitation. This may be due to type of role the end user plays during participatory design.

- Although GOMS analysis is an important contribution to the theory of human-computer interaction, it was not perceived by the experts to be useful in addressing visualness and language problems and was not associated (either positively or negatively) with manipulation, task-mapping, or task-facilitation problems.

- Few design representation techniques were perceived by the experts to address usability problems in individual UPT categories.

- Task-related user interface design is perceived by many to be one of the most difficult aspects of user interface development. As a result, several systems analysis activities, design activities, and testing activities were strongly associated with the task component. However, no guidelines activities, only one design representation technique, and one inspection method are believed to address this component. While developers have several activities that can be performed at specific points during task-related development (analysis, design, and testing), there are few "intermediate" activities that can be performed between analysis and design, and design and testing.

- Only one systems analysis activity was associated with the categories in the artifact component (usability requirements and specifications).

This study showed that although specialized roles such as graphic designer, technical writer, cognitive psychologist, and end user are perceived to be important (each was associated with individual UPT categories), there is no clear way to utilize these skills on a development team. The study also indicated that, ***at the present time***, the roles of human factors specialist, user interface interaction designer, and user interface evaluator are considered to be valuable roles in practice. Not only were they associated with every UPT category and combination of those categories, but they have the background and training to perform the user interface development activities given in each list.

6.3 Final Remarks

This research reaffirmed the importance of empiricism in software engineering and human-computer interaction research. Empiricism was an integral aspect of this research effort as real-world data and experience were incorporated into both steps of this project. By working with industry, government, and academia, not only were relevant concerns identified, but invaluable quantitative and qualitative data were provided. These data were obtained from participants as they commented on the UPT and its usefulness as well as on the roles and activities selected for the association study.

This research has contributed significantly to furthering the science of human-computer interaction. The gaps and inconsistencies detected between roles and activities outline areas for future research. The directions for future research, including specific projects already underway, are described briefly in Chapter 7.

7 FUTURE WORK

This work provides the foundation for many different types of research projects. This chapter outlines these projects. Section 7.1 describes additional reliability studies planned for the UPT. Refinements and extensions to the UPT are described in section 7.2. Section 7.3 examines projects that investigate using the UPT to organize different types of information about usability problems. Further investigation of a new user interface inspection method (described in section 3.1) is outlined in section 7.4. Plans already underway for a toolset based on the UPT are outlined in section 7.5. These projects are summarized briefly in section 7.6.

7.1 Additional Reliability Studies

An additional reliability study is planned to assess the level of agreement achieved in UPT categories in levels 3 and 4 (see Figure 3.1). Although sufficient data were collected for the reliability study on levels 1 and 2 of the UPT described in section 4.1, a larger data set is needed to assess the level of agreement in levels 3 and 4 of the UPT (the samples in each category in levels 3 and 4 must be sufficiently large). Possible approaches include enlarging the number of classifiers, the number of usability problems, or both.

Other reliability-related projects are also under consideration. One project focuses on whether or not the reliability of the artifact and task components is sensitive to the application domain. Others are planned that investigate the effect of classifier expertise, knowledge of the application domain, and familiarity with the UPT on UPT reliability.

7.2 Refinements And Extensions To The UPT

To improve UPT reliability, refinements are planned that focus on category names and explanations provided in the UPT document. Classifier comments made during the reliability study indicate that some terminology used in the task component needs to be reexamined and possibly modified. In particular, the name "task-facilitation" may need to be changed to prevent classifiers from imbuing that category with additional meaning (see discussion in section 4.1.3). In addition, the concepts in the task component were so new to classifiers (even those with significant usability expertise), that additional explanations and examples are needed to further clarify this component. Additional explanations and examples are also needed for two categories in the artifact component: presentation of information/results (in visualness) and user-requested information/results (in language, other wording).

Extensions to the UPT web document are planned that will raise the level of usability achieved in the UPT version on the World Wide Web. Plans are underway to provide a search engine for the glossary, include additional definitions, and improve the user's access to individual UPT categories via the "fast page."

7.3 Information About Usability Problems

The UPT can be used to organize various types of information about usability problems. Information of interest includes problem severity, the cost associated with problem correction, and an assessment of problem importance. As this information is collected, it can be organized according to UPT categories. Problem distributions based on severity, cost, and importance across UPT categories can then be calculated. One such project is nearing completion. This particular project compares and contrasts problem distributions across development organizations, projects, and application domains.

7.4 New Methods

One new inspection method was developed early in this research (object-based evaluation). This evaluation method focused on user interface objects (see section 3.1). One project will compare the types of problems detected using object-based evaluation with the kinds of problems detected using other user interface evaluation methods. Additional projects focus on the development of data analysis techniques that can be associated with this method.

7.5 Tools

Plans are underway for an entire toolset that allows developers to use the UPT to classify usability problems and collect classification data on line. The web version of the UPT will be interfaced to individual tools that can

- store the classification data in a database,
- access that data,
- summarize the data,
- statistically analyze the data, and
- display summary and analysis results.

Additional functionality will include the collection of information relevant to object-based evaluation.

7.6 Summary

The results obtained during this research effort provide the basis for many, varied future projects. Several projects focus on the UPT. Some of these projects investigate UPT reliability at levels 3 and 4, while others concentrate on clarifying UPT terminology and explanatory information. Additional projects are planned that will improve the level of usability achieved in the web version of the UPT. Three other directions for future research are likewise based on the UPT. These include developing new usability problem analysis techniques, new evaluation methods, and a UPT toolset.

REFERENCES

[Basili 87] Basili, Victor, R. and Rombach H. D. "Tailoring the Software Process to Project Goals and Environments". *Proceedings of the 9th International Conference on Software Engineering*. 1987.

[Bias 94] Bias, Randolph G., and Deborah J. Mayhew, Editors. *Cost-Justifying Usability*. Academic Press, Inc. Boston. 1994.

[Boehm 81] Boehm, B. W. *Software Engineering Economics*. Prentice-Hall. Englewood Cliffs, New Jersey. 1981.

[Borenstein 91] Borenstein, Nathaniel S. *Programming as if People Mattered*. Princeton University Press. 1991.

[Bowen 94] Bowen, Karen C., and H. Rex Hartson. "An Evaluation of KMS". Internal document to become a technical report. Department of Computer Science. Virginia Tech. September, 1994.

[Bowen 96] Bowen, Karen C. Personal communication. May, 1996.

[Brooks 75] Brooks, Fred. P., Jr. *The Mythical Man-Month*. Addison-Wesley. Reading, Massachusetts. 1975.

[Brooks 94] Brooks, Patricia. "Adding Value to Usability Testing". *Usability Inspection Methods*. Edited by Jakob Nielsen and Robert L. Mack. John Wiley & Sons, Inc. 1994. pages 255-272.

[Card 93] Card, David H. "Defect-causal Analysis Drives Down Error Rates". *IEEE Software*. July, 1993. pages 98-99.

[Card 83] Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction*. Hillsdale, NJ. Erlbaum. 1983.

[Carroll 90] Carroll, J. M. and Mary Beth Rosson. "Human-computer interaction scenarios as a design representation". *Proceedings of the IEEE HICSS-23. 23rd Hawaii international Conference on System Sciences*. Vol. II. pages 555-561.

[Carroll 91] Carroll, J. M., Kellogg, W. A., and Mary Beth Rosson. "The task-artifact cycle." *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press, Cambridge, U.K. pages 74-102.

[Carroll 92] Carroll, J. M. and Mary Beth Rosson. "Getting around the task-artifact cycle: How to make claims and design by scenario". *ACM Transactions on Information Systems*. Vol. 10. pages 181-212.

[Carroll 95] Carroll, J. M. (editor) "Introduction: The Scenario Perspective on System Development." *Scenario-Based Design: Envisioning Work and technology in System Development.* John Wiley & Sons, Inc. New York 1995.

[Cohen 60] Cohen, Jacob. "A Coefficient of Agreement for Nominal Scales." *Educational and Psychological Measurement.* Vol. XX. No. 1. 1960. pages 37-46.

[CMM 93a] "Capability Maturity Model". CMU/SEI-93-TR-24. 1993.

[CMM 94b] "CMM Practices". CMU/SEI-93-TR-25. 1993.

[Cox 94] Cox, Mary E. and Paige O'Neal. "UPAR Analysis: Dollar Measurement of a Usability Indicator for Software Products". *Cost-Justifying Usability.* Edited by Randolph G. Bias and Deborah J. Mayhew. Academic Press. Boston. 1994. pages 145-158.

[Curtis 92] Curtis, Bill and Bill Hefley. "Defining a Place for Interface Engineering". *IEEE Software.* March, 1992. pages 84-86.

[Desurvire 94] Desurvire, Heather W. "Faster, Cheaper!! Are Usability Inspection Methods as Effective as Empirical Testing?". *Usability Inspection Methods.* Edited by Jakob Nielsen and Robert L. Mack. John Wiley & Sons, Inc. 1994. pages 173-202.

[Dray 94] Dray, Susan M. and Clare-Marie Karat. "Human Factors Cost Justification of an Internal Development Project". *Cost-Justifying Usability.* Editors: Bias, Randolph G., and Deborah J. Mayhew. Academic Press, Inc. Boston. 1994. pages 111-122.

[Dumas 94] Dumas, Joseph S. and Redish, Janice C. *A Practical Guide to Usability Testing*. Ablex Publishing Corporation. Norwood, New Jersey, 1994.

[Ehrlich 94] Ehrlich, Kate, and Janice Anne Rohn. "Cost Justification of Usability Engineering: A Vendor's Perspective". *Cost-Justifying Usability.* Editors: Bias, Randolph G., and Deborah J. Mayhew. Academic Press, Inc. Boston. 1994. pages 71-110.

[Fleiss 71] Fleiss, Joseph L. "Measuring Nomial Scale Agreement Among Many Raters." *Psychological Bulletin.* Vol. 76. No. 5. 1971. pages 378-382.

[Giblin 92] Giblin D. "Defect Causal Analysis: A Report From The Field". *Proceedings of the ASQC Second International Conference on Software Quality.* October, 1992. pages 1 - 5.

[Gould 85] Gould, John D. and Clayton Lewis. "Designing for Usability: Key Principles and What Designers Think". *Communications of the ACM.* March, 1985. Vol. 28, No. 3. pages 300-311.

[Grady 94] Grady, Robert B. "Successfully Applying Software Metrics". *IEEE Computer*. September, 1994. Vol. 27. No. 9. pages 18-25.

[Grudin 91] Grudin, Jonathan. "Interactive Systems: Bridging the Gaps Between Developers and Users". *IEEE Computer.* April, 1991. pages 59-69.

[Heitmeyer 93] Heitmeyer, Constance L. and Bruce G. Labaw. "Consistency Checks for SCR-Style Requirements Specifications". Naval Research Laboratory Report. NRL/FR/5540--93-9586. December 31, 1993.

[Heitmeyer 95] Heitmeyer, Constance L., Bruce G. Labaw, and Daniel Kiskis. "Consistency Checking of SCR-Style Requirements Specifications". To appear in *Proceedings, International Symposium on Requirements Engineering.* York, England. March 26-27, 1995.

[Heninger 80] Heninger, Kathryn L. "Specifying Software Requirements for Complex Systems: New Techniques and Their Application". IEEE Transaction on Software Engineering. Vol. SE-6. No. 1. January, 1980. pages 2-13.

[Henry 91] Henry, Joel E., Susan L. Keenan, Michael A. Keenan, and Sallie Henry. "An Assessment Method for Small Organizations". *Proceedings of the 1st International Software Quality Assurance Conference.* October, 1991.

[Henry 92] Henry, Joel E. Personal communication. March, 1992.

[Henry 93] Henry, Joel E. *An Integrated Approach to Software Process Assessment.* PhD Dissertation. Department of Computer Science. Virginia Polytechnic Institute and State University. May, 1993.

[Henry 94] Henry, Joel E. Personal communication. 1994.

[Hix 93] Hix, Deborah and H. Rex Hartson. *Developing User Interfaces Ensuring Usability Through Product & Process.* John Wiley & Sons, Inc. New York. 1993.

[Hix 94] Hix, Deborah. Personal communication. December 16, 1994.

[Holtzblatt 93] Holtzblatt, K. and H. Beyer. "Contextual design: Integrating customer data into the design process." *Bridges Between Worlds, INTERCHI '93.* Tutorial. Notes 6. Editors: Ashlund, S., K. Mullet, A. Henderson, E. Hillnagel, and T. White)

[Humphrey 90] Humphrey, Watts S. *Managing the Software Process.* Addison-Wesley Publishing Company. Menlo Park, California. 1990.

[Jeffries 91] Jeffries, Robin, Miller, James R., Wharton, Cathleen, and Uyeda, Kathy M. User Interface Evaluation In The Real World: A Comparison of Four Techniques. *CHI '91 Proceedings*, pages 119-124.

[Jeffries 94] Jeffries, Robin. "Usability Problem Reports: Helping Evaluators Communicate Effectively with Developers". *Usability Inspection Methods.* Edited by Jakob Nielsen and Robert L. Mack. John Wiley & Sons, Inc. 1994. pages 273-294.

[Karat 92] Karat, Clare-Marie, Robert Campbell, and Tarra Fiegel. "Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation". *Human Factors in Computing Systems.* CHI '92 Conference Proceedings. Monterey, California. May, 1992. pages 397-404.

[Karat 93] Karat, Clare-Marie. "Usability Engineering in Dollars and Cents". *IEEE Software.* May, 1993. pages 88-89.

[Karat 94a] Karat, Clare-Marie. "A Comparison of User Interface Evaluation Methods". *Usability Inspection Methods.* Edited by Jakob Nielsen and Robert L. Mack. John Wiley & Sons, Inc. pages 203-235.

[Karat 94b] Karat, Clare-Marie. "A Business Case Approach to Usability". *Cost-Justifying Usability.* Editors: Bias, Randolph G., and Deborah J. Mayhew. Academic Press, Inc. Boston. 1994. pages 45-70.

[Keenan 94a] Keenan, Susan L., H. Rex Hartson, and Dennis G. Kafura. "Making the Most of Detected Usability Problems Through Usability Defect Analysis". To be published in the *Proceedings of 38th Annual Meeting of the Human Factors and Ergonomics Society.* Nashville, Tennessee. October 24-28, 1994.

[Keenan 94c] Keenan, Susan L. and Karen C. Bowen. "An Evaluation of the SCRtool A Tool for Specifying, Checking, Simulating, and Verifying Formal Requirements". Internal report to the Information Technology Division of the Naval Research Laboratory. February, 1995.

[Labaw 94] Labaw, Bruce. Personal communication. December , 1994.

[Labaw 95] Labaw, Bruce. Personal communication. January, 1995.

[Lamb 88] Lamb, David Alex. *Software Engineering: Planning for Change.* Prentice Hall. New Jersey. 1988.

[Macaulay 95] Macaulay, Linda. *Human-Computer Interaction for Software Designers.* International Thomason Computer Press. London. 1995.

[Mack 94] Mack, Robert and Frank Montaniz. "Observing, Predicting, and Analyzing Usability Problems". *Usability Inspection Methods.* Edited by Jakob Nielsen and Robert L. Mack. John Wiley & Sons, Inc. 1994. pages 295-339.

[Mantei 88] Mantei, Marilyn M. and Toby J. Teorey. "Cost/Benefit Analysis for Incorporating Human Factors in the Software Life Cycle". *Communications of the ACM.* Vol. 31. No. 4. April, 1988. pages 428-439.

[Mauro 94] Mauro, Charles L. "Cost-Justifying Usability in a Contractor Company". *Cost-Justifying Usability.* Editors: Bias, Randolph G., and Deborah J. Mayhew. Academic Press, Inc. Boston. 1994. pages 123-142.

[Mayhew 94] Mayhew, Deborah J. and Marilyn Mantei. "A Basic Framework for Cost-Justifying Usability". *Cost-Justifying Usability*. Editors: Bias, Randolph G., and Deborah J. Mayhew. Academic Press, Inc. Boston. 1994. pages 9-44.

[Meads 93] Meads, Jon. Personal communication. 1993.

[Miller 94] Miller, Ralph R. "The Interface Development Engineering Methodology". Internal Document. Computer Sciences Corporation. 1994.

[Mrazek 92] Mrazek, Deborah and Michael Rafeld. "Integrating Human Factors on a Large Scale: "Product Usability Champions"". *Human Factors in Computing Systems*. CHI '92 Conference Proceedings. Monterey, California. May, 1992. pages 565-570.

[Myers 79] Myers, Glenford J. *The Art of Software Testing*. John Wiley & Sons, New York. 1979.

[Nielsen 89] Nielsen, Jakob. "Usability Engineering at a Discount". *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*. Eds. G. Salvendy and M. J. Smith. Elsevier Science Publisher B.V. Amsterdam. 1989. pages 394-401.

[Nielsen 93a] Nielsen, Jakob. Personal communication. 1993.

[Nielsen 93b] Nielsen, Jakob. *Usability Engineering*. Academic Press, Inc. San Diego, California. 1993.

[Nielsen 94a] Nielsen, Jakob. "Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier". *Cost-Justifying Usability*. Edited by Randolph G. Bias and Deborah J. Mayhew. Academic Press, Inc. Boston. 1994. pages 245-272.

[Nielsen 94b] Nielsen, Jakob and Robert L. Mack (editors). *Usability Inspection Methods*. John Wiley & Sons, Inc. New York. 1994.

[Nielsen 90] Nielsen, Jakob and Molick, R. "Heuristic evaluation of user interfaces." *Proceedings ACM CHI'90 Conference*. Seattle, WA. pages 249-256.

[Norman 88] Norman, Donald A. *The Design of Everyday Things*. Currency Doubleday. New York. 1988.

[Ostrand 84] Ostrand, Thomas J. and Elaine J. Weyuker. Collecting and Categorizing Software Error Data in an Industrial Environment". *The Journal of Systems and Software*. Vol. 4. 1984. pages 289-300.

[Pfleeger 91] Pfleeger, Shari Lawrence. *Software Engineering The Production of Quality Software*. Macmillan Publishing Company. New York. 1991.

[Poltrock 89] Poltrock, Steven E. "Innovation in User Interface Development: Obstacles and Opportunities". *Human Factors in Computing Systems*. CHI '89 Conference Proceedings. pages 191-195.

[Potosnak 89] Potosnak, Kathleen. "Management: The key to success". *IEEE Software*. Vol. 6. No. 2. March, 1989. pages 86-88.

[Preece 94] Preece, Jenny, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, and Tom Carey. *Human-Computer Interaction.* Addison-Wesley Publishing Company. Reading, Massachusetts. 1994.

[Pressman 87] Pressman, Roger S. *Software Engineering A Practitioner's Approach Second Edition.* McGraw-Hill, Inc. New York. 1987.

[Rideout 91] Rideout, Tom. "Changing Your Methods From the Inside". *IEEE Software*. May, 1991. pages 99-100, 111.

[Rosson 96] Rosson, Mary Beth. Personal Communication. May, 1996.

[Schneiderman 87] Schneiderman, Ben. *Designing the User Interface Strategies for Effective Human-Computer Interaction.* Addison-Wesley. Reading, Massachusetts. 1987.

[Schulman 92] Schulman, Robert S. *Statistics in Plain English with Computer Applications.* Chapman & Hall. New York. 1992.

[Senders 91] Senders, John W., and Neville P. Moray (Analyzed and Synthesized by). *Human Error: Cause, Prediction, and Reduction.* Lawrence Erlbaum Associates. Hillsdale, New Jersey. 1991.

[Sommerville 89] Sommerville, Ian. *Software Engineering.* Addison-Wesley Publishing Company. New York. 1989.

[Thomas 95] Thomas, John C. "Usability Engineering in the Year 2020". To appear in *Advances in HCI.* Vol. 5.

[Vora 95] Vora, Pawan. Classifying user Errors in Human-Computer Interactive Tasks, *Common Ground, Usability Professional Association*, Vol. 5, No. 2 (May, 1995), page 15.

[Weiss 79] Weiss, David. M. "Evaluating Software Development by Error Analysis: The Data from the Architecture Research Facility". *The Journal of Systems and Software*. Vol. 1. 1979. pages 57-70.

[Whiteside 88] Whiteside, John, John Bennett, and Karen Holtzblatt. "Usability Engineering: Our Experience and Evolution". *Handbook of Human-Computer Interaction*. Edited by H. Helander. Elseview Science Publishers. 1988. pages 791-817.

[Wiklund 94] Wiklund, Michael E. "Introduction". *Usability in Practice How Companies Develop User-Friendly Products.* AP Professional. Boston. 1994.

APPENDIX A: UPT Categories Glossary

Glossary For Five Primary Categories

Usability Problems In The Artifact Component

An artifact usability problem is any difficulty encountered by the user when they view, read, or manipulate objects present in (or missing from) the user interface (buttons, scroll bars, data entry fields, icons, hypertext links, menus, menu items, title bars, windows, and dialogue boxes, on-screen text). Although these problems can impede a user's progress through the task, the focus is on difficulties with specific user interface objects, not movement through the task.

Visualness Usability Problems

Visualness usability problems are concerned with the way the user interface looks, *not* how well user tasks are mapped to the system. These problems occur when users have trouble with the

- visual aspect of screen layout (position, proximity, number of user interface objects, grouping related features, use of white space, consistent object placement),
- visual and audio aspect of object appearance (readable font size on button labels, pictures used on icons, visual aspect of colors and shape, including graying out an object, inconsistent appearance), and

- visual aspect of object movement.

This category also includes problems with the visual aspect of information provided in on-line help and tutorials, and results of user queries.

Language Usability Problems

Language usability problems occur when the user has trouble understanding the words that are used in the user interface. This may be due to the lack of user task domain terms (user-centered language), imprecise and inconcise words, and inconsistent use of words. These problems occur when users experience difficulties with

- words used as names on objects (such as buttons, title bars, field labels), and
- words used in phrases and sentences in system messages (feedback, error), on-screen text (on-screen instructions), on-line help, and tutorials.

Manipulation Usability Problems

A manipulation problem occurs when the user has trouble with some aspect of manipulating objects on the user interface. This type of problem occurs when the user has trouble

- recognizing, understanding, and interpreting visual cues,
- with missing or inconsistent visual cues,
- discovering when or how direct manipulation can be used, and

- that involves the user's ability to use the mouse and its buttons to directly manipulate objects (speed of cursor tracking, triple clicking, depressing multiple mouse buttons simultaneously).

Usability Problems In The Task Component

A task usability problem focuses on the user's movement through a task. They include difficulties with the way tasks are mapped to the system (task structure), available system functionality, and whether or not the system facilitates (eases) task completion.

Task-mapping Usability Problems

Task-mapping usability problems are concerned with how well user tasks are mapped to the system. This category includes problems that occur when relevant user tasks are not included in the mapping. These problems focus on the structure of the task, the number and sequence of sub tasks, the functionality to support the task, and the user's ability to navigate within the system. These problems occur when the user has trouble with

- the interaction (possibly due to the lack of user-centered task mapping, cognitively direct task mapping, mental models, metaphors, task structure, sequence of subtasks, number of steps to task completion),
- navigation (techniques include hypertext or hypergraphic links, buttons, windows, menu items),
- available or missing underlying system functionality (core or non-user interface functionality).

Task-facilitation Usability Problems

Task-facilitation usability problems have to do with how well the system assists task completion (eases task performance). Task-facilitation problems occur when the system does not help the user follow the task structure, use the task structure more efficiently, return to the task after diverging, nor access functionality in a way that is more suitable for him/her. These problems occur when the user has trouble with

- user preferences (e.g., customization of tool bar, title bar or window color),
- shortcuts (e.g., key sequences, shortcut icons, aliases, macro facilities),
- task/function automation (which tasks are performed by user, and which functions are performed by the system, includes locus of control),
- default values,
- keeping the user task on track (includes guiding the user to the next part of the task),
- user action reversal,
- error recovery, and
- error prevention.

APPENDIX B: Roles Forms and Glossary

Roles Form 1

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
Team Role	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
cognitive psychologist					
customer					
end user					
graphic designer					
human factors specialist					
market analyst					
marketer					
moderator (focus groups)					
problem domain expert					
quality assurance					
software specifier					
software designer					
software documentor					
software implementor					
software librarian					
software tester					
systems analyst					
technical writer					
user-interface interaction designer					
user-interface software designer					
user-interface software implementor					
user-interface evaluator					
user-interface documentor					

Roles Form 2

	ARTIFACT			TASK	
	USABILITY PROBLEMS			USABILITY PROBLEMS	
Team Role	Visualness	Language	Manipulation	Task-mapping	Task-facilitation
customer					
market analyst					
marketer					
problem domain expert					
quality assurance					
software specifier					
software designer					
software documentor					
software implementor					
software librarian					
software tester					
systems analyst					
technical writer					

Glossary For Roles Forms

Application (Problem) Domain Expert

Has in-depth knowledge in the application area (information processing, mathematics, simulation, word processing) in which the system is being built. .

Cognitive Psychologist

Skilled in extracting, representing, and reasoning about mental representations and processes involved in user tasks. Are familiar with theoretical memory and attention, perception, learning, and problem-solving. Also skilled in the collection and interpretation of verbal protocol data. Could have some familiarity with social and motivational influences in individual cognition.

Customer

Purchases the software. May or may not also use the software system.

End User

Users who utilizes the software to perform job activities. May or may not purchase the software.

Graphic Designer

Use skills to devise various aspects of the appearance of a user interface. This includes icons, desktop, dialogue boxes, and window borders.

Human Factors Specialist

Strong background in the "systems" approach to design and development. This approach includes activities like function allocation, task analysis, requirements analysis, and test and evaluation. The focus is on user-centered design, with strong emphasis in human physical and cognitive limitations and capabilities. Also has an understanding of controlled experimental designs and statistical analysis.

Market Analyst

Determines needs of user groups. *Performs user analysis.*

Marketer

Deals with end users during maintenance phase of software life cycle. Often is the intermediary between the users and the developers. May accumulate a list of usability problems during maintenance after product has shipped.

Moderator (Focus Groups)

Conducts group discussions to determine user-interface requirements. Conducts group discussions after the user interface has been implemented to acquire end-user feedback on level of usability achieved in the system.

Quality Assurance Personnel

Ensures that the system is not delivered unless it is of acceptable quality. Ensures that all process activities have been performed before product is shipped.

Software Specifier

Uses software requirements analysis to specify system behavior *(may include user-interface behavior).*

Software Designer

Generates a system-level description of what the system is to do. Can use software specifications to design system.

Software Documentor

Writes internal documentation for software systems.

Software Implementor

Implements (codes) software system using design document and specifications.

Software Librarian

Prepares and stores documents that are used during the life of the system (e.g., requirements specification, design descriptions, program documentation, training manuals, test schedules). Also can enters, compiles, link, and do preliminary testing of code written by other programmers.

Software Tester

Conducts various tests (module, integration, system, alpha, beta) to catch errors in the code that have been previously overlooked.

Systems Analyst

Analyzes an organization and identifies its processing and information requirements. Works with customers to build discrete requirements from user wants.

Technical Writer

Writes external documentation (user's guides/manuals) for software systems.

User-Interface Interaction Designer

Designs the interaction (dialogue) that the user has with the system.

User-Interface Software Designer

Responsible for the overall design of the software that supports the user interface. Translates the interaction design into an interface software design.

User-Interface Software Implementor

Implements (codes) the user-interface software design.

User-Interface Evaluator

Evaluates the level of usability achieved in the user interface. May use inspection methods (e.g., cognitive walkthroughs, heuristic evaluation) and/or user testing.

User-Interface Documentor

Documents the development history of the user interface from iteration to iteration.

APPENDIX C: Activities Form and Glossary

Activities Form

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
Systems Analysis	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
Contextual Inquiry, Job Analysis, Needs Analysis, Task Analysis, Work-Flow Analysis					
Focus Groups					
Functional Analysis (internal view of functions)					
Learn About Application Domain					
Learn About Existing/ Competing Systems					
Systems Analysis					
System Requirements/ Specifications					
Task/Function Allocation (which tasks performed by user, which functions performed by system)					
Usability Requirements/ Specifications					
User Analysis (user profiles, user class definitions)					
User Site Visits					

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
Guidelines	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
Use Commercial Style Guides (look and feel of individual user-interface objects or artifacts)					
Use Customized Style Guides (consistency)					
Use General Interface Guidelines (know the user...)					

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
Design Activity	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
High-fidelity (interactive) Prototyping (software-based prototypes)					
Low-fidelity (static) Prototyping (paper prototypes, screen mock-ups)					
Participatory Design					
Usage Scenarios					

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
Design Representation Techniques	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
Behavioral Design Representation (CLG, GOMS, TAG, UAN)					
Knowledge & Model Based (e.g. UIDE)					
Object Orientation					
State Transition Diagrams (including concurrent state diagrams)					
Usage Scenarios, Use Cases					

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
Inspection Methods	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
Cognitive Walkthrough (Jogthru)					
Consistency Inspections					
Feature Inspection					
GOMS Analysis					
Guideline Reviews, Heuristic Evaluation					
Pluralistic Walkthroughs					
Standards Inspections					

	ARTIFACT USABILITY PROBLEMS			TASK USABILITY PROBLEMS	
User Testing Activity	**Visualness**	**Language**	**Manipulation**	**Task-mapping**	**Task-facilitation**
Alpha Tests					
Beta Tests					
Co-discovery					
Critical Incident Taking (positive and negative incidents)					
Structured Interviews (gather user opinions)					
Testing User Performance (using benchmark tasks)					
User Preference (Satisfaction) Questionnaires					
Verbal Protocol Taking (thinking aloud studies, possibly retrospective)					

GLOSSARY FOR ACTIVITIES FORM

Systems Analysis Definitions

Contextual Inquiry, Job Analysis, Needs Analysis, Task Analysis, Work-Flow Analysis

Contextual Inquiry is performed on site. Developers talk with and observe real users performing real tasks. Needs Analysis is used to determine that a new system is needed. Task Analysis is a user-interface analysis activity in which the procedures users employ to perform work tasks are identified, categorized, and defined. Job Analysis and Work-Flow Analysis performed to determine how user tasks fit into the work environment.

Focus Groups

Discussion groups conducted to obtain user feedback on the concept and proposed interface. Focus groups rely on discussions and questionnaires.

Functional Analysis

Process of identifying the major activities of a system. Provides developers with an internal view of system functions.

Learn About Application Domain

Is performed to determine needed functionality and appropriate, user-centered language.

Learn About Existing/Competing Systems

Enables comparison of proposed system with competing systems.

Systems Analysis

Preliminary investigation during which the objective, constraints, and scope of the system are identified. Activity seeks to understand the system's existing environment, document its functionality, and determine the new system's requirements.

System Requirements/Specifications

System design requirements are written in a specific format (specification document) that states what the product must do.

Task/Function Allocation (performed by user or by system)

Analysis determines which tasks will be performed by the user (manual) and which functions will be performed by the system (automated).

Usability Requirements/Specifications

Establishes quantitative usability goals that are used as a guide for knowing when an interface is "good enough."

User Analysis (user profiles, user class definitions)

Determines user characteristics and produces a user profile(s).

User Site Visits

Developers visit user sites.

Guidelines Definitions

Use Commercial Style Guides

Helps with "look and feel" of the user interface and the objects (artifacts) on the interface. Commercial style guides typically produced by one organization or vendor, made commercially available, and provides a concrete and useful framework for design. Includes description of specific interaction style or object and guidance of when and how to use a particular interaction style or object.

Use Customized Style Guides

Helps maintain consistency throughout the user interface, possibly across products. Customized style guide is produced internally for a particular interface development project or set of projects.

Use General Interface Guidelines (know the user...)

Helps keep the focus on the user during specification and design. Examples: know the user, practice user-centered design, prevent user errors, keep locus of control with the user, be consistent, keep it simple, recognize rather than recall, and use informative feedback.

Design Activity Definitions

High-fidelity (interactive) Prototyping

A prototyping strategy in which the prototype is a software-based. Uses iterative refinement.

Low-fidelity (static) Prototyping

A prototyping strategy in which the prototype is constructed out of paper, includes screen mock-ups. Uses iterative refinement.

Participatory Design

End users are included in the design process (often design meetings) to provide insights.

Systems Design

Development activity that synthesizes the requirements identified during the analysis stage into a new system blueprint. Process of reassembling the components and functions identified during analysis.

Usage Scenarios

Usage scenarios (story line of users performing tasks on the system) can be used as a design technique and as a documentation technique used to record design history.

Design Representation Techniques Definitions

Behavioral Representation

Formal user-task oriented technique for specifying the behavior of the user interface. Examples:

Command Language Grammar (CLG)
Goals, Operators, Methods, And Selection (GOMS)
Keystroke Level Model

Task Action Grammar (TAG)

User Action Notation (UAN)

Knowledge & Model Based

Technique consists of objects, attributes, action, and pre-and post-conditions on actions that form a declarative description of an interface. From these descriptions, alternative interfaces can be generated for the same underlying functionality. Examples:

User Interface Development Environment (UIDE)

Object Orientation

A methodology that analyzes, designs, and constructs a system's object classes, methods, and attributes. Focuses on objects in the user interface and actions that can be performed on those objects.

State Transition Diagrams (including concurrent state diagrams)

Networks of user-interface states which identify all actions possible in that state and to which state each action leads. Represents control flow in asynchronous interaction using a set of graphical, state diagrams which represent the interface.

Usage Scenarios, Use Case

A usage scenario is a story line for each task to be performed on the system. A use case is an instantiation of the user in the system. Developers work step-by-step, in writing, through every scenario a system will be expected to perform, internally and externally.

Inspection Methods Definitions

Cognitive Walkthrough (jogthru)

Using an explicitly detailed procedure, simulates a user's problem-solving process at each step in the human-computer dialogue. Checks to see if the simulated user's goals and memory for actions can be assumed to lead to the next correct action.

Consistency Inspections

Designers representing multiple projects inspect an interface to see whether it does things in a way that is consistent with their own designs. Aimed at evaluating consistency across the family of products that has been evaluated by an inspection team.

Feature Inspection

Focus on the function delivered in a software system (whether the function, as designed, meets the needs of intended end users). Involves evaluation of a function and the design of that function. Focuses on the usefulness of interface function.

Formal Usability Inspections

Similar to code inspection methods. User interface is inspected individually by team members, then merge findings at a team meeting.

GOMS Analysis

For certain tasks, can model the goals the user had, the methods available in the system to satisfy these goals, and the operator sequences that are followed. Can be used to predict how long a user task will take.

Guideline Reviews, Heuristic Evaluation

Guideline reviews are inspections where an interface is checked for conformance with a comprehensive list of usability guidelines. Heuristic evaluation involves having usability specialists judge whether each dialogue element conforms to established usability principles. Often, the most important parts of the user interface are examined and compared to a list of heuristics.

Pluralistic Walkthroughs

Meetings where users, developers, and human factors people step through a scenario, discussing usability issues associated with dialogue elements involved in the scenario steps.

Standards Inspections

An expert of some interface standard inspects the interface for compliance. Aimed at increasing the degree to which a given interface is in the range of other systems on the market that follow the same standards.

User Testing Activity Definitions

Co-discovery

More than one user tests (uses) the system at the same time or an evaluator and a participant work to together to uncover usability problems. The users verbalize about what they are doing to each other.

Critical Incident Taking (both positive and negative incidents)

Both positive and negative critical incidents are recorded during user testing session. A critical incident is something that happens while a participant is working that has a significant effect.

Structured Interviews

Typically held after the testing session to gather users' opinions about a software system. Consists of preplanned questions that the evaluator asks each participant.

Testing Using Benchmark Tasks

Designed to gather objective measurements with respect to user performance (time to task completion, error rate). A test performed to ensure that system interfaces are easy to learn and user and that they support the desired level of user productivity. Typically conducted to determine where users are most prone to make errors, to evaluate user reactions, and to assess productivity.

User Preference (satisfaction) Questionnaires

User completes a questionnaire designed to capture subjective information about user satisfaction.

Verbal Protocol Taking (retrospective , thinking aloud studies)

Users' comments with respect to what happened during the session are recorded during or after (retrospective) a user testing session.

APPENDIX D: Means And Standard Deviations For Roles Form 1 And The Activities Form

Appendix D is divided into two sections. Section D.1 presents the means and standard deviations (SD) of responses that associate each role with UPT categories. The means and standard deviations of responses for the five primary categories are given in Tables D.1 through D.5. Tables D.6, D.7, and D.8 give the means and standard deviations for roles and the artifct component, the task component, and the overall UPT. Section D.2 presents the means and standard deviations (SD) of responses that associate each activity with UPT categories. The means and standard deviations of responses for the five primary categories are given in Tables D.9 through D.13. Tables D.14, D.15, and D.16 give the means and standard deviations for activities and the artifct component, the task component, and the overall UPT.

The interpretation of each pair of means and standard deviations is given in the column labeled "Intp." Recall that the interpretation is either a strong positive association (+), a weak positive association (w+), no association (blank), a weak negative association (w—), or a strong negative association (—).

D.1 Means And Standard Deviations For Roles Form 1

Table D.1. Means and standard deviations for each role for visualness.

Visualness and role	Mean	SD	Intp
cognitive psychologist	1.333	0.816	
customer	0.833	0.983	
end user	1.5	0.837	
graphic designer	2	0	+
human factors specialist	1.833	0.408	+
market analyst	0.833	0.753	
marketer	0.833	0.753	
moderator (focus group)	1	1	
problem domain expert	1.167	0.753	
quality assurance	0.667	0.816	
software specifier	0.667	0.816	
software designer	0.333	0.516	—
software documentor	0.167	0.408	—
software implementor	0.333	0.516	—
software librarian	0	0	—
software tester	0.5	0.837	
systems analyst	1.167	0.408	
technical writer	1.667	0.516	+
user-interface interaction designer	2	0	+
user interface software designer	1	1.095	
user interface software implementor	0.833	0.983	
user interface evaluator	1.833	0.408	+
user interface documentor	1.167	0.753	

Table D.2. Means and standard deviations for each role for language.

Language and role	Mean	SD	Intp
cognitive psychologist	1.5	0.548	+
customer	0.833	0.983	
end user	1.5	0.837	
graphic designer	0.833	0.408	
human factors specialist	1.833	0.408	+
market analyst	0.5	0.548	—
marketer	0.667	0.516	w—
moderator (focus group)	1	1	
problem domain expert	1.5	0.837	
quality assurance	0.5	0.837	
software specifier	0.667	0.816	
software designer	0.333	0.516	—
software documentor	0.5	0.548	—
software implementor	0.333	0.516	—
software librarian	0	0	—
software tester	0.5	0.837	
systems analyst	1.333	0.516	
technical writer	1.833	0.408	+
user interface interaction designer	2	0	+
user interface software designer	0.833	0.983	
user interface software implementor	0.833	0.983	
user interface evaluator	1.833	0.408	+
user interface documentor	1.167	0.753	

Table D.3. Means and standard deviations for each role for manipulation.

Manipulation and role	Mean	SD	Intp
cognitive psychologist	1.5	0.548	+
customer	0.833	0.983	
end user	1.5	0.837	
graphic designer	1.167	0.753	
human factors specialist	1.833	0.408	+
market analyst	0.333	0.516	—
marketer	0.167	0.408	—
moderator (focus group)	0.6	0.894	
problem domain expert	0.833	0.753	
quality assurance	0.333	0.816	
software specifier	0.667	0.816	
software designer	0.5	0.837	
software documentor	0.167	0.408	—
software implementor	0.5	0.837	
software librarian	0.167	0.408	—
software tester	0.5	0.837	
systems analyst	1.167	0.408	
technical writer	1.333	0.816	
user interface interaction designer	2	0	+
user interface software designer	1.167	0.983	
user interface software implementor	0.667	0.816	
user interface evaluator	1.833	0.408	+
user interface documentor	1	0.632	

Table D.4. Means and standard deviations for each role for task-mapping .

Task-mapping and role	Mean	SD	Intp
cognitive psychologist	1.5	0.837	
customer	1	0.894	
end user	1.667	0.516	+
graphic designer	1	0.894	
human factors specialist	2	0	+
market analyst	1.167	0.753	
marketer	0.667	0.516	w—
moderator (focus group)	1.2	0.837	
problem domain expert	1.667	0.516	+
quality assurance	0.333	0.516	—
software specifier	1.333	0.516	
software designer	0.667	0.516	w—
software documentor	0.333	0.516	—
software implementor	0.167	0.408	—
software librarian	0.167	0.408	—
software tester	0.5	0.548	—
systems analyst	2	0	+
technical writer	1.167	0.753	
user interface interaction designer	2	0	+
user interface software designer	1.167	0.983	
user interface software implementor	0.667	0.816	
user interface evaluator	1.833	0.408	+
user interface documentor	1	0.632	

Table D.5. Means and standard deviations for each role for task-facilitation .

Task-facilitation and role	Mean	SD	Intp
cognitive psychologist	1.5	0.837	
customer	0.833	0.983	
end user	1.5	0.837	
graphic designer	0.833	0.408	
human factors specialist	2	0	+
market analyst	0.667	0.516	w—
marketer	0.167	0.408	—
moderator (focus group)	0.8	0.837	
problem domain expert	1.5	0.548	+
quality assurance	0.667	0.816	
software specifier	1	0.632	
software designer	0.5	0.837	
software documentor	0.167	0.408	—
software implementor	0.333	0.816	
software librarian	0.167	0.408	—
software tester	0.5	0.548	—
systems analyst	1.667	0.516	+
technical writer	1.167	0.753	
user interface interaction designer	2	0	+
user interface software designer	1.167	0.983	
user interface software implementor	0.667	0.816	
user interface evaluator	1.667	0.516	+
user interface documentor	1	0.632	

Table D.6. Means and means of standard deviations for each role for the artifact component.

Artifact component and role	Mean	Mean SD	Intp
cognitive psychologist	1.444	0.637	w+
customer	0.833	0.983	
end user	1.5	0.837	
graphic designer	1.333	0.387	
human factors specialist	1.833	0.408	+
market analyst	0.556	0.606	w—
marketer	0.556	0.559	—
moderator (focus group)	0.867	0.965	
problem domain expert	1.167	0.781	
quality assurance	0.5	0.823	
software specifier	0.667	0.816	
software designer	0.389	0.623	w—
software documentor	0.278	0.455	—
software implementor	0.389	0.623	w—
software librarian	0.056	0.136	—
software tester	0.5	0.837	
systems analyst	1.222	0.444	
technical writer	1.611	0.58	+
user interface interaction designer	2	0	+
user interface software designer	1	1.021	
user interface software implementor	0.778	0.928	
user interface evaluator	1.833	0.408	+
user interface documentor	1.111	0.713	

Table D.7. Means and means of standard deviations for each role for the task component.

Task component and role	Mean	SD	Intp
cognitive psychologist	1.5	0.837	
customer	0.917	0.939	
end user	1.583	0.677	w+
graphic designer	0.917	0.651	
human factors specialist	2	0	+
market analyst	0.917	0.635	
marketer	0.417	0.462	—
moderator (focus group)	1	0.837	
problem domain expert	1.583	0.532	+
quality assurance	0.5	0.666	w—
software specifier	1.167	0.574	
software designer	0.583	0.677	w—
software documentor	0.25	0.462	—
software implementor	0.25	0.612	w—
software librarian	0.167	0.408	—
software tester	0.5	0.548	—
systems analyst	1.833	0.258	+
technical writer	1.167	0.753	
user interface interaction designer	2	0	+
user interface software designer	1.167	0.983	
user interface software implementor	0.667	0.816	
user interface evaluator	1.75	0.462	+
user interface documentor	1	0.632	

Table D.8. Means and means of standard deviations for each role for the UPT.

UPT and role	Mean	SD	Intp
cognitive psychologist	1.467	0.717	
customer	0.867	0.965	
end user	1.533	0.773	
graphic designer	1.167	0.493	
human factors specialist	1.9	0.245	+
market analyst	0.7	0.617	
marketer	0.5	0.52	—
moderator (focus group)	0.92	0.914	
problem domain expert	1.333	0.681	
quality assurance	0.5	0.761	
software specifier	0.867	0.72	
software designer	0.467	0.645	w—
software documentor	0.267	0.458	—
software implementor	0.333	0.619	w—
software librarian	0.1	0.245	—
software tester	0.5	0.721	
systems analyst	1.467	0.37	+
technical writer	1.433	0.649	w+
user interface interaction designer	2	0	+
user interface software designer	1.067	1.006	
user interface software implementor	0.733	0.883	
user interface evaluator	1.8	0.43	+
user interface documentor	1.067	0.681	

D.2 Means And Standard Deviations For The Activities Form

Table D.9. Means and standard deviations for each activity for visualness.

Visualness and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	0.867	0.712	
Focus Groups	1	0.632	
Functional Analysis	0.333	0.816	
Learn Application Domain	1	0.632	
Learn Competing Systems	1.333	0.516	
Systems Analysis	0.5	0.548	—
System Requirements & Specifications	0.6	0.894	
Task/Function Allocation	0.167	0.408	—
Usability Requirements & Specifications	1.5	0.548	+
User Analysis	1	0.632	
User Site Visits	1	0.632	
Guidelines (G)			
Use Commercial Style Guides	2	0	+
Use Customized Style Guides	2	0	+
Use General Interface Guidelines	1.667	0.516	+
Design Activities (D)			
High-fidelity Prototyping	2	0	+
Low-fidelity Prototyping	1.5	0.548	+
Participatory Design	1.333	0.816	
Usage Scenarios	0.833	0.753	
Design Representation Techniques(DRT)			
Behavioral Design Representation	0.333	0.516	—
Knowledge & Model Based	0.667	0.816	
Object Orientation	0.833	0.983	
State Transition Diagrams	0.167	0.408	—
Usage Scenarios, Use Cases	0.667	0.816	
Inspection Methods (I)			
Cognitive Walkthrough	1.5	0.837	
Consistency Inspections	1.667	0.516	+
Feature Inspection	0.333	0.516	—
GOMS Analysis	0.333	0.516	—
Guideline Reviews, Heuristic Evaluation	1.833	0.408	+
Pluralistic Walkthroughs	1.667	0.516	+
Standards Inspections	1.333	0.516	
User Testing Activities (UT)			
Alpha Tests	1.167	0.753	
Beta Tests	1.167	0.753	
Co-discovery	2	0	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1	0.632	
Testing User Performance	1.167	0.753	
User Preference	1	0.894	
Verbal Protocol Taking	1.833	0.408	+

Table D.10. Means and standard deviations for each activity for language.

Language and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	1.367	0.753	
Focus Groups	1.167	0.408	
Functional Analysis	0.333	0.816	
Learn Application Domain	1.667	0.516	+
Learn Competing Systems	1.333	0.516	
Systems Analysis	0.833	0.408	
System Requirements & Specifications	0.6	0.894	
Task/Function Allocation	0.167	0.408	
Usability Requirements & Specifications	1.333	0.516	
User Analysis	1.167	0.753	
User Site Visits	1.667	0.516	+
Guidelines (G)			
Use Commercial Style Guides	1.333	1.033	
Use Customized Style Guides	1.667	0.816	
Use General Interface Guidelines	1.5	0.548	+
Design Activities (D)			
High-fidelity Prototyping	1.833	0.408	+
Low-fidelity Prototyping	2	0	+
Participatory Design	1.833	0.408	+
Usage Scenarios	0.833	0.408	
Design Representation Techniques(DRT)			
Behavioral Design Representation	0.625	0.802	
Knowledge & Model Based	0.833	0.983	
Object Orientation	0.667	0.516	w—
State Transition Diagrams	0.333	0.516	—
Usage Scenarios, Use Cases	0.833	0.408	
Inspection Methods (I)			
Cognitive Walkthrough	1.667	0.816	
Consistency Inspections	1.667	0.516	+
Feature Inspection	0.333	0.516	—
GOMS Analysis	0.333	0.516	—
Guideline Reviews, Heuristic Evaluation	1.833	0.408	+
Pluralistic Walkthroughs	1.667	0.516	+
Standards Inspections	1.333	0.516	
User Testing Activities (UT)			
Alpha Tests	1.333	0.516	
Beta Tests	1.333	0.516	
Co-discovery	2	0	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.5	0.548	+
Testing User Performance	1	0.894	
User Preference	1	0.894	
Verbal Protocol Taking	1.833	0.408	+

Table D.11. Means and standard deviations for each activity for manipulation.

Manipulation and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	0.7	0.469	
Focus Groups	0.667	0.516	w—
Functional Analysis	0.333	0.516	—
Learn Application Domain	0.833	0.408	
Learn Competing Systems	1	0.894	
Systems Analysis	0.667	0.516	w—
System Requirements & Specifications	0.6	0.548	w—
Task/Function Allocation	0.667	1.033	
Usability Requirements & Specifications	1.667	0.516	+
User Analysis	0.833	0.753	
User Site Visits	0.667	0.516	w—
Guidelines (G)			
Use Commercial Style Guides	1.833	0.408	+
Use Customized Style Guides	2	0	+
Use General Interface Guidelines	1.5	0.548	+
Design Activities (D)			
High-fidelity Prototyping	2	0	+
Low-fidelity Prototyping	1.5	0.548	+
Participatory Design	1.167	0.753	
Usage Scenarios	0.667	0.516	
Design Representation Techniques(DRT)			
Behavioral Design Representation	1	0.894	
Knowledge & Model Based	0.833	0.753	
Object Orientation	0.833	0.408	
State Transition Diagrams	1	0.894	
Usage Scenarios, Use Cases	0.5	0.548	—
Inspection Methods (I)			
Cognitive Walkthrough	1.167	0.983	
Consistency Inspections	1.5	0.548	+
Feature Inspection	0.333	0.516	—
GOMS Analysis	1.167	0.983	
Guideline Reviews, Heuristic Evaluation	1.667	0.516	+
Pluralistic Walkthroughs	1.667	0.516	+
Standards Inspections	1.167	0.753	
User Testing Activities (UT)			
Alpha Tests	1	0.632	
Beta Tests	1	0.632	
Co-discovery	1.667	0.516	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.167	0.408	
Testing User Performance	1.5	0.837	
User Preference	1	0.894	
Verbal Protocol Taking	1.667	0.516	+

Table D.12. Means and standard deviations for each activity for task-mapping.

Task-mapping and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	1.833	0.408	+
Focus Groups	1.5	0.548	+
Functional Analysis	1	0.632	
Learn Application Domain	1.833	0.408	+
Learn Competing Systems	1.333	0.516	
Systems Analysis	1.333	0.516	
System Requirements & Specifications	1.6	0.548	+
Task/Function Allocation	1.667	0.516	+
Usability Requirements & Specifications	1.333	0.816	
User Analysis	1.333	0.516	
User Site Visits	1.667	0.516	+
Guidelines (G)			
Use Commercial Style Guides	0.5	0.837	
Use Customized Style Guides	1	0.894	
Use General Interface Guidelines	1.167	0.983	
Design Activities (D)			
High-fidelity Prototyping	2	0	+
Low-fidelity Prototyping	2	0	+
Participatory Design	1.833	0.408	+
Usage Scenarios	2	0	+
Design Representation Techniques(DRT)			
Behavioral Design Representation	1.208	0.749	
Knowledge & Model Based	1.167	0.753	
Object Orientation	1.333	0.816	
State Transition Diagrams	1	1.095	
Usage Scenarios, Use Cases	2	0	+
Inspection Methods (I)			
Cognitive Walkthrough	1.333	0.516	
Consistency Inspections	0.5	0.548	—
Feature Inspection	1.5	0.548	+
GOMS Analysis	1	0.894	
Guideline Reviews, Heuristic Evaluation	1	0.632	
Pluralistic Walkthroughs	1.667	0.516	+
Standards Inspections	0.5	0.548	—
User Testing Activities (UT)			
Alpha Tests	1.333	0.516	
Beta Tests	1.333	0.516	
Co-discovery	2	0	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.5	0.548	+
Testing User Performance	1.5	0.837	
User Preference	1.167	0.408	
Verbal Protocol Taking	1.667	0.516	+

Table D.13. Means and standard deviations for each activity for task-facilitation.

Task-facilitation and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	1.833	0.408	+
Focus Groups	1	0.632	
Functional Analysis	0.833	0.753	
Learn Application Domain	1.333	0.516	
Learn Competing Systems	1.167	0.753	
Systems Analysis	1	0.632	
System Requirements & Specifications	1.4	0.548	w+
Task/Function Allocation	1.5	0.548	+
Usability Requirements & Specifications	1.5	0.837	
User Analysis	1.167	0.753	
User Site Visits	1.167	0.753	
Guidelines (G)			
Use Commercial Style Guides	0.833	0.753	
Use Customized Style Guides	1.167	0.753	
Use General Interface Guidelines	1	0.894	
Design Activities (D)			
High-fidelity Prototyping	1.833	0.408	+
Low-fidelity Prototyping	1.833	0.408	+
Participatory Design	1.5	0.548	+
Usage Scenarios	1.333	0.516	
Design Representation Techniques(DRT)			
Behavioral Design Representation	1.333	0.54	
Knowledge & Model Based	1.167	0.753	
Object Orientation	0.833	0.753	
State Transition Diagrams	1.333	0.816	
Usage Scenarios, Use Cases	1.667	0.516	+
Inspection Methods (I)			
Cognitive Walkthrough	1.333	0.516	
Consistency Inspections	0.833	0.753	
Feature Inspection	1.167	0.408	
GOMS Analysis	0.833	0.753	
Guideline Reviews, Heuristic Evaluation	1.167	0.753	
Pluralistic Walkthroughs	1.5	0.548	+
Standards Inspections	0.5	0.548	—
User Testing Activities (UT)			
Alpha Tests	1	0.632	
Beta Tests	1	0.632	
Co-discovery	1.833	0.408	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.167	0.408	
Testing User Performance	1.5	0.548	+
User Preference	1	0.632	
Verbal Protocol Taking	1.667	0.516	+

Table D.14. Means and standard deviations for each activity for the artifact component.

Artifact component and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	0.978	0.645	
Focus Groups	0.944	0.519	
Functional Analysis	0.333	0.716	
Learn Application Domain	1.167	0.519	
Learn Competing Systems	1.222	0.642	
Systems Analysis	0.667	0.491	
System Requirements & Specifications	0.6	0.779	
Task/Function Allocation	0.333	0.616	—
Usability Requirements & Specifications	1.5	0.527	+
User Analysis	1	0.713	
User Site Visits	1.111	0.555	
Guidelines (G)			
Use Commercial Style Guides	1.722	0.48	+
Use Customized Style Guides	1.889	0.272	+
Use General Interface Guidelines	1.556	0.537	+
Design Activities (D)			
High-fidelity Prototyping	1.944	0.136	+
Low-fidelity Prototyping	1.667	0.365	+
Participatory Design	1.444	0.659	w+
Usage Scenarios	0.778	0.559	
Design Representation Techniques(DRT)			
Behavioral Design Representation	0.653	0.738	
Knowledge & Model Based	0.778	0.851	
Object Orientation	0.778	0.636	
State Transition Diagrams	0.5	0.606	w—
Usage Scenarios, Use Cases	0.667	0.591	w—
Inspection Methods (I)			
Cognitive Walkthrough	1.444	0.879	
Consistency Inspections	1.611	0.527	+
Feature Inspection	0.333	0.516	—
GOMS Analysis	0.611	0.672	w—
Guideline Reviews, Heuristic Evaluation	1.778	0.444	+
Pluralistic Walkthroughs	1.667	0.516	+
Standards Inspections	1.278	0.595	
User Testing Activities (UT)			
Alpha Tests	1.167	0.634	
Beta Tests	1.167	0.634	
Co-discovery	1.889	0.172	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.222	0.529	
Testing User Performance	1.222	0.828	
User Preference	1	0.894	
Verbal Protocol Taking	1.778	0.444	+

Table D.15. Means and standard deviations for each activity for the task component.

Task component and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	1.833	0.408	+
Focus Groups	1.25	0.59	
Functional Analysis	0.917	0.693	
Learn Application Domain	1.583	0.462	+
Learn Competing Systems	1.25	0.635	
Systems Analysis	1.167	0.574	
System Requirements & Specifications	1.5	0.548	+
Task/Function Allocation	1.583	0.532	+
Usability Requirements & Specifications	1.417	0.827	
User Analysis	1.25	0.635	
User Site Visits	1.417	0.635	w+
Guidelines (G)			
Use Commercial Style Guides	0.667	0.795	
Use Customized Style Guides	1.083	0.824	
Use General Interface Guidelines	1.083	0.939	
Design Activities (D)			
High-fidelity Prototyping	1.917	0.204	+
Low-fidelity Prototyping	1.917	0.204	+
Participatory Design	1.667	0.478	+
Usage Scenarios	1.667	0.258	+
Design Representation Techniques(DRT)			
Behavioral Design Representation	1.271	0.644	
Knowledge & Model Based	1.167	0.753	
Object Orientation	1.083	0.785	
State Transition Diagrams	1.167	0.956	
Usage Scenarios, Use Cases	1.833	0.258	+
Inspection Methods (I)			
Cognitive Walkthrough	1.333	0.516	
Consistency Inspections	0.667	0.65	w—
Feature Inspection	1.333	0.478	
GOMS Analysis	0.917	0.824	
Guideline Reviews, Heuristic Evaluation	1.083	0.693	
Pluralistic Walkthroughs	1.583	0.532	+
Standards Inspections	0.5	0.548	—
User Testing Activities (UT)			
Alpha Tests	1.167	0.574	
Beta Tests	1.167	0.574	
Co-discovery	1.917	0.204	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.333	0.478	
Testing User Performance	1.5	0.692	w+
User Preference	1.083	0.52	
Verbal Protocol Taking	1.667	0.516	+

Table D.16. Means and standard deviations for each activity for the overall UPT.

UPT and activity	Mean	SD	Intp
System Analysis (SA)			
Contextual Inquiry	1.32	0.55	
Focus Groups	1.067	0.547	
Functional Analysis	0.567	0.707	
Learn Application Domain	1.333	0.496	
Learn Competing Systems	1.233	0.639	
Systems Analysis	0.867	0.524	
System Requirements & Specifications	0.96	0.686	w+
Task/Function Allocation	0.833	0.583	
Usability Requirements & Specifications	1.467	0.647	
User Analysis	1.1	0.681	
User Site Visits	1.233	0.587	
Guidelines (G)			
Use Commercial Style Guides	1.3	0.606	
Use Customized Style Guides	1.567	0.493	+
Use General Interface Guidelines	1.367	0.698	
Design Activities (D)			
High-fidelity Prototyping	1.933	0.163	+
Low-fidelity Prototyping	1.767	0.301	+
Participatory Design	1.533	0.587	+
Usage Scenarios	1.133	0.439	
Design Representation Techniques(DRT)			
Behavioral Design Representation	0.9	0.7	
Knowledge & Model Based	0.933	0.812	
Object Orientation	0.9	0.695	
State Transition Diagrams	0.767	0.746	
Usage Scenarios, Use Cases	1.133	0.458	
Inspection Methods (I)			
Cognitive Walkthrough	1.4	0.734	
Consistency Inspections	1.233	0.576	
Feature Inspection	0.733	0.501	
GOMS Analysis	0.733	0.733	
Guideline Reviews, Heuristic Evaluation	1.5	0.544	+
Pluralistic Walkthroughs	1.633	0.523	+
Standards Inspections	0.967	0.576	
User Testing Activities (UT)			
Alpha Tests	1.167	0.61	
Beta Tests	1.167	0.61	
Co-discovery	1.9	0.185	+
Critical Incident Taking	1.833	0.408	+
Structured Interviews	1.267	0.509	
Testing User Performance	1.333	0.774	
User Preference	1.033	0.745	
Verbal Protocol Taking	1.733	0.473	+

VITA

Susan L. Keenan received her BA in mathematics from Christopher Newport University in May, 1983, and an MS in operations research from the College of William and Mary in May, 1985. She then accepted a teaching position at Dickinson College in Carlisle, Pennsylvania. During her four years at Dickinson, she attended Shippensburg University and earned an MS in computer science. At the completion of her masters degree in May of 1989, Susan decided to return to school as a full-time PhD student. She graduated from Virginia Tech in December, 1996, with a PhD in computer science.

During her stay at Virginia Tech, she received the Goff award for teaching excellence. Her dissertation research was funded, in part, by the Graduate Research Development Project at Virginia Tech. Supplementing her education, she worked as a usability-consultant for the Naval Research Laboratory in Washington, DC. She is a member of the ACM and SIGCHI.

In August of 1996, Susan will become a member of the computer science faculty at Columbus State University in Columbus, Georgia. There she will teach both undergraduate and graduate level courses, and supervise masters students' research projects.

Susan Lynn Keenan