Chapter 2 *Heuristic Evaluation*

Jakob Nielsen
SunSoft

Most usability engineering methods can contribute substantially to the usability of the resulting interface *if only they were used* during the project lifecycle. But even the best method will have zero impact on the product if it does not get used. Unfortunately, experience has shown that many developers find many usability methods to be intimidating, too expensive, and too difficult and time-consuming to use (Nielsen 1994a). In order to overcome this problem, I have long advocated the use of so-called discount usability engineering (Nielsen 1989b, 1990b) with methods that are cheap, fast, and easy to use. Heuristic evaluation is one of the main discount usability engineering methods. It is easy (can be taught in a half-day seminar); it is fast (about a day for most evaluations); and it is as cheap as you want it. Even though the results of using heuristic evaluation improve with the more you know and the more carefully you apply the method, one of its virtues is that the "intimidation barrier" is very low, leading to immediate gratification.

2.1 *How to Conduct a Heuristic Evaluation*

Heuristic evaluation (Nielsen and Molich 1990; Nielsen 1992a) is a usability engineering method (Nielsen 1993a) for finding the usability problems in a user interface design so that they can be attended to as part of an iterative

design process. Heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the "heuristics").

In general, heuristic evaluation is difficult for a single individual to do because one person will never be able to find all the usability problems in an interface. Luckily, experience from many different projects has shown that different people find different usability problems. Therefore, it is possible to improve the effectiveness of the method significantly by involving multiple evaluators. Figure 2.1 shows an example from a case study of heuristic evaluation where 19 evaluators were used to find 16 usability problems in a voice response system allowing customers access to their bank accounts (Nielsen 1992a). Each of the black squares in Figure 2.1 indicates the finding of one of the usability problems by one of the evaluators. The figure clearly shows that there is a substantial amount of nonoverlap between the sets of usability problems found by different evaluators. It is certainly true that some usability problems are so easy to find that they are found by almost everybody, but there are also some problems that are found by very few evaluators. Furthermore, one cannot just identify the best evaluator and rely solely on that person's findings. First, as further discussed in Section 2.5 on page 58, it is not necessarily true that the same person will be the best evaluator every time. Second, some of the hardest-to-find usability problems (represented by the leftmost columns in Figure 2.1) are found by evaluators who do not otherwise find many usability problems. Therefore, it is necessary to involve multiple evaluators in any heuristic evaluation (see the section titled "Determining the Number of Evaluators" on page 32 for a discussion of the best number of evaluators). My recommendation is normally to use three to five evaluators since one does not gain that much additional information by using larger numbers.

Heuristic evaluation is performed by having each individual evaluator inspect the interface alone. Only after all evaluations have been completed are the evaluators allowed to communicate and have their findings aggregated. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator. The results of the evaluation can be recorded either as written reports from each evaluator or by having the evaluators verbalize their comments to an observer as they go through the interface. Written reports have the advantage of presenting a formal record of the evaluation, but require an additional effort by the evaluators and the need to

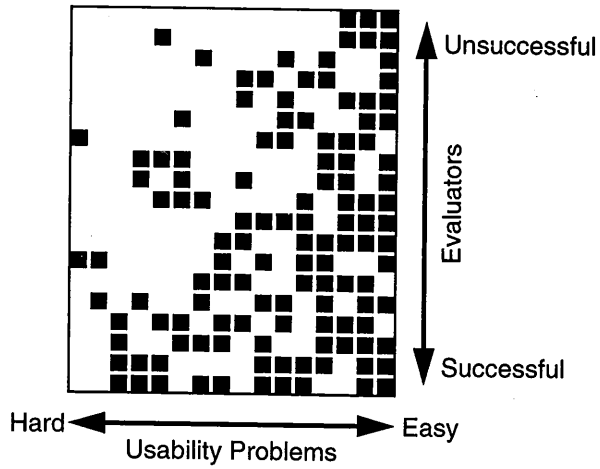

Figure 2.1 *Illustration showing which evaluators found which usability problems in a heuristic evaluation of a banking system. Each row represents one of the 19 evaluators and each column represents one of the 16 usability problems. Each square shows whether the evaluator represented by the row found the usability problem represented by the column: The square is black if this is the case and white if the evaluator did not find the problem. The rows have been sorted in such a way that the most successful evaluators are at the bottom and the least successful are at the top. The columns have been sorted in such a way that the usability problems that are the easiest to find are to the right and the usability problems that are the most difficult to find are to the left.*

be read and aggregated by an evaluation manager. Using an observer adds to the overhead of each evaluation session, but reduces the workload on the evaluators. Also, the results of the evaluation are available fairly soon after the last evaluation session since the observer only needs to understand and organize one set of personal notes, not a set of reports written by others. Furthermore, the observer can assist the evaluators in operating the interface in case of problems, such as an unstable prototype, and help if the evaluators have limited domain expertise and need to have certain aspects of the interface explained.

In a user test situation, the observer (normally called the "experimenter") has the responsibility of interpreting the user's actions in order to infer how these actions are related to the usability issues in the design of the interface. This makes it possible to conduct user testing even if the users do not know

anything about user interface design. In contrast, the responsibility for analyzing the user interface is placed with the evaluator in a heuristic evaluation session, so a possible observer only needs to record the evaluator's comments about the interface, but does not need to interpret the evaluator's actions.

Two further differences between heuristic evaluation sessions and traditional user testing are the willingness of the observer to answer questions from the evaluators during the session and the extent to which the evaluators can be provided with hints on using the interface. For traditional user testing, one normally wants to discover the mistakes users make when using the interface; the experimenters are therefore reluctant to provide more help than absolutely necessary. Also, users are requested to discover the answers to their questions by using the system rather than by having them answered by the experimenter. For the heuristic evaluation of a domain-specific application, it would be unreasonable to refuse to answer the evaluators' questions about the domain, especially if nondomain experts are serving as the evaluators. On the contrary, answering the evaluators' questions will enable them to better assess the usability of the user interface with respect to the characteristics of the domain. Similarly, when evaluators have problems using the interface, they can be given hints on how to proceed in order not to waste precious evaluation time struggling with the mechanics of the interface. It is important to note, however, that the evaluators should not be given help until they are clearly in trouble and have commented on the usability problem in question.

Typically, a heuristic evaluation session for an individual evaluator lasts one or two hours. Longer evaluation sessions might be necessary for larger or very complicated interfaces with a substantial number of dialogue elements, but it would be better to split up the evaluation into several smaller sessions, each concentrating on a part of the interface.

During the evaluation session, the evaluator goes through the interface several times and inspects the various dialogue elements and compares them with a list of recognized usability principles. These heuristics are general rules that seem to describe common properties of usable interfaces (see Table 2.1 and Table 2.2 for two possible sets of heuristics). In addition to the checklist of general heuristics to be considered for all dialogue elements, the evaluator obviously is also allowed to consider any additional usability principles or

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

Table 2.1 *Original list of usability heuristics. This specific list was developed by the author and Rolf Molich (Molich and Nielsen 1990), but it is similar to other usability guidelines. Each heuristic is discussed in considerable detail in (Nielsen 1993a). The last heuristic, help and documentation, was added in 1991.*

results that come to mind that may be relevant for any specific dialogue element. Furthermore, it is possible to develop category-specific heuristics that apply to a specific class of products as a supplement to the general heuristics. One way of building a supplementary list of category-specific heuristics is to perform competitive analysis and user testing of existing products in the given category and try to abstract principles to explain the usability problems that are found (Dykstra 1993).

In principle, the evaluators decide on their own how they want to proceed with evaluating the interface. A general recommendation would be that they go through the interface at least twice, however. The first pass would be intended to get a feel for the flow of the interaction and the general scope of the system. The second pass then allows the evaluator, to focus on specific interface elements while knowing how they fit into the larger whole.[1]

[1] This two-pass approach is similar in nature to the phased inspection method for code inspection (Knight and Myers 1993). In theory, it should be possible to extend the phased inspection approach fully to heuristic evaluation and have evaluators conduct several passes through the interfaces, each time looking for violations of a single heuristic. In practice, however, such a multi-pass approach would be tedious, and it would seem unnatural to evaluators to "overlook" usability problems that were not related to the one issue they were supposed to inspect for in a given pass.

- *Visibility of system status:* The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- *Match between system and the real world:* The system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- *User control and freedom:* Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- *Consistency and standards:* Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- *Error prevention:* Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- *Recognition rather than recall:* Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- *Flexibility and efficiency of use:* Accelerators—unseen by the novice user—may often speed up the interaction for the expert user to such an extent that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- *Aesthetic and minimalist design:* Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- *Help users recognize, diagnose, and recover from errors:* Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- *Help and documentation:* Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Table 2.2 *Revised set of usability heuristics derived from a factor analysis of 249 usability problems (Nielsen 1994c).*

Since the evaluators are not *using* the system as such (to perform a real task), it is possible to perform heuristic evaluation of user interfaces that exist on paper only and have not yet been implemented (Nielsen 1990c). This makes heuristic evaluation suited for use early in the usability engineering lifecycle.

If the system is intended as a walk-up-and-use interface for the general population or if the evaluators are domain experts, it will be possible to let the evaluators use the system without further assistance. If the system is domain-dependent and the evaluators are fairly naive with respect to the domain of the system, it will be necessary to assist the evaluators to enable them to use the interface. One approach that has been applied successfully is to supply the evaluators with a typical usage scenario (Carroll 1994; Carroll and Rosson 1990; Clarke 1991; Nielsen 1990c, 1994b), listing the various steps a user would take to perform a sample set of realistic tasks (as further discussed in Section 2.2 on page 36. Such a scenario should be constructed on the basis of a task analysis of the actual users and their work in order to be as representative as possible of the eventual use of the system.

The output from using the heuristic evaluation method is a list of usability problems in the interface with references to those usability principles that were violated by the design in each case in the opinion of the evaluator. It is not sufficient for evaluators to simply say that they do not like something; they should explain *why* they do not like it with reference to the heuristics in Table 2.1 or 2.2 or to other usability results. The evaluators should try to be as specific as possible and should list each usability problem separately. For example, if there are three things wrong with a certain dialogue element, all three should be listed with reference to the various usability principles that explain why each particular aspect of the interface element is a usability problem. There are two main reasons to note each problem separately: First, there is a risk of repeating some problematic aspect of a dialogue element, even if it were to be completely replaced with a new design, unless one is aware of all its problems. Second, it may not be possible to fix all usability problems in an interface element or to replace it with a new design, but it could still be possible to fix *some* of the problems if they are all known.

Heuristic evaluation does not provide a systematic way to generate fixes to the usability problems or a way to assess the probable quality of any redesigns. However, because heuristic evaluation aims at explaining each observed usability problem with reference to established usability principles, it will often be fairly easy to generate a revised design according to the guidelines provided by the violated principle for good interactive systems. Also, many usability problems have fairly obvious fixes as soon as they have been identified.

For example, if the problem is that the user cannot copy information from one window to another, then the solution is obviously to include such a copy feature. Similarly, if the problem is the use of inconsistent typography in the form of upper/lower case formats and fonts, the solution is obviously to pick a single typographical format for the entire interface. Even for these simple examples, however, the designer has no information to help design the exact changes to the interface (e.g., how to enable the user to make the copies or on which of the two font formats to standardize).

One possibility for extending the heuristic evaluation method to provide some design advice is to conduct a debriefing session after the last evaluation session. The participants in the debriefing should include the evaluators, any observer used during the evaluation sessions, and representatives of the design team. The debriefing session would be conducted primarily in a brainstorming mode and would focus on discussions of possible redesigns to address the major usability problems and general problematic aspects of the design. A debriefing is also a good opportunity for discussing the positive aspects of the design, since heuristic evaluation does not otherwise address this important issue.

Heuristic evaluation is explicitly intended as a "discount usability engineering" method (Nielsen 1989b, 1990b, 1993a, 1994a). Independent research (Jeffries et al. 1991) has indeed confirmed that heuristic evaluation is a very efficient usability engineering method. One recent case study found a benefit-cost ratio for a heuristic evaluation project of 48: The cost of using the method was about $10,500 and the expected benefits were about $500,000 (Nielsen 1994a). As a discount usability engineering method, heuristic evaluation is not guaranteed to provide "perfect" results or to find every last usability problem in an interface.

Determining the Number of Evaluators

In principle, individual evaluators can perform a heuristic evaluation of a user interface on their own, but the experience from several projects indicates that fairly poor results are achieved when relying on single evaluators. Averaged over six projects (Molich and Nielsen 1990; Nielsen and Molich 1990; Nielsen 1992a), single evaluators found only 35 percent of the usability problems in the interfaces. However, since different evaluators tend to find different problems, it is possible to achieve substantially better performance by aggregating the evaluations from several evaluators. Figure 2.2

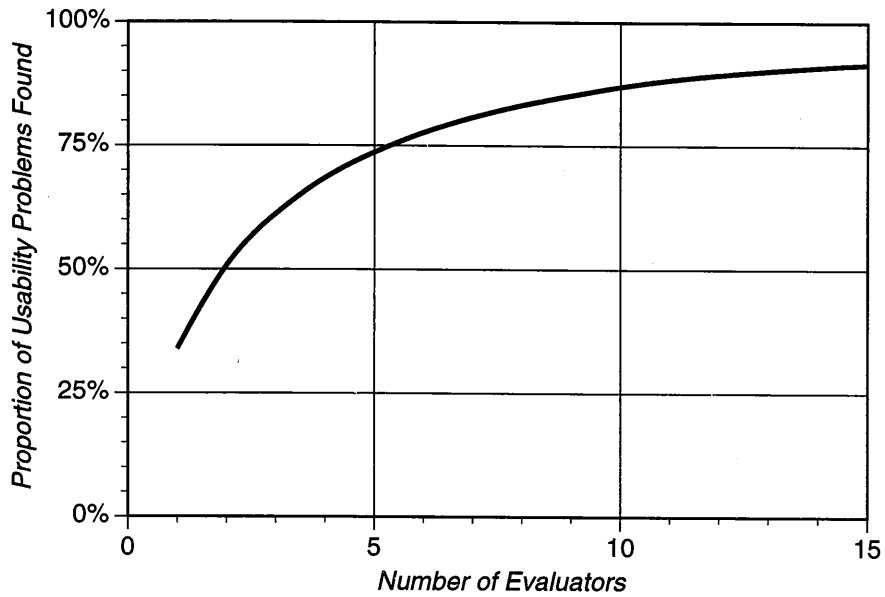

Figure 2.2 *Curve showing the proportion of usability problems in an interface found by heuristic evaluation using various numbers of evaluators. The curve represents the average of six case studies of heuristic evaluation (Nielsen 1992a).*

shows the proportion of usability problems found as more and more evaluators are added. The figure clearly shows that there is a nice payoff from using more than one evaluator. It would seem reasonable to recommend the use of about five evaluators, but certainly at least three. The exact number of evaluators to use would depend on a cost-benefit analysis. More evaluators should obviously be used in cases where usability is critical or when large payoffs can be expected due to extensive or mission-critical use of a system.

Nielsen and Landauer (1993) present such a model based on the following prediction formula for the number of usability problems found in a heuristic evaluation:

$$\text{ProblemsFound}(i) = N(1 - (1-\lambda)^i) \qquad \textbf{(EQ 1)}$$

where ProblemsFound(i) indicates the number of different usability problems found by aggregating reports from i independent evaluators, N indicates the total number of usability problems in the interface, and λ

indicates the proportion of all usability problems found by a single evaluator. In six case studies (Nielsen and Landauer 1993), the values of λ ranged from 19 percent to 51 percent with a mean of 34 percent. The values of N ranged from 16 to 50 with a mean of 33. Using this formula results in curves very much like that shown in Figure 2.2, though the exact shape of the curve will vary with the values of the parameters N and λ, which again will vary with the characteristics of the project.

In order to determine the optimal number of evaluators, one needs a cost-benefit model of heuristic evaluation. The first element in such a model is an accounting for the cost of using the method, considering both fixed and variable costs. Fixed costs are those that need to be paid no matter how many evaluators are used; these include time to plan the evaluation, get the materials ready, and write up the report or otherwise communicate the results. Variable costs are those additional costs that accrue each time one additional evaluator is used; they include the loaded salary of that evaluator as well as the cost of analyzing the evaluator's report and the cost of any computer or other resources used during the evaluation session. Based on published values from several projects the fixed cost of a heuristic evaluation is estimated to be between $3,700 and $4,800 and the variable cost of each evaluator is estimated to be between $410 and $900.

The actual fixed and variable costs will obviously vary from project to project and will depend on each company's cost structure and on the complexity of the interface being evaluated. For illustration, consider a sample project with fixed costs for heuristic evaluation of $4,000 and variable costs of $600 per evaluator. In this project, the cost of using heuristic evaluation with i evaluators is thus $\$(4{,}000 + 600i)$.

The benefits from heuristic evaluation are mainly due to the finding of usability problems, though some continuing education benefits may be realized to the extent that the evaluators increase their understanding of usability by comparing their own evaluation reports with those of other evaluators. For this sample project, assume that it is worth $15,000 to find each usability problem, using a value derived by Nielsen and Landauer (1993) from several published studies. For real projects, one would obviously need to estimate the value of finding usability problems based on the expected user population. For software to be used in-house, this value can be estimated based on the expected increase in user productivity; for software to

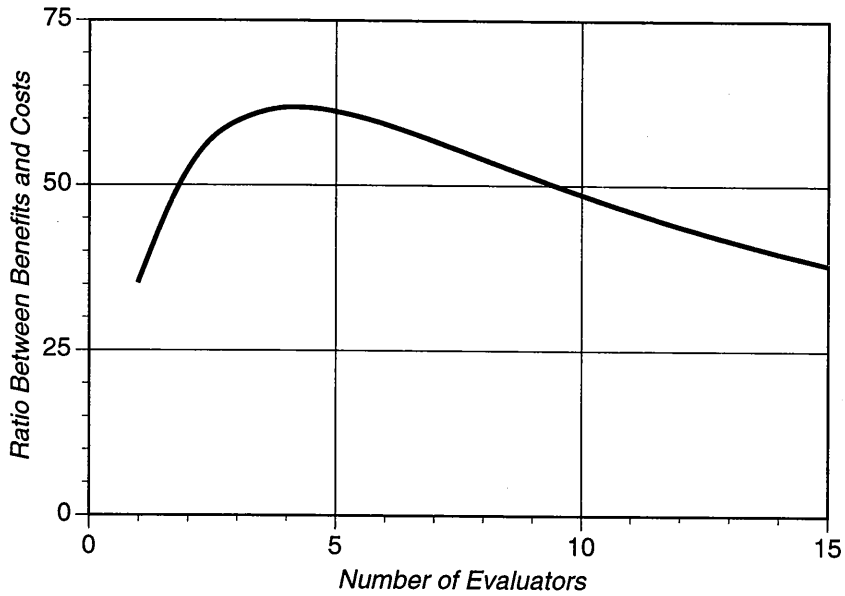

Figure 2.3 *Curve showing how many times the benefits are greater than the costs for heuristic evaluation of a sample project using the assumptions discussed in the text. The optimal number of evaluators in this example is four, with benefits that are 62 times greater than the costs.*

be sold on the open market, it can be estimated based on the expected increase in sales due to higher user satisfaction or better review ratings (Nielsen 1993c). Note that real value only derives from those usability problems that are in fact fixed before the software ships. Since it is impossible to fix all usability problems, the value of each problem found is only some proportion of the value of a fixed problem.

Figure 2.3 shows the varying ratio of the benefits to the costs for various numbers of evaluators in the sample project. The curve shows that the optimal number of evaluators in this example is four, confirming the general observation that heuristic evaluation seems to work best with three to five evaluators. In the example, a heuristic evaluation with four evaluators would cost $6,400 and would find usability problems worth $395,000.

2.2 *Case Study: Evaluating a Highly Domain-Dependent System*

This section presents detailed information about the use of heuristic evaluation to find usability problems in a rather complex telephone company application. Most uses of heuristic evaluation have concerned interfaces for broader categories of users, such as online information services or word processors, but as this example shows, it is also possible to use the method for interfaces that are intended for very specialized user populations.

The Integrating System

The domain-specific interface that was evaluated for this study was a prototype user interface for a system for internal telephone company use; it will be called the Integrating System in this chapter. Understanding the details of the Integrating System is fairly complicated and requires extensive knowledge of telephone company concepts, procedures, and databases. Since a detailed explanation is not necessary to understand the generally applicable lessons from the study, the Integrating System will only be outlined here. Briefly, the Integrating System provides a graphical user interface to access information from several backend systems in a uniform manner, despite the differences between the backend systems. The Integrating System can be used to resolve certain problems when data inconsistencies require manual intervention by a technician because the computer systems cannot determine which information is correct. The traditional method for resolving these problems involves having the technician compare information across several of these databases by accessing them through a number of traditional alphanumeric terminal sessions. The databases reside on different computers and have different data format and user interface designs, so this traditional method is somewhat awkward and requires the technicians to learn a large number of inconsistent user interfaces.

Performing this task involves a large amount of highly domain-specific knowledge about the way the telephone system is constructed and the structure of the different databases. Technicians need to know where to look for what data and how the different kinds of data are related. Also, the individual data items themselves are extremely obscure for people without detailed domain knowledge.

It should be noted that, even though it might seem possible in theory to avoid many of the underlying data inconsistencies by use of various well-known computer science techniques, maintaining consistency has proven very difficult because of the complexity and opportunity for human error involved in the application. The real-world systems underlying this application involve a large number of databases that have been in place for a long time with literally billions of data elements which cannot easily be changed, thus making some solution like the Integrating System necessary.

At the time of the usability study reported here, the Integrating System was prototype software that was not yet connected to the multitude of backend computers holding the real databases. Therefore, the system could not be used to freely access data but could only provide access to the information needed to resolve one particular request for manual assistance for one given problem. On the basis of a task analysis of the way real technicians resolved similar problems, a sample problem had been defined together with a series of steps a technician could go through to resolve it. Users had access to most of the functionality of the full Integrating System system, but could only retrieve the data that had been determined as relevant to the sample problem. Requests for any other data resulted in an error message. In other words, users of the prototype were restricted to working within a simple, prespecified usage scenario (Nielsen 1990c), but they could vary the sequence in which they initiated the various steps.

It should be emphasized that the user interface evaluated in this study was indeed an experimental prototype. Although it was developed by a highly skilled, experienced team, it was nevertheless *expected* to contain usability problems.[2] Indeed, the number of interface problems found was not unusual: Such problems are unavoidable without evaluation; iterative design is a necessary part of the usability engineering lifecycle (Nielsen 1993b). Finding as many usability problems as possible in the prototype stage of a project makes them considerably easier and cheaper to fix then than if the problems were to be discovered in later phases of the lifecycle (or not at all!).

[2] The presence of several such problems in the interface should not be taken as a sign that its designers were not competent or were insensitive to usability. Most of the evaluators were impressed by the general quality of the prototype user interface. The focus of the present discussion is the problems in the interface rather than its positive aspects, but readers should not interpret this focus as implying that the interface was not in other ways quite good.

The Evaluators

The evaluators were 11 usability specialists who had not been involved with the design of the prototype interface. Only one of the evaluators had a real understanding of the domain; the other 10 had no knowledge of the domain. To assess the evaluators' domain knowledge, they were asked to indicate their level of understanding of 17 system-related terms on a 1 to 3 scale in which 1 = does not know term, 2 = has some idea of its meaning, and 3 = knows the meaning of the term. As a control, the questionnaire also contained an eighteenth term with no meaning; this term was rated an average of 1.1, indicating that it was indeed unknown to the evaluators and that they did not overestimate their own knowledge. The evaluators gave the terms a mean rating of 1.9 when excluding the rating of the meaningless control term. This rating indicates that the evaluators did not have sufficient understanding of the central concepts of the system to be able to use it without help.

One should note the potential problems in using this term assessment method for the purpose of checking whether the terminology in a user interface is understandable. People who claim not to know a term probably do not know it. In contrast, people who claim that they do know a term may only *think* they know it, but in fact may be thinking about a related term with a different meaning. For the purpose of the current study, however, we were aware that the terminology in the system was somewhat obscure to non-specialists but could assume that it was familiar to the highly trained technicians who would be the intended real users. In contrast to traditional usability studies, the purpose of the questionnaire was therefore not to assess the terminology but to assess the evaluators.

Procedure for the Heuristic Evaluation

The heuristic evaluation consisted of four phases: a pre-evaluation training session, the actual evaluation, a debriefing session to discuss the outcome of the evaluation, and a severity rating phase during which the evaluators assessed the severity of the usability problems that had been found in the evaluation session. The first three phases were compressed into two days of real time, ensuring that a list of the usability problems found in the interface could be made available to the project developers at the end of the second day.

Briefing on the Method, the Domain, and the Scenario

Since the evaluators knew much too little about the domain to be able to use the system without help, the actual heuristic evaluation was preceded by a short (90 minutes) training session. First, the evaluators were given a general introduction to the heuristic evaluation method and were reminded of the set of heuristics. Since the evaluators were all usability specialists,[3] it was not necessary to teach these well-known usability principles as such; so, the presentation was mostly intended to provide a shared terminology for referring to usability problems and to ensure that the evaluators considered a broad spectrum of usability concerns when judging the interface.

The evaluators were then given a lecture on the domain being addressed by the Integrating System and were told about the general concepts of resolving database inconsistencies. Obviously, such a short lecture could in no way make the evaluators experts in this complicated domain, but it could at least give them some notion of the purposes of the system.

After the lecture on the domain, the evaluators were given a presentation on the specific scenario they were going to go through with the system. To ensure that the evaluators approached the Integrating System user interface from a fresh and unbiased perspective, they were not shown any screen-dumps of the actual system, nor were they told of the specific interaction techniques involved in using it. The presentation of the scenario covered each step in the resolution of the sample problem with respect to the underlying semantics of the steps and the domain-related reasons for the steps. For each step, the scenario explained what information had been retrieved by the previous step, how a technician would analyze that information, and what further action would be needed as a result of that analysis. For example, one step involved looking up further information about telephone numbers that had been found in a previous step to be associated with two addresses of interest. The scenario contained a total of seven such steps.

[3] No official certification exists to determine whether a person is a usability specialist. The evaluators in this study were employed with usability as the only or major part of their job description and had an average of seven years experience in the human-computer interaction field.

The Actual Evaluations

For the actual heuristic evaluation sessions, each evaluator conducted his or her evaluation individually and independently of the other evaluators. The evaluators were asked not to discuss the evaluation results until all evaluators had completed their evaluations. This procedure was intended to ensure unbiased evaluations from all evaluators.

Each evaluator was given one hour to perform the evaluation and was asked to evaluate the interface in two passes: first, by stepping through the prespecified usage scenario, and second by performing a more detailed analysis of individual dialogue elements. The reliance on the usage scenario was intended to expose the evaluators to the flow of the system and to give them as realistic a feel for the use of the system as possible, given that they did not really understand the domain of the system.

The list of steps in the scenario was made available to the evaluators for use during the evaluation session. The scenario description highlighted the specific actions that needed to be taken at each stage without explaining how these actions were to be achieved in the interface. Given the information in the scenario, all the evaluators were able to complete a session with the system, going through all the steps necessary to resolve the sample problem.

The evaluators were asked to find as many usability problems in the interface as possible, including both major and minor problems. They were asked to explicitly identify each problem to the observers who made a note of the problems for further analysis. In contrast to traditional usability testing, the observers were not charged with the responsibility of interpreting the comments since the evaluators pointed out the specific elements of the user interface they wanted changed.

The evaluators were also asked to state what established usability principle was violated by each usability problem and to classify the problem as either major or minor. The evaluators were generally not very good at providing this information during the evaluation session where they were more focused on inspecting the interface and finding new usability problems. Occasionally, the observers interrupted an evaluator to ask for more explicit references to established usability principles and the severity classification for a usability problem, but these interruptions were quickly abandoned as they seemed to interfere too much with the evaluators' work flow when finding usability

problems. It appears that finding usability problems and the more detailed analysis of such problems are two different processes that should not be interleaved in a single session. Instead of the elicitation of severity classifications during the evaluation session, information about the severity of the usability problems was gained from a later survey of the evaluators, as discussed in a later section.

The evaluation sessions were observed by two specialists who made notes of the usability problems identified by the evaluators. These notes avoided the need for the evaluators to write reports listing the usability problems they had found; thus the evaluation process was speeded up since the aggregated list of usability problems was available immediately after the end of the last evaluation session.

One observer (the author) was a specialist in the heuristic evaluation method and one observer was a specialist in the Integrating System application and a member of the development team. Two observers were used since we were uncertain about the potential difficulties involved in managing the evaluation sessions; this study was our first attempt at using the heuristic evaluation method for the evaluation of a highly domain-specific interface. It turned out that the evaluation sessions ran smoothly; therefore, there should be no need to allocate more than a single observer for future studies. It is important, though, that the observer has a good understanding of the application so that the observer can answer the evaluators' questions during the evaluation sessions.

On the other hand, though, it was valuable to have a member of the development team present during the evaluation sessions. General insights gained during the observations were used later to formulate a new way of looking at certain aspects of the system. These insights were aimed at improvements in the overall interface and interaction flow rather than fixing a specific usability problem.

Debriefing

After the completion of all 11 evaluation sessions, a one-hour debriefing session was conducted with three of the evaluators, the two observers, and two additional members of the development team. All evaluators had been invited to the debriefing, but most were prevented from coming for practical

reasons. The principal content of this session was a discussion of the general characteristics of the interface as well as speculations on potential improvements to address some of the major usability problems that had been found.

A major shortcoming of the heuristic evaluation method is that it identifies usability problems without indicating how they are to be fixed. In general, ideas for appropriate redesigns have to appear magically in the heads of the designers on the basis of their sheer creative powers without special assistance from the method. Of course, in practice, the identification of a usability problem often implies an appropriate and sometimes even obvious solution.

The debriefing session indicated one possible approach towards modifying the heuristic evaluation method to include advice for the redesign phase. Several of the usability problems were discussed in more or less of a brainstorming mode, leading to several ideas for redesigns. The brainstorming format was not followed completely, however, as critical and comparative comments were allowed.

A major outcome of the debriefing session was an idea for a higher-level redesign of the interface to make it more object-oriented. This single redesign would have the potential to fix eight of the ten most severe usability problems as well as several other problems. The prototype interface treated query outputs as a plain text event, even though they were highly formatted and consisted of a predetermined number of fields with specific meaning. An alternative, more object-oriented, interface design could recognize the individual data elements on the screen as user-oriented objects, even though they had been produced as output from external database queries. Instead of the function-oriented construction of new queries into which data could be pasted, the object-oriented view would concentrate on the data and allow users to apply further queries to any selected data. For example, one possible redesign would have users select a data field and pop up a list of those external databases for which a query for that datatype would be meaningful. Thus, at the same time, the interface would be simplified (by only presenting the relevant databases) and several steps and usability problems in the construction of the query could be avoided.

Similar insights as to the need for a more object-oriented interface could have been derived either from first principles concerning graphical user interface design (Nielsen et al. 1992), a careful task analysis of the users' true information needs rather than their current work practices, or some forms of user testing. It is notable that the debriefing provided similarly concrete design ideas converting the more abstract theoretical principles into a practical opportunity for achieving a redesigned interface.

Usability Problems

Forty usability problems were found in the part of the user interface that was actually covered by the usage scenario. These problems will be referred to as the "core" usability problems, since they are related to that part of the user interface that was evaluated in depth. The user interface included a few additional features that were not part of the usage scenario; these features were not subjected to nearly the same degree of scrutiny. Such a focus on part of a user interface in an early evaluation exercise is acceptable as long as one remembers to study the remaining parts more closely later. Also, the part of the user interface that is being studied should be a fairly large and preferably important part of the overall interface. These conditions were met in our case.

One example of a major core usability problem was that the window containing the systems' responses would not automatically scroll when new information had been retrieved. It was thus possible for new information to be invisible, causing the users to wait too long before acting on it. An example of a minor problem was that the input field in a certain dialog box did not automatically get the input focus when the box appeared on the screen. Thus, users had to waste a small amount of time on clicking in the field before they could start typing.

In addition to the 40 "core" usability problems in the part of the user interface covered by the scenario, four usability problems were found in other parts of the interface that were not subjected to intensive evaluation. Since these parts of the interface were not part of the scenario, they were not evaluated by all the evaluators, but only by those who chose to explore additional parts of the system. One example of a noncore usability problem was that the `Find Text` function was case-sensitive, so that searches for `FOO`, `Foo`, and `foo` would give different results. Users may not always remember what case was used for certain text, so a case-insensitive search would be better. Since the `Find Text` function was not part of the scenario, most evaluators never used it; thus, it was not evaluated exhaustively.

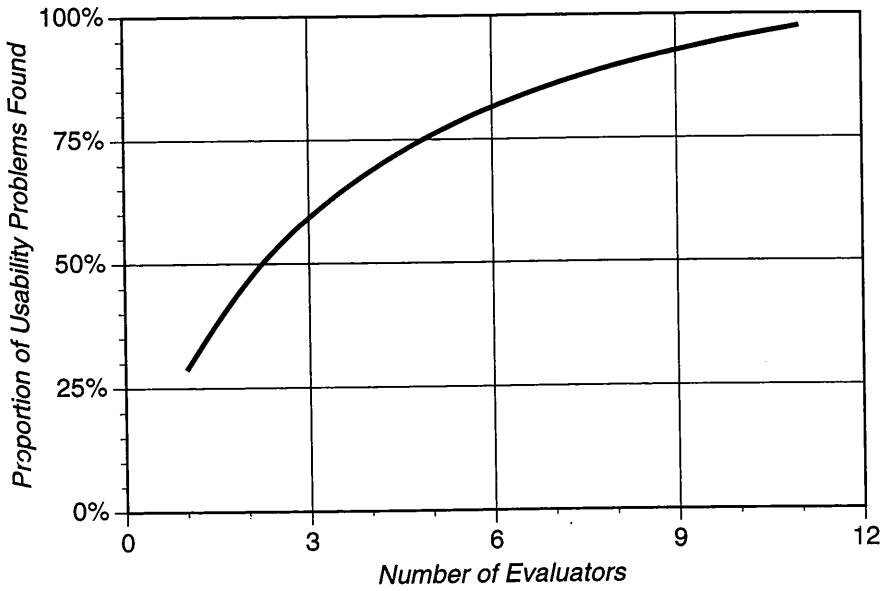

Figure 2.4 *Graph showing the number of core usability problems in the Integrating System found by aggregating the findings of 1 through 11 evaluators.*

The statistical analyses of the usability problems that are reported below have been limited to the core usability problems since they were the only ones that were investigated systematically. The only exception is the estimation of the benefits from the heuristic evaluation, where all 44 problems are taken into account. The four noncore usability problems were indeed discovered as a result of the study and therefore do contribute to the practical benefits from the use of the method.

The mean number of core usability problems found by a single evaluator was 11.5 or 29 percent, confirming the aforementioned result that one cannot rely on having a single person perform heuristic evaluation (Nielsen and Molich 1990). Figure 2.2 shows the number of usability problems that would have been found by using various numbers of evaluators from 1 to 11.

In addition to the usability problems, four minor programming errors (bugs) were discovered during the heuristic evaluation. These problems cannot be classified as usability problems as such, even though bugs obviously do impact usability negatively. The main conclusion from the discovery of bugs

during the heuristic evaluation is that users always find new ways to exercise a program; therefore, a debugging process cannot be completed without exposing the software to use outside the development team.

A further lesson from the bugs is that prototype software is notoriously prone to crashing and that one should be prepared to reboot the system during any interface testing. In fact, the Integrating System prototype was very robust for a prototype and only crashed once during the 11 hours of evaluation sessions. We had made sure to have the programmers stand by using a rotating scheme that guaranteed that one of them would always be available. Recovery from the one crash therefore took only about five minutes.

User Testing

One cannot expect heuristic evaluation to address all usability issues when the evaluators have no knowledge of the actual users and their tasks, since these two aspects are critical elements in usability engineering. Therefore, we supplemented the heuristic evaluation with a small scale user test. We conducted formal user testing with four users and also had one of the users' managers use the system. In retrospect, we regret not having taken the manager through a regular test, since we got several interesting ideas from observing her use of the system.

The user test was performed as a thinking-aloud study (Lewis 1982; Nielsen 1992b, 1993a) in which the users were asked to verbalize their thoughts while using the system. The same sample scenario as that used for the heuristic evaluation was used for the user test since the prototype only contained the data necessary to handle these specific tasks.

Seventeen of the 40 core usability problems that had been found by heuristic evaluation were confirmed by the user test. The correlation between the number of users observed having a problem and the heuristic evaluators' mean severity rating of that problem is .46 which is reasonably high (and significant at $p<.01$), lending some credence to the validity of the severity ratings. Of course, .46 is not an extremely high correlation, but there are other factors to take into account when calculating the severity of a usability problem than just how many users experience the problem. One obvious additional variable is the impact of the problem on the users and their work, which could not be measured in this test of a prototype.

One can discuss whether the 23 core problems that were not observed in the user test are in fact valid "problems" since they could not be seen to bother the real users. As argued elsewhere (Nielsen 1992a), such problems can indeed be very real, but their impact may just have too short a duration to be observable in a standard user test. Problems that have the effect of slowing users down for 0.1 second or so simply cannot be observed unless data from a very large number of users is subjected to statistical analysis, but they can be very real and costly problems nevertheless. Also, some problems may occur too infrequently to have been observed with the small number of users tested here. Finally, some problems were related to deviations from graphical user interface standards. Such deviations can be expected to impact users who have previous experience with standards-compliant interfaces or who shift between using several standards-compliant systems. However, none of the test users had any previous experience with graphical user interfaces and could not be expected to be impacted by the deviations from the standard. It would therefore be impossible to find these usability problems by user testing with these users, but they are still usability problems.

A very interesting result from the user testing was the discovery of four new usability problems that had not been found in the heuristic evaluation. Three of the problems were similar to those found by heuristic evaluation. For example, one of the new problems related to the case sensitivity of one of the query features, just as the heuristic evaluation had found a problem with the case sensitivity of the `Find Text` feature. These two case sensitivity problems are examples of exactly the same underlying phenomenon (a description error, to use Norman's terminology [Norman 1983]).

One problem was very different from those found by heuristic evaluation, however. The prototype interface produced standard error messages in separate dialog boxes when the user performed database queries for records that were not in the database. Even though this behavior is appropriate for most traditional database uses, it is suboptimal for the particular task supported by the Integrating System. Knowing that a certain record is *not* in the database is actually important information for the task of resolving data inconsistencies; the feedback message that something could not be found should therefore appear in the normal system response window integrated with other information retrieved from the databases. Dialog boxes containing error messages are normally dismissed by the user and do not remain visible to remind the user of the problem.

The fact that four new problems were found shows the value of running tests with real users to supplement heuristic evaluations. The problem that missing data is interesting information rather than an error condition is obvious in hindsight, but could probably not have been predicted by domain-ignorant usability specialists. It is truly a domain-specific usability problem, so the discovery of this problem by user testing is an indication of the need to use multiple usability engineering methods to supplement each other.

2.3 Severity Ratings

In addition to the simple list of usability problems that were found in an interface, heuristic evaluation can be used to assess the relative severity of the individual usability problems. Such severity ratings can then be used to allocate the most resources to fix the most serious problems and also provide a rough estimate of the need for additional usability efforts. If the severity ratings indicate that several disastrous usability problems remain in an interface, it will probably be unadvisable to release it. But one might decide to go ahead with the release of a system with several usability problems if they are all judged as being cosmetic in nature.

The severity of a usability problem is a combination of three factors:

- The *frequency* with which the problem occurs: Is it common or rare?
- The *impact* of the problem if it occurs: Will it be easy or difficult for the users to overcome?
- The *persistence* of the problem: Is it a one-time problem that users can overcome once they know about it or will users repeatedly be bothered by the problem?

Finally, of course, one needs to assess the *market impact* of the problem since certain usability problems can have a devastating effect on the popularity of a problem, even if they are "objectively" quite easy to overcome. Even though severity has several components, it is common to combine all aspects of severity in a single severity rating as an overall assessment of each usability problem in order to facilitate prioritizing and decision-making.

It is difficult to get good severity estimates from the evaluators during the evaluation session when they are more focused on finding new usability problems. Also, as noted previously, each evaluator will only find a small number of the usability problems, so a set of severity ratings of only the problems found by that evaluator will be incomplete. Instead, severity ratings can be collected by sending a questionnaire to the evaluators *after* the actual evaluation sessions, listing the complete set of usability problems that have been discovered, and asking them to rate the severity of each problem. Since each evaluator has only identified a subset of the problems included in the list, the problems need to be described in reasonable depth, possibly using screendumps as illustrations. The descriptions can be synthesized by the evaluation observer from the aggregate of comments made by those evaluators who had found each problem (or, if written evaluation reports are used, the descriptions can be synthesized from the descriptions in the reports). These descriptions allow the evaluators to assess the various problems fairly easily even if they have not found them in their own evaluation session. Typically, evaluators need only spend about 30 minutes to provide their severity ratings. It is important to note that each evaluator should provide individual severity ratings independently of the other evaluators.

Often, the evaluators will not have access to the actual system while they are considering the severity of the various usability problems. It is possible that the evaluators can gain additional insights by revisiting parts of the running interface rather than relying on their memory and the written problem descriptions. At the same time, there is no doubt that the evaluators will be slower at arriving at the severity ratings if they are given the option of interacting further with the system. Also, scheduling problems will sometimes make it difficult to provide everybody with computer access at convenient times if special computer resources are needed to run a prototype system or if software distribution is limited due to confidentiality considerations.

The 0 to 4 rating scale shown in Table 2.3 was used in the "Integrating System" case study for rating the severity of the usability problems.

Since any individual evaluator typically has personally found only a small fraction of the total set of usability problems, it is reasonable to ask to what extent the evaluators' judgments might be biased. One might expect the evaluators to rate their "own" problems as more severe than the problems that had been found by others but that they had overlooked themselves. In the

0 I don't agree that this is a usability problem at all
1 Cosmetic problem only—need not be fixed unless extra time is available on project
2 Minor usability problem—fixing this should be given low priority
3 Major usability problem—important to fix, so should be given high priority
4 Usability catastrophe—imperative to fix this before product can be released

Table 2.3 *Five-point rating scale for the severity of usability problems found by heuristic evaluation.*

case study, it turned out, however, that any given evaluator's severity rating of a usability problem was essentially independent of whether that evaluator had found that problem. When separating out the effect of the total number of evaluators finding each problem, the remaining partial correlation between the severity judgments from an individual evaluator and whether that evaluator found the problem was only .09. There was a positive correlation of .33 between the evaluators' ratings and the number of evaluators having found each problem. Since the individual evaluators did not know how many other evaluators had found each problem, they would have no way of letting their severity judgments depend on this number. Therefore, the correlation between the severity ratings and the overall probability of finding the problems is due to the well-established phenomenon that the more severe usability problems are found more frequently by heuristic evaluation (Nielsen 1992a).

Reliability of the Severity Judgments

The average correlation between the severity ratings provided by any two evaluators is .24. Kendall's coefficient of concordance between the eleven evaluators is $W=.31$, which is statistically significant ($\chi^2=132.3$, $df=39$, $p<.01$) and thus indicates that the agreement is not just chance. Also, of the 55 pair-wise comparisons between evaluators, only 4 have negative correlations, whereas the remaining 51 are positive ($p<.001$).

Even though the statistics indicate better than random agreement between evaluators, the inter-rater reliability is still very low compared to the standards of most respected rating methods. Basically, the reliability of the severity ratings from single evaluators is so low that it would be advisable not to base any major investments of development time and effort on such single

ratings. On the other hand, the better-than-random agreement between evaluators means that it is possible to use the mean of the severity judgments from several evaluators and get much more reliable results. It is a fairly simple task for an evaluator to produce severity ratings for an interface which is known to that evaluator from a heuristic evaluation session; the evaluators in the case study presented here spent about half an hour each on doing so. Therefore, it would seem reasonable to ask for severity judgments from all or at least most evaluators.

The Spearman-Brown formula for estimating the reliability of combined judgments from several evaluators is

$$r_{n\text{-}n} = \frac{n \cdot r_{1\text{-}1}}{1 + (n-1) \cdot r_{1\text{-}1}} \qquad \textbf{(EQ 2)}$$

Figure 2.5 shows a plot of the way the reliability of the mean severity estimate increases as more evaluators are added.

The standard error of measurement for the true underlying value of a rating derived as the mean of n ratings is

$$\sigma_{\infty} = \sigma_n \cdot \sqrt{1 - r_{n\text{-}n}} \qquad \textbf{(EQ 3)}$$

where $r_{n\text{-}n}$ is the reliability of a group of n raters and σ_n is the standard deviation of the mean of the n ratings which again is

$$\sigma_n = \frac{\sigma_1}{\sqrt{n}} \qquad \textbf{(EQ 4)}$$

if the individual ratings can be assumed to be independent variables. This assumption of independence obviously only holds if the evaluators perform their evaluations separately (as was done in this study) and do not discuss the usability problems before giving their severity judgments.

Combining equations 2 to 4 gives the standard error of measurement shown in Figure 2.5. Since the severity judgments are fairly close to following a normal distribution as shown in Figure 2.6, the normal distribution can be

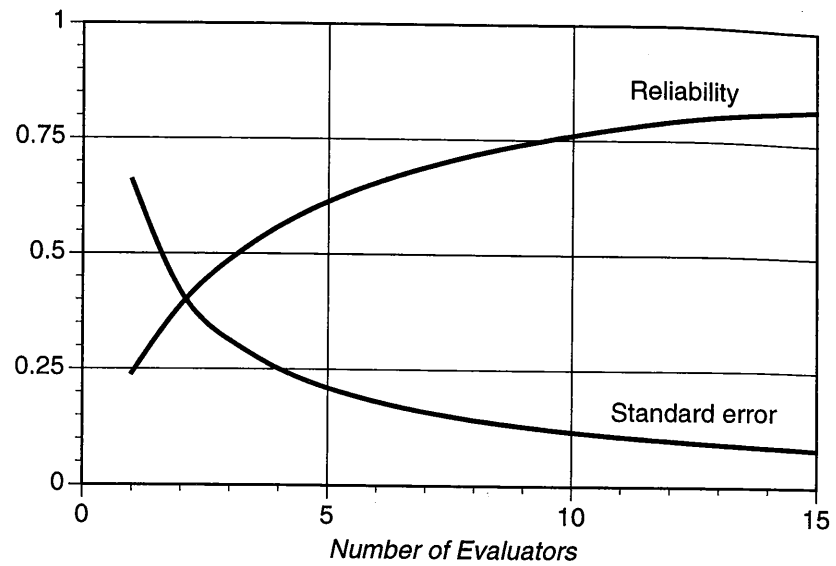

Figure 2.5 *The upper curve shows the reliability of the severity estimates as a function of the number of evaluators used. The lower curve shows the standard error of measurement (in rating units from the 0-4 scale) for the underlying true mean severity value when measured by the mean of the ratings from that number of evaluators. Values for more than one evaluator are plotted according to the Spearman-Brown formula.*

used to calculate confidence intervals for the severity estimates. The standard deviation of the complete set of severity ratings adjusted for means is 0.75 which can be used for a general estimate of σ_1.

Because of the low reliability of the severity ratings and the high standard deviation for the individual ratings, the probability of having a single evaluator provide an estimate that is within ±0.5 rating units of the true severity of a problem is only 55 percent. In other words, almost half of the time the absolute rating will be substantially different from the "true" rating (that would result from having an infinite number of evaluators). On the other hand, combining the estimates of several evaluators considerably improves the confidence intervals for the mean estimate. With just two evaluators, one has a 77 percent chance of getting within ±0.5 of the "true"

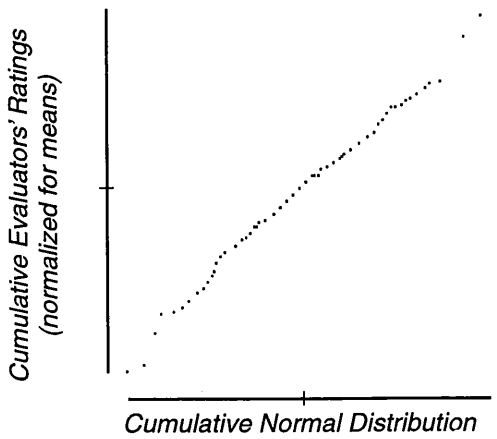

Figure 2.6 *Probability (percentile vs. percentile) plot comparing the distribution of the severity ratings minus their means and the normal distribution. The plot is close to a straight line, indicating that the severity ratings follow a distribution close to the normal distribution.*

severity, and the probability is 95 percent with four evaluators. Figure 2.7 shows how the probability of getting within ±0.5 and ±0.25 goes up with the number of evaluators.

Severity estimates for usability problems have two possible practical applications: First, one needs to know the absolute need for usability improvements in order to determine the need for continued usability efforts and to consider the extent to which an increased usability engineering budget or delays in product introduction is warranted. Second, given a specific budget for usability activities and the need to move on in the product lifecycle within a given timeframe, one needs to set relative priorities for which problems to fix first.

For the first application of estimating the absolute severity of the usability problems in an interface, it may be sufficient to estimate the severity of each usability problem with an uncertainty of ±0.5 rating units. The length of a ±0.5 interval is of course one unit; it would be difficult to interpret the meaning of subjective usability severities with a much finer resolution than that. For example, any cost-benefit estimate of the potential fixing of usability problems has to include estimates of the programming effort

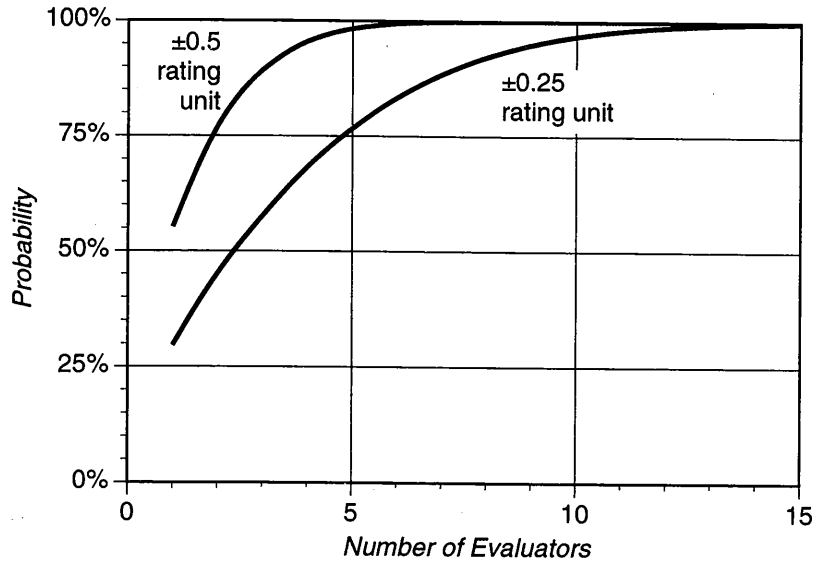

Figure 2.7 *Probability for having the mean severity rating from sets of 1 through 15 evaluators be within an interval of ±0.5 and ±0.25 rating unit of the true severity of a usability problem.*

needed to implement a redesign as well as an estimate of the usability of the new design. Neither value can be estimated with great precision in any case, so it would be a wasted effort to measure the severity of the usability deficiencies in the old design with extremely narrow confidence intervals.[4]

For the second application of deciding the relative severity of the usability problems in order to prioritize them, estimates from more evaluators will be needed for tight confidence intervals, since one is comparing two average scores, each of which has been estimated with some uncertainty.

One way of addressing the accuracy of relative severity estimates is to consider the probability with which a usability problem can be assigned to its proper decile or its neighboring decile. Deciles divide the total space of problems into 10 groups sorted by severity. For example, the first decile contains the 10 percent least severe usability problems and the tenth decile

[4] It would be valuable to have externally valid data to calibrate the subjective rating scale with respect to the economic impact of fixing usability problems of various severities.

contains the 10 percent most severe problems (that is, those rated as the 91 to 100 percent most severe). Allowing an estimate to fall within the neighboring deciles as well as the true decile would mean, for example, that we could count a claim that a particular usability problem was among the 71 to 80 percent most severe as being close enough as long as the problem in fact was among the 61 to 90 percent most severe. Obviously, allowing assignment to neighboring deciles implies a fairly low precision, since one can only be sure of having estimated the relative severity of a usability problem to within a group of 30 percent of the total set of usability problems. On the other hand, such fairly low precision is probably about as good as one needs, given that the final decision of which usability problems to fix also involves estimates of the value of possible redesigns and the programming cost of such redesigns.

The probability of assigning usability problems to their proper decile or its neighboring decile can be estimated under the assumption that the severity estimates are normally distributed (as supported by Figure 2.6) with standard errors for the various number of evaluators as shown in Figure 2.5. These probabilities have been calculated for each of the 40 core usability problems; Figure 2.8 shows the result averaged over all 40 problems. As can be seen from this figure, relative severity estimates are quite poor, with the probability of getting within one decile of the correct one being only 42 percent for a single evaluator. Six evaluators are needed for achieving an 81 percent probability of getting within one decile, and eleven evaluators are needed to reach the 95 percent level of confidence.

In some cases, less precise relative severity estimates may be adequate. Figure 2.8 also shows the probabilities for estimating within the correct quartile or its neighboring quartile. Quartiles divide the set of usability problems into 1 to 25 percent, 26 to 50 percent, 51 to 75 percent, and 76 to 100 percent severity intervals. Getting within one quartile of the correct one means that the least severe 25 percent of the problems will never be estimated within the top 50 percent, and the most severe 25 percent of the problems will never be estimated within the bottom 50 percent. In other words, one can be assured that fixing the problems that are estimated as the top 50 percent will take care of all the 25 percent most severe problems. Also, this level of precision makes sure that one will never rate one of the least severe 25 percent of the problems as more important than one of the most severe 25 percent of the problems.

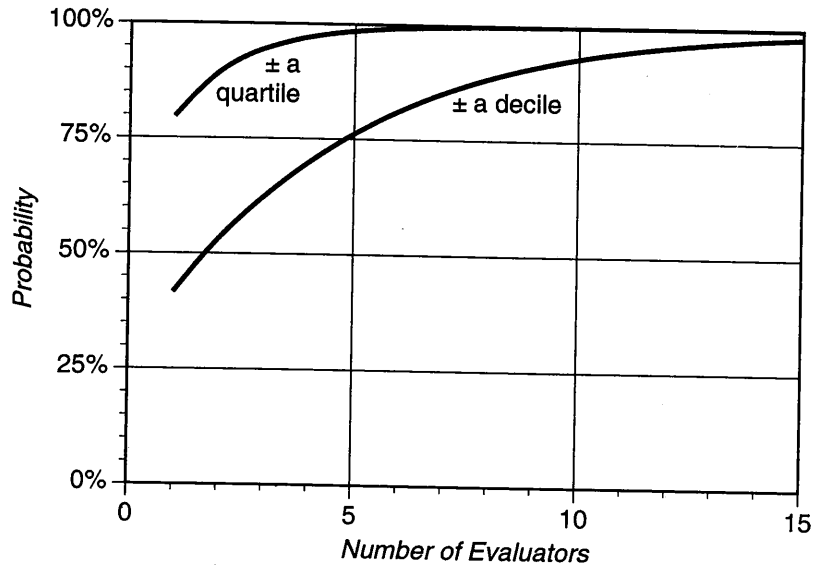

Figure 2.8 *Probability of having the mean severity rating from sets of 1 through 15 evaluators assign a usability problem to the correct quartile or decile or the neighboring quartile or decile, respectively.*

As can be seen from Figure 2.8, it is much easier to get within one quartile of correct than to get within one decile; the 94.5 percent confidence level is reached with as little as three evaluators. Even with this more liberal relative severity requirement, one should still be wary of using estimates from single evaluators, since they would get it wrong 20 percent of the time.

Based on these considerations, one can definitely conclude that severity ratings from a single evaluator are too unreliable to be trusted. As more evaluators are asked to judge the severity of usability problems, the quality of the mean severity rating increases rapidly, and ratings from three or four evaluators would seem to be satisfactory for many practical purposes.

2.4 Characteristics of Usability Problems Found by Heuristic Evaluation

Heuristic evaluation is a good method for finding both major and minor problems in a user interface. As one might have expected, major problems are slightly easier to find than minor problems, with the probability for finding a given major usability problem at 42 percent on the average for single evaluators in six case studies (Nielsen 1992a). The corresponding probability for finding a given minor problem was only 32 percent.

Even though major problems are easier to find, this does not mean that the evaluators concentrate exclusively on the major problems. In case studies of six user interfaces (Nielsen 1992a), heuristic evaluation identified a total of 59 major usability problems and 152 minor usability problems. Thus, it is apparent that the lists of usability problems found by heuristic evaluation will tend to be dominated by minor problems,[5] which is one reason severity ratings form a useful supplement to the method. Even though major usability problems are by definition the most important ones to find and to fix, minor usability problems are still relevant. Many such minor problems seem to be easier to find by heuristic evaluation than by other methods. One example of such a minor problem found by heuristic evaluation was the use of inconsistent typography in two parts of a user interface. The same information would sometimes be shown in a serif font (like this one) and sometimes in a sans serif font (like this one), thus slowing users down a little bit as they have to expend additional effort on matching the two pieces of information. This type of minor usability problem could not be observed in a user test unless an extremely careful analysis were performed on the basis of a large number of videotaped or logged interactions, since the slowdown is very small and would not stop users from completing their tasks.

Usability problems can be located in a dialogue in four different ways: at a single location in the interface, at two or more locations that have to be compared to find the problem, as a problem with the overall structure of the interface, and finally as something that *ought* to be included in the interface

[5] Using the average probabilities for finding major and minor problems, the number of major usability problems found by a single evaluator in the six case studies would be 25 and the number of minor problems would be 49. Thus, there would have been about twice as many minor problems as major problems included in each evaluation report.

but is currently missing. An analysis of 211 usability problems (Nielsen 1992a) found that the difference between the four location categories was small and not statistically significant. In other words, evaluators were approximately equally good at finding all four kinds of usability problems. However, the interaction effect between location category and interface implementation *was* significant and had a very large effect. Problems in the category "something missing" were slightly easier to find than other problems in running systems, but much harder to find than other problems in paper prototypes. This finding corresponds to an earlier, qualitative, analysis of the usability problems that were harder to find in a paper implementation than in a running system (Nielsen 1990c). Because of this difference, one should look harder for missing dialogue elements when evaluating paper mock-ups.

A likely explanation of this phenomenon is that evaluators using a running system may tend to get stuck when needing a missing interface element (and thus notice it), whereas evaluators of a paper "implementation" just turn to the next page and focus on the interface elements found there.

Alternating Heuristic Evaluation and User Testing

Even though heuristic evaluation finds many usability problems that are not found by user testing, it is also the case that it may miss some problems that can be found by user testing. Evaluators are probably especially likely to overlook usability problems if the system is highly domain-dependent and they have little domain expertise. In the case study of the Integrating System discussed in Section 2.2, several additional usability problems were found by user testing, including some that were so domain-specific that they would have been virtually impossible to find without user testing.

Since heuristic evaluation and user testing each finds usability problems overlooked by the other method, it is recommended that both methods be used. Because there is no reason to spend resources on evaluating an interface with many known usability problems only to have many of them come up again, it is normally best to use iterative design between uses of the two evaluation methods. Typically, one would first perform a heuristic evaluation to clean up the interface and remove as many "obvious" usability problems as possible. After a redesign of the interface, it would be subjected to user testing both to check the outcome of the iterative design step and to find remaining usability problems that were not picked up by the heuristic evaluation.

There are two major reasons for alternating between heuristic evaluation and user testing as suggested here. First, a heuristic evaluation pass can eliminate a number of usability problems without the need to "waste users," who sometimes can be difficult to find and schedule in large numbers. Second, these two categories of usability assessment methods have been shown to find fairly distinct sets of usability problems; therefore, they supplement each other rather than lead to repetitive findings (Desurvire et al. 1992; Jeffries et al. 1991; Karat et al. 1992).

As another example, consider a video telephone system for interconnecting offices (Cool et al. 1992). Such a system has the potential for changing the way people work and interact, but these changes will become clear only after an extended usage period. Also, as with many computer-supported cooperative work applications, video telephones require a critical mass of users for the test to be realistic: If most of the people you want to call do not have a video connection, you will not rely on the system. Thus, on the one hand field testing is necessary to learn about changes in the users' long-term behavior, but on the other hand such studies will be very expensive. Therefore, one will want to supplement them with heuristic evaluation and laboratory-based user testing so that the larger field population does not have to suffer from glaring usability problems that could have been found much more cheaply. Iterative design of such a system will be a combination of a few, longer-lasting "outer iterations" with field testing and a larger number of more rapid "inner iterations" that are used to polish the interface before it is released to the field users.

2.5 *Getting Good Evaluators*

As always in computing (Egan 1988), there are major individual differences between the performance of evaluators in heuristic evaluation. In eight case studies, the Q_3/Q_1 ratio between the number of usability problems found by the top and bottom quartile (best 25 percent versus worst 25 percent) of the evaluators ranged from 1.4 to 2.2 with a mean of 1.7. These numbers represent cases where evaluators with essentially the same background and qualifications were compared. Thus, there are major benefits to be gained if one could identify people who are good at doing heuristic evaluation as the evaluators. In one case study (Nielsen and Molich 1990), 34 evaluators with

the same background evaluated two different user interfaces, and the correlation between the number of usability problems found by individual evaluators in the two systems was $r=.57$. This definitely indicates better than random consistency in the evaluators' ability to find usability problems ($p<.01$), but at the same time also indicates substantial unexplained variability in performance from one evaluation to the next. Even though it might be possible to establish a group of "good" evaluators over time by selecting people who exhibit good performance in several evaluations, it will likely be a slow process and will sacrifice performance on the first several evaluation studies.

In a case study (Nielsen 1992a), the same user interface was subjected to heuristic evaluation by three groups of evaluators: usability "novices" with knowledge about computers in general but no special usability expertise; "single experts" who were usability specialists but not specialized in the domain of the interface; and "double experts" with expertise in both usability in general and the kind of interface being evaluated. The performance of the novice evaluators was fairly poor, with each of them finding an average of 22 percent of the usability problems in the interface. The single experts found 41 percent of the problems each, making them 1.8 times as good as the novices, and the double experts found 60 percent each, making them 2.7 times as good as the novices and 1.5 times as good as the single experts. These results show that there are systematic group differences in evaluator performance in addition to the individual differences. Even though heuristic evaluation can be performed by people with little or no usability expertise (which is an advantage from a discount usability engineering perspective), it is preferable to use usability specialists as the evaluators; optimal performance requires double specialists.

Users as Evaluators

In addition to developers, usability specialists, and "double experts," a fourth category of possible evaluators might be actual end users. One might argue that the users are the ultimate domain experts and thus might contribute some of the same insights as the "double experts." Even so, I do not recommend performing heuristic evaluations with users as evaluators because of their poor conceptual understanding of usability and computer principles.

True, users know their work, but it is a very common experience in usability engineering that they do not know how their *future* work might be changed by new computer systems.

To the extent that one has access to real users during product development it is likely to generate substantially better insights to let them act as *users* and have them try out the user interface as part of a user test where they perform real tasks with the system. Instead of asking users whether they think that a certain part of an interface design is easy to understand, one can directly observe whether they are able to understand it when they encounter the dialogue element in the context of a real task. This latter kind of empirical data is likely to have significantly higher validity than the users' opinions. Similarly, one can certainly ask users for suggestions on how to change or improve an interface, but since users are not designers one should treat these suggestions as a source of further inspiration and not as the ultimate truth about how to improve the interface.

Educational Use of Heuristic Evaluation

One of the ways of getting better evaluators is to have the evaluators do many heuristic evaluations and have them compare lists. The first few times people serve as evaluators in a heuristic evaluation, they will often end up finding comparatively few usability problems; but a discussion of the many additional problems found by their colleagues can be an effective learning experience (see Kahn and Prail, Chapter 6). Thus, heuristic evaluation can be used in a bootstrapping fashion to serve as its own educational method. In addition to teaching the heuristic evaluation itself, having people perform heuristic evaluations also involves other instructional opportunities since the discussion of usability problems is the main way to make abstract usability principles concrete (Nielsen and Molich 1989).

Heuristic evaluation can also be used to improve the teaching of user interface design. In one project I was involved with, we wanted to teach graphical user interface design to developers who were very experienced in the design of character-based user interfaces (Nielsen et al. 1992). Design exercises served as a major element in the course, since design is much easier to learn by doing than by listening to lectures. We were able to provide the course participants with very rapid feedback on their designs by having two usability specialists perform heuristic evaluations on the designs during the coffee breaks following the design sessions. Admittedly, two evaluators are

less than normally recommended, but the evaluators gained a status of at least double experts with respect to the exercise designs throughout the series of courses. Also, for instructional purposes it was not necessary to find all usability problems in the designs since there was not enough time in the course schedule to discuss more than a few aspects of each design. The important contribution to the course from the heuristic evaluations was in providing immediate feedback to the course participants; they did not have to rely on the traditional very slow feedback provided days or weeks after the completion of an assignment.

2.6 Conclusions

Heuristic evaluation is a basic usability engineering method that is very easy to use. The present description of heuristic evaluation includes several aspects of heuristic evaluation that may seem slightly complicated at first, such as the use of severity ratings or the evaluation of domain-specific interfaces. In fact, even these extensions to the method are quite easy. But most importantly, it is possible to start using heuristic evaluation without these elements. The basic components of heuristic evaluation are those described in Section 2.1:

- Have evaluators go through the interface twice; once to focus on its flow and once to focus on its individual dialogue elements.
- Ask the evaluators to inspect the interface with respect to whether it complies with a short list of basic usability heuristics and your general knowledge of usability principles.
- Combine the findings from about three to five evaluators and have them work independently of each other.

After the individual evaluations, you can have the evaluators meet for a debriefing session. You can also consider collecting severity ratings to help prioritize the fixing of the usability problems you will have found.

In any case, heuristic evaluation can only find usability problems in your interface if you in fact *use* the method. Reading about it is only the first step. Practical experience with using the method not only will improve whatever product you use it on, but will also make you better at using the method which, in turn, will benefit future projects.

Acknowledgments

Much of the work described in this chapter was done while the author was with Bellcore and on the faculty of the Technical University of Denmark. Gay Norwood contributed substantially to the case study of the Integrating System discussed in Section 2.2. Much of the author's early work on heuristic evaluation was done in collaboration with Rolf Molich. The author would also like to thank Tom Landauer for helpful comments on earlier versions of this manuscript.

Chapter 3

The Pluralistic Usability Walkthrough: Coordinated Empathies

Randolph G. Bias
IBM Corporation

When the first product designer first asked, aloud or subvocally, "How will my users use my [say] stone implement?" the usability walkthrough was born.

Sometime later, two or more early human factors professionals gathered to review a product design for its projected usability and the usability walkthrough became a more structured and accepted inspection method. In recent years, professionals at IBM-Austin, driven by a desire for testing efficiency, have extended the traditional usability walkthrough. Primarily, we have included three types of participants in the same walkthrough—actual users, product developers, and usability experts. This pluralistic usability walkthrough has yielded not only increased efficiency, but also increased utility, in that we have obtained new and valuable results, not typically gleaned from traditional usability walkthroughs.

This chapter describes the pluralistic usability walkthrough, including enough detail to allow the usability professional to conduct one confidently. The method's benefits and limitations are highlighted. And, in the absence of a crisp, a priori, theoretical motivation for the new approach, some of the recent empirical literature on participatory design, group dynamics, and the integration of evaluation and design are reviewed, to retrofit a theoretical explanation for the serendipitous findings. In the process, I will address this

Usability Inspection Methods

Edited by

JAKOB NIELSEN
SunSoft

and

ROBERT L. MACK
IBM T. J. Watson Research Center

John Wiley & Sons, Inc.
New York • Chichester • Brisbane • Toronto • Singapore