
Overview of the EC Wise Heuristic Evaluation Methodology

EC Wise, Inc.

info@ecwise.com

© 2001, EC Wise, Inc., All rights reserved

Table of Contents

Introduction..... 2

Sources of Usability Errors..... 2

User Interface Development Issues..... 3

Evaluation Method 3

Appendix A: ISO 9241 Ergonomic requirements for office work with visual display terminals. Part 10 - Dialogue Principles 4

Task adequacy 4

Self-descriptiveness 4

Controllability..... 4

Conformity to user expectations 4

Error tolerance 4

Flexibility 5

Learnability..... 5

Introduction

The EC Wise heuristic evaluation methodology is based on an analysis of the source of usability errors and how to prevent them. It conforms to international standard ISO 9241, *Ergonomic requirements for office work with visual display terminals (Part 10 - Dialogue Principles)*. In performing a heuristic evaluation, we consider the users for whom the user interface is intended, and the tasks that the designer intends them to perform. We usually perform a heuristic evaluation after at least a candidate version of product or service has been developed. Our objective is to identify potential problems that target users may have performing the intended tasks and to recommend alternative design approaches that would mitigate those problems.

The next (second) section of this paper provides an analysis of how much the three aspects of a user interface – the information model, the idiom model, and the presentation model – contribute to the creation of usability problems. This knowledge allows us to focus our evaluation on those parts of the user interface that contribute most to the usability of a system. The following (third) section of this paper discusses those user interface development issues that impact creation of the information model, the idiom model and the presentation model. In this section, we assert that an effective user interface is one that is user centered and task mapped. The fourth and final section explains how we evaluate a user interface to determine how effectively user centered and task mapped it really is. In this context, we describe how we apply the ISO 9241 standards.

Sources of Usability Errors

A user interface can be divided into 3 parts – the information model, the idiom model, and the presentation. The information model is a data model of the information the user needs to perform his tasks. It shows how data and functionality is grouped, and how those groups relate to each other. The information model is the first aspect of the user interface to get designed, although it may be produced implicitly as part of a user interface design. Implicit production of the information model can be a problem, because producing an accurate information model is the most critical step in designing a user interface. Problems with the information model account for 60% of the usability errors commonly found during testing.

The idiom model is created next, after the information model. It embodies the mapping between the actions the user must make to accomplish domain operations (such as finding an appropriate investment for a 401K portfolio) and the actions the user must make with the computer (selecting values from a series of controls that allow the user to set investment criteria). Inconsistencies and mismatches in the idiom model account for almost a third of usability errors.

The presentation model is the part of the user interface that is visible to the user. Once the information and idiom models exist, a designer can create a presentation model. An effective presentation model communicates the organization of domain information and relationships within that information (the information model) to the user, and elucidates how the user can perform operations on the information model (the idiom model). Note that, by the time that you get around to working on this visible part of the user interface, approximately 90% of the user interface (embodied in the information and idiom models) has already been designed. Most common user interface problems result from the propensity to jump into designing screens before taking time to determine and validate the information and idiom models.

It must be pointed out that, whether or not the user interface designer realizes it, they **do** design these models – they exist in the final user interface whether the designer recognizes it or not. The only choice the user interface designer has is whether to produce these models explicitly, based on an analysis of user needs, or to imply them in a user interface designed screen-first.

Figure 1 – Sources of Usability Errors	
Parts of a User Interface	Percentage of Errors
Presentation “.. what you see on the screen ...”	10%
Idiom Model “... does it use standard interaction mechanisms?”	30%
Information Model “... the way each part (object) works and how all the parts are put together.”	60%
After Roberts, D., Berry, D., Isensee, S., and Mullaly, J. (1997) <i>Developing Software Using OVID</i> IEEE Software 14(4) pp 51-57.	

User Interface Development Issues

The user of a computing system is actually performing 2 tasks at once. She first has to determine the next step in her work process, and then she must determine how to make the computer perform that step. The ease with which the user can perform a sequence of steps to accomplish a task determines the usability of the interface. The user interface must be both user centered and task mapped if it is to be easy to use.

A user interface is user centered if:

- The user can relate UI objects to concepts the user understands
- The UI objects behave in ways the user expects
- The UI follows the users' language

For an information model to be user centered, it must reflect the users' internal representation of the application domain. For an idiom model to be user centered, it must enable the user to perform operations in the application domain with computer operations he naturally associates with corresponding domain operations. To be user centered, the presentation must communicate the information model and the idiom model to the user in a manner that the user can understand, based on past experience or knowledge.

A user interface is task mapped if it has these 3 characteristics:

- Meaningful: Task mapping is meaningful if the required actions can be generated from the semantics of the task.
- Interactive: Task mapping is interactive if the device and display implicitly suggest the correct mapping.
- Consistent: Task mapping is consistent if it shares structural properties, such as syntax, with other task-action mappings.

The user must be able to easily translate the workflow in the application domain into a series of actions in the computing domain. This requires the relationships among the objects in the information model be easily translatable, by the user, into actions taken in the application domain. The idioms in the idiom model must mirror how the user relates actions in the idiom domain with actions in the computing domain.

The requirement for consistency is probably the most important. Tests have shown that even a small inconsistency will disrupt a users' ability to learn and use a user interface. Sometimes an inconsistency will cause usability errors in places besides where the inconsistent design feature appears. For instance, a user may misconstrue another user interface idiom in an attempt to make it consistent with one that uses a different idiom to represent the same information structure, or the same idiom to represent a different information structure.

Evaluation Method

The first step in the EC Wise Heuristic Evaluation Methodology is to gather information about users and tasks. Information about users comes from several sources that may already be available. If a user profiling survey has been performed, it may already have much of the required information. For instance, the profiling survey asks about what websites the users visit most often, and what websites they consider the most usable. An examination of those websites will reveal the idioms with which they are most familiar, and the presentation techniques with which they are most comfortable.

Information about tasks comes from use cases and scenarios used in connection with the development of the product or service, or in connection with associated usability tests. If use cases or scenarios are not available, we would need to generate them, through interview with representative users and domain experts.

After the information is gathered, the heuristic evaluation methodology is conceptually a 3-stage, 3 pass process, as described below:

- Develop external task description - the way the task should be done
- Develop internal task description - the way the user internalizes how to do work in the application domain
- Step through the tasks, matching the external and internal task descriptions, 3 times:
 - First pass, evaluating the information model
 - Second pass, evaluating the idiom model
 - Third pass, evaluating the presentation

During each pass of matching the external and internal task descriptions, we use our experienced engineering judgment to evaluate each task step in accordance with the 7 dialogue principles from ISO 9241. A full description of the ISO 9241 dialogue principles is contained in Appendix A.

The first time through this three stage, three pass process, we often discover that we need additional information about users or tasks. In that eventuality, we complete the current stage with partial information and then gather additional information and start the three stage process again. This iterative approach allows the evaluator to more rapidly converge on a good evaluation of the user interface.

Appendix A: ISO 9241 Ergonomic requirements for office work with visual display terminals. Part 10 - Dialogue Principles

A user interface should be based on accepted user interface and user interaction design principles in order to build optimal systems. The general principles from ISO 9241 part 10 may be more or less important depending on the application: task adequacy, self-descriptiveness, controllability, conformity to user expectations, error tolerance, flexibility, and learnability.

Task adequacy

This requirement should help to avoid that work with software tools makes it more difficult to find a solution to the actual problem instead of actually supporting it. The former would be a sign that the task requirements of the user were insufficiently known during the development of the tool.

To be more precise, the user should not be burdened to carry out unnecessary interaction steps, e.g. if the position of the cursor on the screen logically follows from the course of the task, it should be placed there automatically, in the absence of overriding arguments to the contrary.

Self-descriptiveness

The interaction between user and application (working tool) becomes more difficult if the user does not understand the relationship between the system functionality on the one hand and the requirements of his work context on the one hand.

Therefore, transparency and flexibility of the interface should always be in the foreground during the development of an application. In order to think and see through the system the user can be supported by a help function. For example: During a "save" function the message "File is saved to drive a: " is displayed.

Controllability

It is important to find a useful middle course between automation of work and full control by the user. The requirement of controllability refers to the principle possibility of influence by the user. The limits are reached at a point where results of system processes are pressed on the user while he or she is not able to influence the intermediate steps, if necessary.

For example: it should be possible to control the speed of the dialogue or interaction, the selection and order of the working tools as well as type and extent of the output.

In this respect, flexibility must be to the fore also: the user should be able to freely choose the working tools and determine the work process himself.

For example: The user should usually have the choice between mouse or keyboard as an input device.

Conformity to user expectations

Professional experience of the user should be taken into account with the introduction of a new system. Therefore, the user interface should conform to user's expectations in order to avoid that the user has to adapt his familiar terminology to the dialogue style. However, if this is at the expense of system performance, a reasonable middle course must be found in co-operation with the user.

A standardization of the work routine and of the interface features supports a 'reliable' task environment, especially if different application programs have to be integrated.

For example: it should be possible to quit the dialogue with different application contexts in one and the same way.

Error tolerance

Ergonomic design principles are focused to support learnability processes. Therefore, an error situation will be considered a learning situation that gives the user an opportunity to develop.

Thus, the user should not feel the behavior of the system in an error situation as punishment, but still be able to achieve the intended results in spite of faulty input. In this respect, an error message should not contain evaluations of judgment (e.g. absurd input) but need to be phrased in an objective and constructive way. In order to put them right, errors should be made transparent for the user, i.e. comprehensible.

For example: a dialogue box containing information about an error just occurred should on demand point at possibilities to recover from this error.

Flexibility

The user should be able to adjust the application to his individual needs and requirements without great efforts. However, a vast range of design possibilities cannot replace accurate ergonomic design. Therefore, the amount of liberty the user receives should be carefully balanced so that arising problems are excluded. For example: the user can set the scroll speed of the text on the screen.

Learnability

The user should be able to get used to the application without great efforts. In order to make the dialogue between user and system easier, he can be supported by systems which include the relevant learning strategies (e.g. via examples), offer transparent systems (see above error tolerance), and provide an aid for the user (e.g. help functions).

For example: a standardized menu layout is maintained throughout all parts of the system.