Human-Computer Interaction -- INTERACT'03
M. Rauterberg et al. (Eds.)
Published by IOS Press, (c) IFIP, 2003, pp. 655-662

Classification of Usability Problems (CUP) Scheme

Ebba Thora Hvannberg[*], Lai-Chong Law[+]

[*]University of Iceland, Hjardarhaga 2-6, 107 Reykjavik, Iceland

ebba@hi.is

[+]Computer Engineering and Networks Laboratory, ETH Zürich, Gloriastr. 35, CH-8092 Zürich, Switzerland

law@tik.ee.ethz.ch

Abstract: Defect classification can improve product quality and motivate process improvement. Several defect classification schemes have been developed and used with good results in software development. This paper describes the design and test of a defect classification scheme that specifically extracts information from usability problems. We describe the attributes of the defects, the attribute values, including the phases of user interface development. We have tested the classification scheme on defects that were found during heuristic evaluation and user test of software that implements a broker for exchange of on line learning resources. The paper presents results of the classifications and assesses the repeatability of the defect classification scheme. We show how developers, project manager, and process engineers can use information from the defect classification for improvement of the user interface, personal skills and training, and process improvement.

Keywords: Usability, defect classification, heuristic evaluation, think-aloud, user test, process, product

1 Introduction

Defect tracking and analysis is an imperative factor for successful software quality assurance (Grady, 1992). Indeed, 'learning from error' is important for advancing knowledge in most, if not all, disciplines. Defects, being manifestations of underlying errors, carry much information that can be used to characterize the product quality, to track the project progress and control it, and to improve the process. Defect measurements consist of quantitative (frequency) and qualitative (characteristics) aspects, and can be effectively obtained with systematically and carefully designed defect classification schemes (**DCS**).

Fredericks (1999) provides a comprehensive review of DCS that have been developed since 1975. The goals for most of the schemes, released by large companies such as Hewlett Packard, IBM, and Sperry Univac, are to identify common causes for defects in order to determine corrective actions. Orthogonal Defect Classification (ODC) is one of the most popular DCS and has been widely used for software testing since its emergence in the early 1990s. ODC can overcome a number of drawbacks inherent in the traditional techniques, namely Statistical Defect Modelling and Root Causal Analysis. The former is too general to be useful for process improvement, whereas the latter is time consuming, costly and has low generalizability.

Although ODC and other highly elaborated DCS (e.g., IEEE 1993) aim to be used for the whole software development lifecycle, they can be customized for more specific applications. Indeed, several DCS have been developed specifically for investigating inspections by adapting the existing overarching schemes mentioned above (e.g., Kelly & Shepard, 2001; Porter et al., 1995). However, to our best knowledge, there is yet no DCS that is particularly developed for investigating defects identified in usability tests. Based on the conviction that defects identified from usability tests (i.e. a list of usability problems) will much benefit product quality and process improvement if precise and concrete information about these defects can be presented to the development team, we assume the challenge of developing such a scheme, which is

entitled **Classification of Usability Problems** (*CUP*) scheme.

2 Background

The challenge for designing DCS is to identify a smallest possible number of defect attributes in order to keep it simple and easy to use, and to minimize the overhead added to the development process, while completely mapping all activities of the development process (Bridge & Miller, 1998). Freimut (2001), based on his review of the literature on defect measurements, proposes a possible structure of DCS, which has inspired the development of CUP. Furthermore, existing DCS (Card, 1998; Chillarege, 1992; Grady, 1992; Laszek et al., 2000; Mays et al., 1990) provide us reference frameworks. Clearly, we need to adjust the granularity of existing attributes and their subsuming values and to create new ones to meet our specific needs (see Section 4). Specifically, in classifying usability problems, raters or reviewers in principle have no access or do not refer to the source code of the user interface. Hence, a number of attributes (e.g., triggers of code inspection in ODC) are not relevant for usability problems.

The repeatability of a DCS is very important. Since the classification of defects is a subjective exercise, it is plausible that different raters classify the same defect in a different category. The credibility of defect data is dubious if such disagreement in classifications is prevalent (El Emam & Wieczorek, 1998). To estimate the inter-rater reliability, we use kappa statistics (Section 9).

To distinguish a set of closely related terms we employed in our work, which are not uniformly defined in the software engineering literature, we present their definitions according to IEEE standard 729. A **fault** occurs when a human error results in a mistake in some software product, i.e., the fault is the encoding of the human error. A **failure** is the departure of a system from its required behaviour. Failures can be discovered both before and after system delivery. **Defects** refer collectively to faults and failures. Here we equate defects with **problems** identified in the context of usability test.

3 Usability Tests on the UBP

3.1 Product Description

The user interface of which defects are analysed in the present study is developed in the context of UNIVERSAL (http://www.ist-universal.org) - a project with the primary goal to demonstrate an open exchange of online learning resources (LR) among European higher education institutions. The technology that enables cataloguing, offering, enquiries, booking and accessing a variety of LR, ranging from a short video to a complete live course, is known as UNIVERSAL Brokerage Platform (**UBP**).

3.2 Method and Procedure

***Heuristic Evaluation (HE)*:** The two usability specialists performed HE on the experimental version of the UBP, inspecting the system independently. The heuristics used were taken from a set prepared by Molich and Nielsen (1990), supplemented by three of Shneiderman's (1998).

User Test (UT): Ten participants from two different institutes were recruited, including university professors, project managers, research assistants, and administrators. Each test participant was required to perform a set of 12 (out of the total 20) task scenarios covering the core functionalities of the UBP (e.g., providing a learning resource) and to think aloud while carrying them out. All test sessions were videotaped and analysed.

3.3 Measurements

***Heuristic Evaluation*:**

- *Quantitative measure:* The number of usability problems (UPs) falling in each of the two severity levels: major vs. minor.
- *Qualitative measure:* Detailed descriptions of individual UPs and their locations (i.e. scope)

User Test:

- *Performance measures*, which were obtained primarily through observations, include time-on-task, number and types of errors, number and types of help sought, and frequency of expressed frustration.
- *Subjective measures*, which were obtained mainly through participants' self-reporting, include post-test questionnaires and thinking aloud protocols

3.4 General Results

The number of unique usability problems identified by both evaluation methods is 63. The mean effectiveness and mean efficiency of the twenty tasks over the ten participants are 75% and 48%/minutes, respectively. The mean error rate per participant over the twelve tasks performed is 11.8 (SD=6.56). More detailed results and a comparison of the effectivenes of the two evaluation methods, heuristic evaluation and usability testing can be found in (Law and Hvannberg, 2002).

4 Classification of Usability Problems (CUP) Scheme

4.1 Goal

When designing a defect classification scheme to suit the needs of a project or an organization, we follow the procedures recommended by IEEE (IEEE 1994). The initial step is to define a clear goal that serves the needs of the organization. Most often, these needs are directed towards improving the quality and/or the economy of the software development. For those researchers studying defect classification methods, the goal may be to measure and benchmark attributes of different methods. In our case, the primary goal is:

To classify usability defects further in order to give the developers/designers better feedback on how to correct the defects.

4.2 Attributes and Attribute Values

To attain the above-mentioned goal, we have selected 10 attributes. Subsequently, for each attribute, we elaborate the rationale for choosing it and a set of possible attribute values it subsumes:

Identifier (ID): For reference's sake, each usability problem (UP) is assigned an ID. It facilitates communication between different evaluators.

Description: A concise description of the UP is necessary for the purpose of recognition and communication. A full description can be stored elsewhere such as in a usability report.

Defect Removal Activity: This attribute is derived from ODC. However, in our case it is confined to usability testing method, i.e., user test or heuristic evaluation.

Trigger: This attribute is derived from ODC. It describes what a test participant or a usability engineer is doing when he or she discovers the UP. In case of user test, triggers are task scenarios that test participants are asked to perform on a user interface. In case of heuristic evaluation, triggers are heuristics that a usability engineer applies to examine the interface, and it is necessary to register where in the interface the UP is found.

Impact: Collecting measurements pertinent to the impact of a UP can help developers decide how serious the UP is. This can influence how much effort developers should use in correcting the fault or how much they have to change the user interface. For heuristic evaluation, the attribute values of impact are: minor, moderate, and severe. Severe usability problems are those that prevent the user from completing a task or result in catastrophic loss of data or time. Moderate usability problems are those that significantly hinder task completion but for which the user can find a work-around. Minor usability problems are those that are irritating to the user but do not significantly hinder task completion. For user test, the attribute values of impact are: effectiveness, efficiency, frustration expressed and help sought, either on-line or at a help-desk. These values are computed for each task.

Expected phase: This attribute indicates in which of the eight phases of software development lifecycle (see Section 5) we expect the developer to be able to correct the faults that lead to the failure. In the case of user test, the expected phase does not have to be specified right after the test but before feedback is given to developers.

Failure qualifier: A usability engineer registers this attribute to capture more meaning about the UP. In the context of user test, the qualifier further explains whether the test participant (i.e. the user) experiences the UP as something being Missing (M), Incongruent mental model (I), Extraneous (E), Wrong (W), or whether there could be a Better way (B) for doing a task or the test participant Overlooked (O) something. A usability engineer records the value based on the verbal protocol of the test participant. In the context of heuristic evaluation, the usability engineer records his or her experience.

A UP is marked as missing (M) when the test participant fails to find something that is supposed to be present. A UP is marked as incongruent (I) when the user interface is unclear, because it does not match the test participant's mental model or her previous experience. In some cases, the test participant can notice that something has gone wrong (e.g., an apparent programming bug). If the test participant is experienced with different user interfaces, she can tell that something in the user interface could have been done in a better way (B). Sometimes, the test participant is given a task but she overlooks (O) an entity in the user interface, i.e. the user does not see the existing entity or fails to realize that she is supposed to interact with it.

Cause: With the aim to prevent future errors by improving the development processes, it is useful to understand why developers have committed the error. Put differently, what caused the fault that lead to the failure? A process engineer, a peer developer or a project manager can record this attribute after the fault has been found. We have selected five attribute values for the causes: Personal, Technical, Methodological, Managerial and Review. The category 'Personal' describes errors that can be

attributed to developers, who lack relevant knowledge and skills, experience or self-discipline. Errors in the category 'Technical' are due to transcription errors, lack of documentation or defective development environment. The category 'Methodological' includes errors that can be attributed to inadequate development methods used in the user interface development. Types of causes include inadequate use of models, prototypes, user interviews, user testing, etc. The category 'Managerial' refers to the errors that people make when working under great time pressure, especially if human resources are scarce or the requisite information is not well communicated. Errors of the category 'Review' occur when the task that the test participant is asked to perform does not align with his context or role or when the test participant is asked to perform some sequence of activities that are not included in the functional requirements of the system. These types of UPs can be called non-defects. The cause attribute can guide quality engineers when they determine what to improve or whether the personnel need more training.

Actual Phase: This attribute is to be collected after the fault has been fixed. The main reason for collecting it is to see whether evaluators are right in predicting the phase where the fault leading to UP actually lies (cf. Expected Phase). In addition to the eight phases (Section 5), N/A (not applicable) is a possible value when developers do not fix the UP identified, because they are not convinced about the necessity or urgency of fixing it. This attribute can also give developers some ideas which phases carry most faults.

Error Prevention: In the final step of UP analysis, ideas are to be collected on what can be done in the future to prevent the fault. No categories of attribute values are designed for this attribute. What we have obtained for this attribute are various guidelines for error prevention. When more data of this sort are systematically collected, categories representing different types of attribute values can be developed. To qualify this attribute, reviewers can examine the cause and see how the same problem can be prevented from happening again.

4.3 Application

Collecting the defect classification data is necessary but insufficient to meet the goal stated in section 4.1. We need to determine how to report the data and more important, we have to decide how to use them effectively. To give the developers better guidance on how they can correct the UP, we can give them a list of UPs with the attribute values on the removal activity, trigger, impact, failure qualifier and expected phase. There is little need to aggregate the data except over all defects of a single task or groups of tasks. The project manager can use this analysis to decide what tasks are most important to focus on. The attribute impact can also be used to prioritise the sequence of the defect removal. Before the defect classification begins, raters have to be trained. Training should include practice sessions so that we can verify that the attributes and attribute values are understood correctly.

5 User Interface Development Processes

In this section, we set forth a simple user interface lifecycle model. The standard, ISO 13407 Human Centred Design processes for interactive systems (ISO 1999), has four main activities. Our definition gives a more detailed list of processes to make it easier for the developer to relate to deliverables. We describe the phases by listing activities and deliverables. Deliverables are the outputs of the corresponding phases and can be prepared according to a defined template. Note that the last two development phases, namely, "Evaluation" and "Feedback" are **not** included as attribute values for the attribute "Expected Phase" and "Actual Phase" described in Section 4.2. The processes are not necessarily carried out in the order described below. The lifecycle can be iterative, cyclic or star-like with evaluation placed in the centre of the star.

Task analysis and context of use:

Activities: In task analysis we analyse the domain of the application and what cognitive tasks and physical actions a user performs in order to achieve a goal.

Deliverables: UML (Pooley and Stevens 1999) Use case diagrams and activity diagrams of tasks and/or hierarchical tasks. Scenarios. Descriptions of actors and environment.

Functional requirements:

Activities: Select which of the tasks from the task analysis phase that the system should perform. Translate tasks into functions. Determine the major input and output data to functions and pre- and post-conditions of functions.

Deliverables: Use case diagrams, Text description of use cases according to a template. Included in a requirement specification according to IEEE 830.1998 (IEEE 1998).

Quality attribute analysis:
Activities: Analysis of non-functional requirements such as usability, security, performance/efficiency, reliability/fault tolerance, interoperability between services or between devices, safety-criticality, environment sensitivity and people characteristics.
Deliverables: Requirement specification according to IEEE 830-1998.
Conceptual modelling:
Activities: Analysis of the system as a set of ideas, what the system should do, how it should behave and what the main objects in the system are and its operations.
Deliverables: list of concepts in a data dictionary, simple relationships between concepts, actions on concepts, content model or class diagram. Scenarios, mock-ups, low-fidelity prototyping.
Dialogue design:
Activities: Divide the responsibilities and design the dialogue between man and machine. Decide how the user translates goals or intentions into actions on objects. Decide how the system responds in terms of what objects and actions are displayed as a response to an action.
Deliverables: Essential use cases (Constantine and Lockwood 1999), context maps, sequence diagrams.
Navigational design:
Activities: Divide the user interface objects and actions into groups, i.e., contexts. Determine navigations between contexts.
Deliverables: User interface architecture, navigational model.
Presentation design:
Activities: Design the look of the user interface, i.e. layout of objects within a context, presentation of objects and actions as user interface elements, i.e. icons, buttons, layout, font, colour scheme, etc.
Deliverables: Graphic design of user interface objects.
Implementation:
Activities: Programming
Deliverables: programs and/or prototypes of user interface.
Evaluation:
Activities: Designing and performing user and expert tests. Validation. Reviews.
Deliverables: Test case design and results from testing and reviews.
Feedback:
Activities: Analyse and categorize the results of the evaluation. Analyse impact or risk of change caused by defect removal. Analyse cost of correcting errors.
Deliverables: A report on the cause of the usability problems, impact and priorities.

6 Examples of Measurements

The user test and heuristic evaluation of the UBP identified 39 and 52 defects, respectively. Two usability specialists analysed the defects independently using the CUP scheme. Both of them had been involved in the usability tests but not the development of the user interfaces. The deliverables available during the defect analysis were the software and several documents, including Functional Specifications, Developers' Bug Fixing Report, and Usability Test Report. Table 1 and 2 show how two defects have been classified.

Attribute	Attribute value
Defect id	1.1
Short description	Do not know how to log on
Trigger	Task 1: Login to the system
Effectiveness of task	90%
Efficiency of task	69%/minutes
Frustration of task	9
Help sought during task	0
Failure qualifier	I(ncongruent mental model)
Expected phase	Navigation (Nav)
Actual phase	Navigation (Nav)
Type of fault removed	Navigational design revised
Cause	Methodological (III)
Technique to prevent future errors	Design navigational map

Table 1: A defect found during User test

Recall that the attributes 'Type of fault removed' and 'Technique to remove future errors' are not categorized. Other attributes and their attribute values have been coded as we have described previously in sections 4 and 5.

Given the clear structure, classifications of the attributes according to the CUP scheme were perceived to be of moderate difficulty.

Attribute	Attribute value
Defect id	a10
Short description	Unconventional design of the menu system
Trigger	Simple and natural dialogue: Heuristic
Severity	Moderate
Failure qualifier	I(ncongruental)
Expected phase	Presentation
Actual phase	Presentation
Type of fault removed	Colours, font size and formats were modified
Cause	Methodological (III)
Technique to prevent future errors	Design user interface consistently with previous applications

Table 2: A defect found during HE

What the usability specialists were most unsure of was the attribute "Cause" (Section 9). One of the specialists, who had not followed the development of the product so closely, also reported unconfident of classifying the attribute "Actual Phase of the Fault".

7 Results for Personal, Product and Process Improvement

Our primary goal is to give the developers constructive feedback on usability testing for improving the product. By simply looking at the more detailed classifications of the defects, as shown in Section 6, the developers can focus right away on defect removal. Figure 1 shows the distribution of the failure qualifier of defects found during heuristics evaluation for each of the two raters (R1, R2). The aggregated values so displayed can tell developers about the overall experience of the test participants.

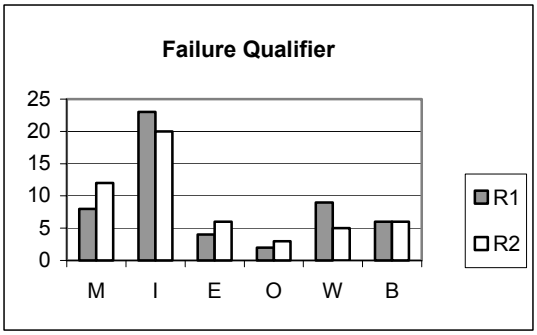

Figure 1: Distribution of the failure qualifier in HE

The developers can also fill in the attributes 'Actual Phase' and 'Type of Fault Removed'. They can look at those defects that were attributed to personal qualities and thus try to get better training, e.g., user interface design. The project manager can examine the attribute values of 'Impact' to prioritize defect removal. The project manager can also look at the source of the defects (Figure 2) to see in what development phases most of the defects lie.

The project manager can base on these results for planning short-term project. He can examine the phases where most of the errors occur and plan to improve the management of human resources, milestones or technical environment if those are the causes for the fault. A project manager may or may not be able to introduce a new method to the development, like drawing a navigational map or starting to use other types of tests, since that usually calls for training of developers and may take longer, respectively.

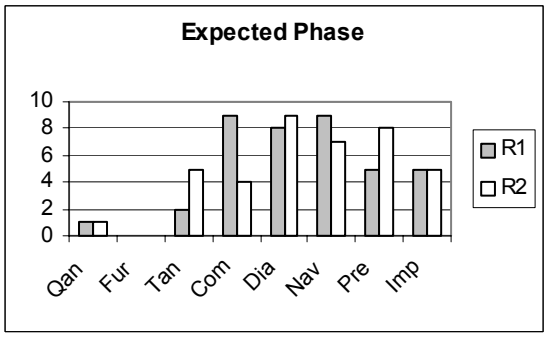

Figure 2: Expected phase of defects from UT

A quality or process engineer can investigate the results and learn what methods can be introduced to prevent errors recurring in the future. To see what phases have the most expensive faults, in terms of impact on users, may help the process engineer and the project manager to determine where to put the

most process and product improvement efforts, respectively.

8 Analysing Testing Methods

The results of CUP can be used to compare usability evaluation methods. This can provide the relevant data to the project manager and the process engineer who have to decide the extension of testing. Table 3 shows what types of faults were identified through the two different methods.

	HE	UT
Qan	0	2,6%
Fur	3,8%	0
Tan	9,6%	5,1%
Com	13,5%	23,1%
Dia	30,8%	20,5%
Nav	9,6	23,1%
Pre	30,8%	12,8%
Imp	1,9%	12,8%

Table 3: Comparison of expected phase of defects from HE and User test from rater R1

The defects found during heuristic evaluation are already classified into groups. That makes it easier to predict the Expected Phase. On the down side, it is more difficult to measure the impact since severity tends to be subjective, as illustrated in our measurements (Section 9).

9 Repeatability of CUP

For nominal scale data classified by two raters, kappa coefficient (Cohen, 1960) is the recommended statistical method for verifying inter-rater agreement. For other types of data and larger numbers of raters, different statistics are preferred. By analysing the kappa statistics calculated from the first iteration of our ratings, we could locate where our disagreements lied and thus improve the CUP scheme.

In case of 'Failure Qualifier', there were a number of cases when R1 chose "Changed" (i.e. alteration of user's mental model), R2 chose "Unclear", or vice versa. Discussion of these two problematic rating categories led to blending them into one category "Incongruent mental model", based on the assumption that an object appears unclear to a user if it is inconsistent with his or her mental model. Revision of the ratings followed suit, and the kappa coefficient increased from 0.0475 to 0.7457 (Table 4). Similar procedures were done for other attributes, resulting in improved agreements. Ranges of kappa can be qualified from "poor" to "very good" (Altman, 1991).

Attribute	No. of category	Kappa	Agreed cases*	Level of agree-ment
Failure Qualifier	6	0.7457	33	good
Expected Phase	8	0.5365	24	moderate
Actual Phase	9	0.6321	27	good
Cause	5	0.0363	22	poor

*Note the total number of cases is 39

Table 4: Kappa for the *User Test* Attributes

Attri-bute	No. of cate-gory	Kappa	Agreed cases*	Level of agree-ment
Failure Qualifier	6	0.5379	34	moderate
Severity	3	0.3309	29#	fair
Expected Phase	8	0.4717	31	moderate
Actual Phase	9	0.4388	28	moderate
Cause	5	-0.0606	14	poor

Note: * the total number of cases is 52; # weighted kappa was computed because 'severity' data are of ordinal scale.

Table 5: Kappa for the *Heuristic Evaluation* Attributes

In general, the inter-rater agreements for the attributes in both user test and heuristic evaluation are adequate, with the former being better than the latter. Nonetheless, what disappointing us is the kappa for the attribute 'Cause'. The negative value implies that the two raters agreed less than would be expected just by chance. In fact, the attribute values

of 'Cause' are derived from the previous literature. Inherent weaknesses of the existing schemes can be identified. For instance, Mays et al (1990) focus exclusively on developers; Card's (1998) scheme is too vague, whereas Leszak's is too encompassing. Definitely, we need to further refine this attribute.

10 Conclusion

To develop a classification scheme that is both complete and repeatable is a great challenge. CUP is complete for our goal of tracking and analysing usability problems, but further refinement of the scheme is necessary, especially in defining the causes underlying usability problems. For any kind of defect classification scheme, be it of large scale as ODC or smaller one as CUP, it is extremely difficult, if not impossible, to factor out the subjectivity of human raters. Reliance on the expertise and related experiences of individual raters is unavoidable. Hence, training and practice are required to ensure common understanding of a scheme and thus to improve the repeatability. There is a definite need for a defect classification scheme that is specifically designed for analysing usability problems. Our CUP scheme can fill the gap in the existing literature. It is not expensive and easy to apply. Furthermore, defining attribute values entails several iterations, as exemplified by our meticulous exercises in revising CUP based on kappa statistics. Clearly, CUP is not yet perfect, but it has the potential to be a useful tool for enhancing software quality and analysing research results on usability evaluation methods.

Acknowledgements

This work was supported by the Universal project and is partly sponsored by the European Commission (IST-1999-11747). The authors would like to extend special thanks to Professor Oddur Benediktsson for a review of an earlier version of the paper.

References

Altman, D.G., *Practical statistics for medical research*, Chapman & Hall, 1991.

Bridge, N., & Miller, C., Orthogonal Defect Classification: Using defect data to improve software development. *Software Quality, 3*, 1997-8.

Card, D., Learning from our mistakes with Defect Cause Analysis, *IEEE Software*, Jan/Feb 1998.

Chillarege, R, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray and M. Wong, Orthogonal Defect Classification- A Concept for In-Process Measurements, *IEEE Transactions on Software Engineering, Vol. 18, No. 11*, November 1992

Cohen, J., A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20, 37-46,* 1960.

Constantine, L. and L. Lockwood, *Software for Use*, Addison Wesley, 1999.

El Emam, K., & Wieczorek, I., The repeatability of code defect classifications. In: *Proceedings of the Ninth ISSRE,* Paderborn, Germany, Nov. 4-7, 1998.

Fredericks, M. Using defect tracking and analysis to improve software quality, 1999. http://citeseer.nj.nec.com/fredericks99using.html

Freimut, B., Developing and using defect classification schemes. *Fraunhofer IESE-Report No. 072.01/E Version 1.0,* September 1, 2001.

Grady, R., *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, 1992

IEEE Std 729-1983 Glossary of Software Engineering Terminologies

IEEE Std. 1044-1993, Standard Classification for Software Anomalies

IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications

ISO 13407:1999 Human-centred design processes for interactive systems

Kelly, D., & Shepard, T., A case study in the use of defect classification in inspections. Paper presented in *CASCON 2001*, Canada.

Law, E. Lai-Chong and Hvannberg, E., Complementarity and Convergence of Heuristic Evaluation and Usability Test: A Case Study of UNIVERSAL Brokerage Platform, *in NordiCHI 2002, Proceedings of the Second Nordic Conference on Human-Computer Interaction*, October 19-23, 2002, ACM ISBN: 1-1-58113-616-1

Leszak, M., Perry, D.E., Stoll, D., A case study in root cause defect analysis. *In: Proceedings of ICSE 2000*, Limerick, Ireland.

Mays, R.G., Jones, C.L., Holloway, G.J., & Studinski, D.P., Experiences with defect prevention. *IBM Systems Journal, 29(1)*, 4-32, 1990.

Molich, R., & Nielsen, J., Improving a human-computer dialogue. *Communications of the ACM, 33*(3), 338-348, 1990.

Pooley, R. and P. Stevens. *Using UML, Software Engineering with Object and Components*, Addison Wesley, 1999.

Porter, A., Votta, L.G., Basili, V.R., Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on Software Engineering*, Vol. 21, No. 6, 563-575, 1995.

Shneiderman, B., *Designing the user interface: Strategies for effective human-computer interaction (3rd Ed.)*. Reading: MA: Addison-Wesley, 1998