A Framework for Collecting and Analyzing Usability Data

Zhijun Zhang and Victor R. Basili
Department of Computer Science
University of Maryland
College Park, MD20742, USA
{zzj|basili}@cs.umd.edu

Abstract

Usability has been recognized as an important factor in software quality. In this paper, we give a framework for collecting and analysis data for software usability evaluation. We use the Goal/Question/Metric method, build a model for the human-computer interaction process, derive operational scenarios from the model, and define the questions and metrics for usability evaluation for each of these scenarios. We discuss the different mechanisms needed for collecting different usability data, and give suggestions about data analysis as well as organizational strategies for data interpretation. The framework is intended to be used to form the basic structure of an organization's usability engineering activities.

1 Introduction

Usability is now widely recognized as an important software quality alongside other aspects such as functionality and reliability. The usability issue of computer systems has become important because of the growth of the following two factors:

Application domain At first, computers have been used for tasks that depend on mathematical calculations, in which they prove to be very powerful and successful. But now computer application covers a wide range of things that people do that cannot be taken over completely by a numerical machine, including talk, understand speech and language, write, read, organize, administer, looking for information, entertain, etc.[10]. Computer systems have been designed and built to act as assistants, aids, and "power tools". In these recent applications, computers are often not doing enough. They are often too hard to operate, i.e., not quite usable.

User population With the rapid growth of computer use, the population of computer users also increases very fast. Furthermore, computers are being used DIRECTLY by more and more people with different computer literacy, including those who have never used a computer before. Sometimes a computer system must be usable for handicapped people, or people with different languages. In other words, current systems have to be usable to a larger group of people with more diversity.

The usability of a software can be defined as the extent to which it supports the effective, efficient, and satisfying use by the users for their tasks. Usability is generally associated with

CONFERENCE PROCEEDINGS
Ninth International
Software Quality Week 1996

21-24 May 1996

the following five attributes[14][17]: learnability, efficiency, memorability, error handling, and user satisfaction.

There are different approaches to usability assessment or evaluation. These approaches can be classified as expert-, theory-, and user-based[18]. User-based evaluation is also called user testing or usability testing, which involves one or more users completing one or more tasks in an appropriate environment. Theory-based evaluation involves evaluators analyzing, based on a theory, the user's interaction activities(cognition, perception, and motor) needed to complete the tasks with the system and looking for problems. Expert-based evaluation involves evaluators assessing a system's conformity to usability principles, guidelines, or standards. It may involve usability experts only, as in "heuristic evaluation"[12]; or it may involve people representing different knowledge areas, as in the "pluralistic walkthroughs" approaches[4]. Given the state-of-the-art of the other two types of evaluation, the user-based approach, i.e. user testing, provides the most dependable information for evaluating usability. At the same time, user testing is generally the most expensive one among all approaches.

In this paper, we first review the current approaches for usability evaluation, as well as some of the empirical studies that compared the different methods in terms of effectiveness, cost-effectiveness, consistency of result, etc.. We ended the review by a qualitative summarization of the approaches and the empirical studies. Then a framework is given for collecting and analyzing data for usability evaluation, in which we show that besides user testing, expert analysis before and after user testing are also very important, and that some data can be collected just by expert analysis and it's not necessary to depend everything on the expensive user testing. The aim of this framework is to perform usability evaluation in a structured and cost-effective way, with the guidance of the qualitative analysis of what kind of data need to be collected. We use the Goal/Question/Metric approach to decompose the usability evaluation goal into questions and metrics, based on a model of human-computer interaction and its submodels, identify the components that the less expensive tool analysis or expert analysis will suffice, so that user testing can be focused on the parts that other methods are not effective. We suggest practitioners to analyze the user testing records deeply to reveal the information beyond the result data themselves. We also suggest organizations to keep a database of the usability data of all its products and of its competitor's products, and to use this database to support decision making.

2 State of the practice

Many organizations in the computer industry have realized the importance of usability and have devoted efforts to usability engineering. In this section, the three types of usability evaluation methods will be described. The characteristics of each of them will be discussed. Results from some of the empirical studies will be presented in comparing these different approaches. Finally, a qualitative analysis summarizes the state-of-the-practice.

2.1 Current approaches

So far, the most common usability activity among organizations is the user-based approach. It involves building a usability lab, where a group of people can test the usability of a product by letting representative real users to work on real tasks with the system. The major activities of the people working at the usability lab are to design and conduct usability testing with users, and to analyze the data collected from the usability testing[6]. A usability lab generally consists of one or more user rooms and one observation room. The observation room is separated from the user room(s) by on-way mirror, so that the observers can see the activities of the users but the users won't see the observers. Also there are cameras and human or automatic logger to record the user's activities during user testing in very detail. The users are encouraged to "think aloud" so that the observer(s) may better understand the interaction process. Relevant developers are often asked to observe the users' interaction with the system, so that they can see the users' problems more directly. Problems are found through observation (including watching videotape) of users' activities, analysis of users' performance data, and interview or questionnaire with users after the test. Based on these detected problems, recommendations are made to change the product, and possibly the process by which it was developed.

Expert-based usability evaluation or inspection methods are also practiced in some organizations. These methods are based on expert analysis of the design or product, with regard to standards, guidelines, and heuristics, considering the characteristics of the users, tasks, and working environment.

One such method is "heuristic evaluation"[12], which involves having a set of evaluators examine the system and judge its compliance with recognized usability principles (the "heuristics"). Each individual evaluator inspects the system alone. Communication between evaluators are not allowed until all evaluations have been completed and results are aggregated. This method is fast (usually one to two hours for an individual evaluator session), easy to carry out, and flexible. There is not restriction on how the evaluators should perform the evaluation. They are not even given task scenarios. The result depends heavily on the expertise of the evaluators and the number of evaluators involved.

Another expert-based method, called "pluralistic usability walkthroughs"[4], involves the meeting of a group of people with different knowledge areas, including usability experts, software developers, and possibly users. They will work together going through a set of task scenarios. The walkthrough process includes asking each participants to write down the action(s) he/she would take for a given task. After all participants have written the action(s) they would take, a discussion begins, with the users speaking first. Only after the first round of comments from the users are exhausted do the usability experts and developers offer their opinions.

Theory-based methods are occasionally used in practice. These methods are built on models and theories. Comparing to the other two types of methods, these methods have more formal procedures and analyze the human-computer interaction process in more detail.

One theory-based method is the GOMS method. The GOMS(Goals, Operators, Methods, and Selection rules) model[5] was originally developed to characterize user behavior in text editing. It says that users want to achieve their goals (may be divided into subgoals); they are knowledge-

able of some methods for achieving a goal; they can perform certain elementary perceptual, motor, or cognitive acts (operators); and they need to follow certain selection rules for choosing the right method when there are more than one methods available. GOMS usability evaluation is to decompose user's tasks into goals, operators, methods, and selection rules, and use the model human processor (MHP) to predict the time for the user to complete the task. It is originally only for analyzing error-free interaction of expert users, but later expanded to accommodate learning and error recovery in NGOMSL[9].

Another theory-based method, cognitive walkthrough[19], is based on a theory of learning by exploration and on modern research in problem solving. It uses as inputs a description of the system, a task scenario, assumptions about the knowledge a user will bring to the task, and the specific actions a user must perform to accomplish the task with the interface. The evaluator(s) then examines each step in the correct action sequence asking the following question: (1) Will the user try to achieve the right effect? (2) Will the user notice that the correct action is available? (3) Will the user associate the correct action with the effect that the user is trying to achieve? (4) If the correct action is performed, will the user see that progress is being made toward solution of the task? This approach is focused very explicitly on one aspect of usability, ease of learning. The analysis is also low-level.

2.2 Empirical comparison of current approaches

Different methods are being tested and their pros and cons are being presented and debated.

Jeffries et al.[7] compare four methods: heuristic evaluation (by 4 usability experts), software guidelines(a team of 3 software engineers), cognitive walkthrough(the same 3 software engineers), and user testing(with 6 subjects). The result shows that heuristic evaluation finds about 1/2 of the reported problems, while each of the other three methods finds about 1/6 of the problems. Heuristic evaluation and user testing are more effective at finding severe problems than the other two methods. Among the problems found by the guidelines group, only about 1/3 were found via the technique, with others found as side effect or through prior experience, conforming the ineffectiveness of guidelines method for either design or evaluation. The general conclusion is that heuristic evaluation is much more cost-effective than all the other methods (cost is measured by the time spent on evaluation in person-hours):
Heuristic evaluation(12)$\gg$Cognitive walkthrough(3)>Guidelines(2) >User testing(1)

Karat et al.[8] try to compare user testing with individual and team walkthrough methods. Their walkthrough method involves an individual user or a team of users using provided usability heuristics to go through task scenarios and report problems. The result shows that user testing(with 6 subjects) reports many more problems(both in total number of problems and in the number of severe problems) than the walkthroughs; team walkthrough(with 6 pairs of users) achieves better results than individual walkthrough(with 6 individual users) in some areas. About 1/3 of the severe problems were found by all three methods. Cost-effectiveness data show that empirical testing required the same or less time to identify each problem than walkthroughs.

Nielsen and Landauer[13] analyze the effectiveness and cost-effectiveness of user testing and heuristic evaluation, based on data from 11 studies. They model the detection of usability problems

as a function of the number of test users or heuristic evaluators as a Poisson process. The number of usability problems that have been found at least once by i test users or evaluators is

$$Found(i) = N(1 - (1 - \lambda)^i)$$

where N is the total number of problems in the interface, and the parameter λ is the probability of finding the average usability problem when running a single, average test user, or using a single, average heuristic evaluator. The data show that for user testing 16-51% of the problems can be found by one person, while for heuristic evaluation, the range is 19-60%, with 60% corresponding to evaluators with "double" (both usability and domain) expertise. This model can be used to predict the number of evaluations needed to achieve the desired levels of coverage or benefits. For a "medium" system, they estimate that 16 evaluations would be worth their cost, with benefit/cost ratio at 4.

Nielsen and Phillips[11] use GOMS, user testing, and three forms of heuristic evaluation to estimate user performance with two alternative designs for database query tasks. All methods give good estimation of the relative advantage of one design over the other. For absolute user performance, GOMS is only better than the cheapest form of heuristic evaluation, where 12 evaluators were given only a written specification of the two designs. But GOMS has much less variance among its 19 evaluators than any form of the heuristic evaluation methods. As to cost-effectiveness, for achieving a relative estimation with a standard error of 10%, the cheapest form of heuristic evaluation costs $319, with GOMS costing $560, and user testing costing $1553. These numbers are the result of adding the costs to set up the evaluation, and the costs for each test user, evaluator, with variable hourly cost of different types of staff.

2.3 Qualitative comparison of current approaches

Relating to the results from the above studies, a qualitative analysis is also necessary. User testing is expensive but is the best method for obtaining user performance data, especially those related to the user's cognitive activities like learning and understanding. User testing must be observed and analyzed by usability specialists. For instance, the nature of user's problems in using the system often needs to be analyzed through observation or interview. Heuristic evaluation is the easiest and most flexible method. But it's loosely structured both in the heuristics themselves and the way the evaluation is carried out. So its results vary largely among different evaluators, depending heavily on the evaluators' expertise both in usability and in the application domain. Empirical studies show that heuristic evaluation is more suitable for finding problems than for predicting user performance. The formal methods, GOMS and cognitive walkthrough, yield results with less variation than heuristic evaluation. But they are also more tedious to carry out. Also, since their focus is on low-level analysis, global issues are often not addressed. Another limitation of cognitive walkthrough is that it is restricted to analyzing user's learning.

In summary, theory-based approaches tend to be too tedious and hard to scale up to accommodate practically large systems. Expert analysis such as heuristic evaluation is good at finding usability problems based on usability criteria and guidelines (although some of the problems may not necessarily be experienced by users), and has the potential of being more effective when it is better structured, but is not accurate in predicting user performance and is not quite capable of

predicting user's cognitive activities. User testing often is the most dependable and most expensive way for usability evaluation, but need to be carefully planned beforehand and need to be carefully analyzed afterwards.

3 Goal/Question/Metric framework for usability evaluation

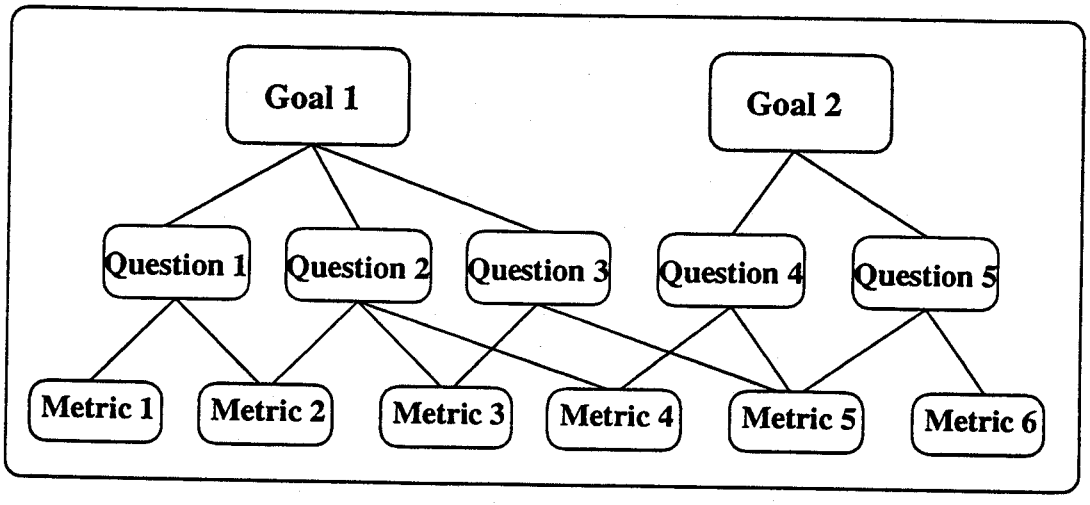

Figure 1: The GQM model

The Goal/Question/Metric (GQM) method[2][3] is a mechanism for supporting the setting of operational goals for software projects, with regard to the appropriate perspective and relevant environment. Based on models, questions are generated that define those goals as completely as possible. Measurements needed to answer those questions are specified, collected, verified and analyzed, to answer the questions and to achieve the goals.

In this work, the GQM method is used to decompose the goal of usability evaluation into questions, based on a model of human-computer interaction. We propose to decompose the main model into three sub-models along to its "use" dimension. Each sub-model describes the knowledge and activities involved in one type of use. Based on each of these three sub-models, questions and metrics are defined for usability evaluation for the corresponding use scenario.

The aim of the decomposition of the model into submodels is to divide the whole problem space into several operationally separable parts, so that each part is more manageable and is also operationally self-content. The result of the decomposition is operational scenarios. This idea of scenario-based inspection/evaluation techniques has been studied at the University of Maryland, NASA Software Engineering Lab and AT&T Bell Lab, with positive empirical results[1][16].

The resulting questions and metrics for usability evaluation(under the three operational scenarios) are given in Section 5. Data for these metrics are going to be collected. For some metrics, the data can be obtained from expert analysis of the system and the use characteristics. For other metrics, the data can only be obtained through user testing. Knowing what data need to be collected through user testing can make user testing more focused and more cost-effective. After

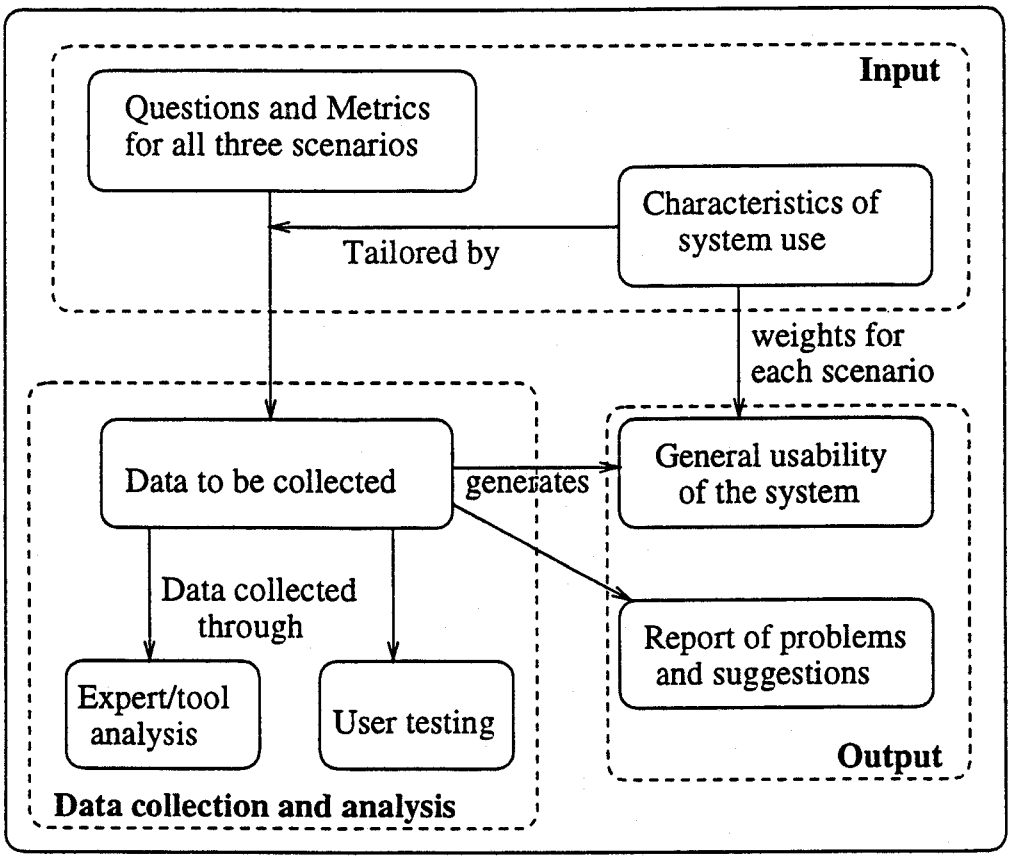

Figure 2: The framework for data collection and analysis

data for all the metrics have been collected, the general usability of the system can be assessed, according to the weights assigned to the different use scenarios for the system. The framework is illustrated in Figure 2.

4 A model for human-computer interaction

The model captures the human-computer interaction process by the following three dimensions:

Use This can be divided into three aspects: **novice using**, **error-free expert using**, and **error handling**. For a user using a computer system, each time he/she is in one of these three kind of uses. As to the five aspects of usability introduced at the beginning of this paper, novice using mostly relates to "learnability" and "memorability"; error-free expert using mostly relates to "efficiency"; error handling relates to "errors"; and all of them affects "user satisfaction".

Knowledge and skill User's nowledge is divided into **semantic** and **syntactic** knowledge[17]. Users have semantic knowledge about the task domain and the computer domain beforehand. They need to learn the syntactic knowledge of the system and establish a mapping between the objects and actions in the semantic knowledge with the objects and actions presented in

the user interface (i.e. in the syntactic knowledge). User's skill include typing skills, etc..

Activity This is divided into **execution** and **evaluation**. Norman[15] characterize users' activities during interacting with a computer into the following seven stages: (1) Forming the goal (2) Forming the intention (3) Specifying an action (4) Executing the action (5) Perceiving the system state (6) Interpreting the system state (7) Evaluating the outcome. The first four stages corresponds to execution. The rest three stages corresponding to evaluation.

According to this model, a user experiences execution and evaluation in each of the three uses, driven by semantic and syntactic knowledge, in order to achieve the goal.

To further explain this model, consider a user using a text editor to compose a document. First look at the "use" dimension of the model,

- When the user is not sure how to carry out an action on the system for the current task, or when the user is not sure what would happen after executing a certain action, or what the system response means or implies, he/she is in the state of "novice using".
- When the user knows exactly what command(s) need to be executed for the current task and what the system's response and state will be after the execution, and that the user carries out the commandes without any mistake (either on the user's side or on the system's side), he/she is experiencing "error-free expert using".
- If something goes wrong when the user is using the system, he/she enters the state of "error-handling".

As to the "knowledge and skill" dimension, for a text editing application, the user generally has semantic knowledge about the steps needed to compose a document (i.e. the tak domain), including typing/writing the text, delete some text, insert some text, go to a particular location of the document, check spelling errors, etc.. The user may or may not have the semantic knowledge of the computer domain, including the concept of a computer file and the associated "open" and "save" operations, the concept of memory and disk, the knowledge that keyboard and mouse are to be used for entering and/or selecting a command and file, typing the text, selecting and executing a command, etc.. The user may or may not know the syntactic knowledge of the system, e.g., the command for deleting a character, for going to the end of the document, for moving a block of text to another location, for spell checking, etc., as well as the way that the system gives feedbacks to the user.

Generally, in any kind of use, the user needs first to formulate the goal to achive, e.g., to eliminate all the spelling errors in the document if there is any. Then the user need to decide what he/she needs to do to achive that goal, i.e., to run the spell checker of the system. Then the user needs to map this action to the system's command by using the syntactic knowledge. Then the user executes that command. Then the user perceive the feedback from the computer, e.g., the system asking the user to confirm a correction or to select from several possible corrections. Then, with the syntactice knowledge of the system, the user undertands that the spell checker is proceeding as expected. And with the semantic knowledge, the user understands that he/she is moving toward the goal.

5 Questions, metrics, and data collection

Our goal is to collect data for software usability evaluation. The object being studies is a software in the form of product, or prototype, or design. Since our focus is on usability, we will take the user's point of view and study the use of the system by the users, for their tasks, and in their environment of use. The result of the evaluation can be used to improve a design, or to compare competitive products, or to assess the usability improvement of a product across versions, or for the purpose of quality control.

In order to come up with the set of questions and metrics for usability evaluation, the process of human-computer interaction needs to be analyzed in detail so that the relevant issues will be covered as complete as possible. To make this GQM framework operational, when defining metrics, we must consider the availability of the mechanism for data collection. So in the following, we will indicate how each piece of data can be collected, either through user testing, or expert analysis, or tool analysis, etc..

Based on the model defined in Section 4, the human-computer interaction process can be illustrated in Figure 3. This diagram indicates that generally there are four steps in the human-

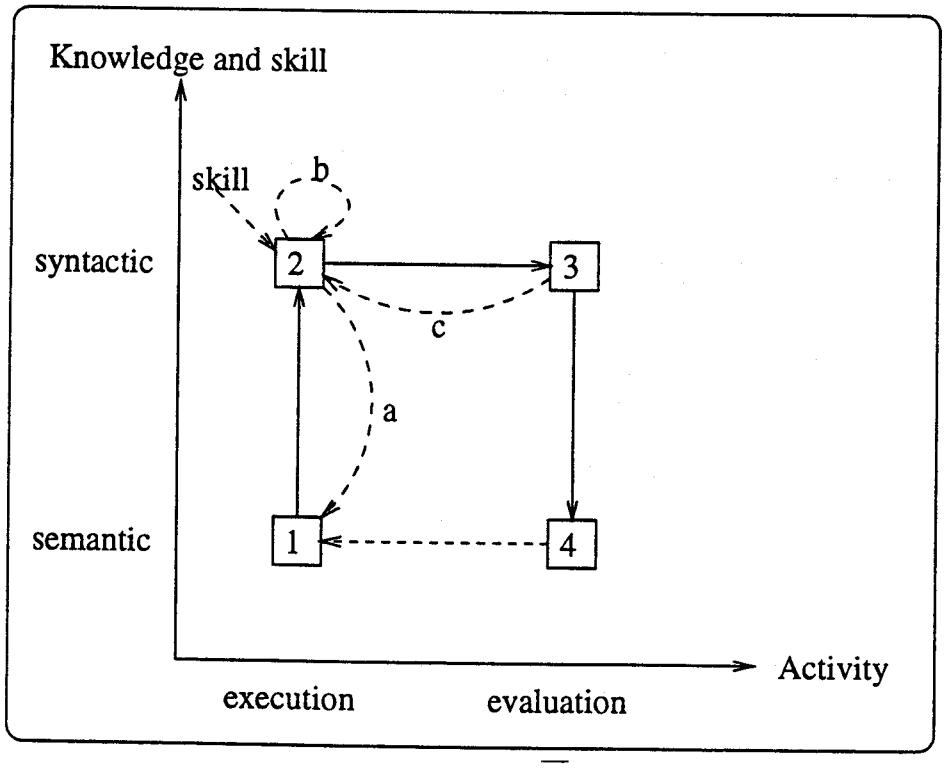

Figure 3: The human-computer interaction process

computer interaction process.

1. User uses semantic knowledge to formulate the action to be executed;
2. User uses syntactic knowledge and computer skill to execute the action;

3. User uses syntactic knowledge to perceive the action result and the system state;
4. User uses semantic knowledge to evaluate the outcome of the action.

Usually, when using a computer system to perform some task, a user will experience an iteration of the above four steps. But the steps may vary among the different kind of uses as defined in the model of Section 4. For "error-free expert using", there can be the shortcut a (as indicated in Figure 3) where the user knows exactly what the system response and action result will be so he/she can skip the "evaluation" part. Also for "error-free expert use", it is possible that after starting with step 1, the user experiences an iteration of steps 2 and 3 for a while, as indicated as shortcut c in the figure. This means that the user thinks about his task totally in the domain of the system being used. Shortcut b can be seen to represent the cases that conditions for both shortcuts a and c are true.

In the following three sections, the human-computer interaction process in each of the three use scenarios will be analyzed. The three groups of questions and metrics for usability evaluation will be generated one by one, based on the models and the analysis.

5.1 Novice using

First, a submodel for "novice using" can be derived from the main model. It can be described by the following:

- User has certain semantic knowledge in both the task domain and computer domain, both about execution and about evaluation.
- User needs to learn syntactic knowledge(of this particular system), both about execution and about evaluation, including
  - To understand objects presented by the system,
  - To understand how to perceive system responses and action results,
  - To know how to execute actions on the system.

  This can be facilitate by the easy mapping from the semantic knowledge to the syntactic knowledge, or by the use a metaphors, or by the consistency in the syntactic knowledge, as well as by the syntactic flexibility supported by the system, etc..

Based on the above submodel the questions and metrics are defined as follows (Q=Question, M=Metric).

Q1.1: How easy is it for the user to understand the system in general, including objects, actions, as well as the outcomes of actions?

The best way to answer this question is through user testing, in which each user is told to get familiar with the system, with any on-line or off-line help facilities provided, until he/she feels that he/she understands the following issues regarding the system and the tasks:

- The mapping between objects in task domain and in the system;

- The mapping between actions in task domain and in the system;
- The relationship between objects in the system;
- The way(s) to carry out an operation which involves an action with one or more objects;
- The way(s) that system responses to an operation that the user initiates;
- The way(s) to find out if the result of the operation is as expected.

When the user says he/she understands these, a test is given that covers the above issues. The two metrics relating to this user testing are:

M1.1: The time taken for a user to reach the state that he/she feels understand these aspects in general. The value is in seconds.

M1.2: The score a user gets on a test of the above understanding. The value is a percentage.

The time as in M1.1 will be adjusted by the test score as in M1.2 as follows:

Adjusted time = M1.1 / M1.2

Then the mean and standard deviation of the adjusted time will be computed.

Q1.2: To what extent can a user correctly carry out an action without having to remember the command name, object name, command or data format, etc.?

M1.3: Percentage of actions, objects, command formats and data formats that users can immediately recall or are directly visible on display when needed.

M1.4: Percentage of actions, objects, command formats and data formats that are not in M1.3 but are retrievable on display (through pull-down menu, pop-up menu, directory browsing, or on-line help).

M1.5: Percentage of actions, objects, command formats and data formats that are neither in M1.3 nor in M1.4.

M1.6: The difficulty measure of items in M1.4, corresponding the difficulty with which a user may have in finding the item.

Data for M1.3, M1.4, and M1.5 can be collected by experts, possibly with the help of tools. For M1.6, values can be assigned by experts for each of the different ways of finding the information.

All these data can also be collecting by conducting a user testing, after the users have gone through the activities corresponding to question Q1.1. Here the users are required to locate a set of objects, find out a set of commands, command formats and data formats. The users can recall them from memory, locate them on the screen, or use on-line help. Then the data for M1.3, M1.4, M1.5, and M1.6 are obtained based on the performance of the users when they work on the set of tasks for the first time. This measure of the usability can be computed as follows:

Easiness to remember and recall = $M1.3 \times 1 + M1.4 \times (mean(M1.6)) + M1.5 \times 0.1$. The ideal value is 1.

Q1.3: To what extent can a user correctly map a task into operations of the system, carry out the operations, and understand the outcome?

M1.7: The possibility that a user finds the set of system operations for the task. The value is $\frac{1}{\textit{the number of tries until success}}$.

M1.8: The possibility that a user successfully execute a step. The value is $\frac{1}{\textit{the number of tries until success}}$.

M1.9: The possibility that a user correctly understands the result of the operations. The value is $\frac{\textit{number of test users that understood correctly}}{\textit{total number of test users}}$.

These data need to be collected through user testing, based on the performance of the users when they work on the given set of tasks.

The overall value of this part of the usability is

$$\frac{\sum_{\textit{all tasks}}(M1.7 \times M1.9 \times \prod_{\textit{all steps for the task}} M1.8)}{\textit{the number of tasks}}$$

The ideal value is 1.

Q1.4: How flexible is the system on the order of command parameters or input data that are semantically orderless?

M1.10: All the cases that semantically it is orderless.

M1.11: The cases of M.10 that flexibility is supported by the system.

These data can be collected by experts by going through the objects and actions of the system. It can also being collected with cooperation with the people in charge of testing the systems.

Q1.5: How consistent is the system in terms of objects, actions, system responses, etc.?

M1.12: To what extent are objects consistently represented, including their icon, color, position, size, etc.?

M1.13: To what extent can actions be carried out in a consistent way, e.g., through command line, menu selection, or direct manipulation?

M1.14: To what extent are the system responses given consistently, including position, format, etc.?

These data can be collected by experts by going through the objects and actions of the system. It can also being collected with cooperation with the people in charge of testing the systems.

5.2 Error-free expert using

The usability of a system in this scenario means the extent to which the system facilitate the efficient and satisfying use of the system through the user interface design[1].

In this use scenario,

[1] Efficiency and user satisfaction are also affected to a large extent by the design of the algorithm and the capacity of the underlying computer hardware. In usability evaluation we only analyze the way that efficiency and user satisfaction are affected by the design of the user interface. For example, if a transaction takes a long time (say, longer than 2 seconds), it would be more satisfying to the user if the system constantly informs the user about the proceeding of the transaction as well as the portion that has been finished.

- User have good syntactic knowledge of the system for the current task(s);
- User may demand more efficient and satisfying operation of the system, which can be achieved by
  - Facilities like shortcut, macro, default values, etc. that can let the user achieve the same goal with less work,
  - Less stress on the user side,
  - More appropriate feedbacks,
  - More customizable user interface.

The questions and metrics for usability evaluation under this scenario is as follows.

Q2.1: How efficient is the system for the defined tasks?

M2.1: The time taken for completing each task in the set with the developer-specified most efficient method. This should include the times for mental planning, execution of operations, waiting for system response, and understanding the outcome.

M2.2: The least number of operations for each task in the set. Ideal value is 1.

M2.3: The cases that a default value can be provided.

M2.4: The cases in M2.3 that default values are provided by the system.

Data for M2.1, M2.2, M2.3, M2.4 can all be collected by expert analysis and trial. M2.1 can also be collected by conducting user testing. For metrics M2.3 and M2.4, the ideal situation is M2.4 = M2.3, meaning that default values are provided whenever possible.

Q2.2: How easy is it to use the system for the tasks?

M2.5: The difficulty index of each action. The value is at least 1. Actions (either execution or evaluation) with difficulty index greater than 1 include:

- Double click a mouse button: 1.2
- Hold a mouse button for longer than 1 second: the time in seconds
- Combination keys that can be carried out with one hand: 1.4
- Combination keys or combined use of mouse and keyboard that must be carried out with two hands: 2
- Read blinking text, text in small fonts, or text in hard-to-read colors: 1.2 – 2

These difficulty index values in M2.2 can be decided by the evaluators, but should be used consistently for the purpose of comparison. There can be other actions that cause extra stress on the user. The important thing is to record all the types of "difficult actions" and the difficulty index value used for reference.

The overall value for Q2.2 is

$$\frac{\sum_{all\ actions} M2.5}{the\ number\ of\ actions}$$

The ideal value is 1.

Q2.3: How appropriate are the system feedbacks presented?

M2.6: The number of non-error short system messages for simple confirmation that should not require further user input.

M2.7: The number of system messages in M2.6 that require user reaction on the messages themselves. The ideal value is 0.

M2.8: The number of cases that a long-time (longer than 2 seconds) operation is going on.

M2.9: The number of cases in M2.8 that a constant message shows the percentage of work having been done for that operation. The ideal value is 0.

Data for M2.6, M2.7, M2.8, and M2.9 can be collected in cooperation with people in charge of testing the software. For each case in M2.7 and M2.9, a recommendation for improvement should be added to the evaluation report.

Q2.4: How customizable is the user interface of the system?

M2.10: Number of parameters that users want to customize.

M2.11: The parameters in 2.10 that are made customizable in the user-desired way in the system. Ideally M2.10 = M2.11.

The list of parameters can be collected by the usability experts in cooperation with the designers/developers. Then data for M2.10 and M2.11 can be collected through a users survey (among users having at least tried the system).

5.3 Error handling

When an error occurs, it means that the following has happened:

1. User wanted to do something;
2. User executed some command(s);
3. One of the following happened:
   (a) The command(s) executed successfully, but the result was not as the user had expected;
   (b) The last command failed to execute, possibly generating some error message;
   (c) The system crashed.

In terms of error handling, a usable system should be:

- Error preventive. The best way for error handling is to prevent them.
- Error informative. When an error occurs, inform the user what error has occurred, as specific and context-sensitive as possible, and in the user-understandable language.
- Error corrective. Provide facilities for the users to recover from errors, and tell them how to do that when errors occur.

Questions for usability evaluation under this scenario are as follows:

Q3.1: What's the error rate of using the system.

M3.1: The number of errors a user made when working on the set of tasks.

M3.2: The time a user takes to recover from errors when working on the set of tasks.

Data for M3.1 and M3.2 are collected based on the users working on the set of tasks during all kinds user testings, either for novice using, or for expert using.

The ideal values for M3.1 and M3.2 are 0.

Q3.2: To what extent is the system error-preventive?

M3.3: The percentage of invalid operations/commands disabled.

The data for M3.3 can be collected in cooperation with the people in charge of testing the system.

Q3.3: To what extent is the system error-informative?

M3.4: The extent to which the user understands what the error is.

M3.5: The extent to which the user understands what to do to recover from the error.

The data for M3.4 and M3.5 should be collected based on user's experience in dealing with errors during all the user testings.

Q3.4: To what extent is the system error-corrective?

M3.6: The interval that the user's work is saved periodically in case the system crashes?

M3.7: When the system restarts after crashing, whether the system inform the user how to retrieve the most recent backup when it exists?

The data for M3.6 and M3.7 can be obtained by analyze the system or consulting the designers/developers of the system.

5.4 Summary of data collection

As shown above, usability data need to be collected through the analysis of usability specialists, through the use of analysis tools, through user testing, and through the cooperation with people who test the system in general. But before all these, it is important to analyze the characteristics of the system, the users, the tasks, and the environment, to make it clear what usability data need to be collected for the current project, and how to collect them.

The parts that user testing is needed the most are those relating to the cognitive process of human users, such as in the "novice using" scenario and in checking the understandability of error message, etc.. So far, user testing is still the most dependable method for assessing the usability for these aspects. The cognitive walkthrough approach introduced in Section 2.1 has not been shown successful in dealing with general human-computer interactions other than the simple "walk-up-and-use" systems. In case user testing cannot be conducted, the heuristic evaluation method (as one "discount usability engineering" method) can be used where usability experts use

certain usability heuristics and see if user will have usability problem when using the system. But this kind of heuristic analysis is hard to be accurate, is less objective, and depends heavily on the experts understanding of the user and task characteristics.

It is also important to collect qualitative information of user testing. For user testing, current usability labs often provides the facility for observing the user's interaction process unobtrusively, and for recording the whole process on videotape, sometimes with multiple views. This provides the great opportunity for deep analysis of user testing data. Since the quantitative data alone, like task completion time, may not be informative enough for a detailed analysis of how much time is spent on which part of the interaction process.

Another reason for analysis beyond quantitative data is the importance of one often neglected aspect of data, their context. Context information can be obtained through observing or watching the videotape of user testing. Context information that may affect the interpretation of test results should all be reported. Otherwise the data can be misleading. More importantly, the detailed information about users' interaction process, especially what kind of problems users had with the system during the process, is more valuable for the development team to make changes to the system for usability improvement, although quantitative evaluation is often carried out from the organization's point of view for comparison and decision-making.

6 Organizational strategy for data interpretation

Two kinds of output can be generated from the data collection process: quantitative usability measures and qualitative information about usability problems in the system. Quantitative evaluation in form of an assigned value of the usability is quite informative for understanding the general usability of the whole system. The qualitative outputs, especially the ones pointing out specifically where users experience usability problems, are more significant for later work on usability improvement.

To draw conclusions from the collected data, people need to compare them with the ideal values of the corresponding metrics provided, with data from competitive systems, or with data from older systems. Some metrics have ideal values associated with them. For the metrics that do not have ideal values, other baseline values are needed to make the evaluation meaningful. For this reason we suggest that each organization collect and organize usability data of all its products, as well as usability data of competitive products from other organizations. These data can be used for different kinds of decision making like quality control and business strategy decisions.

References

[1] Victor Basili, Scott Green, Oliver Laitenberger, Forrest Shull, Sivert Sorumgard, and Marvin Zelkowitz. The empirical investigation of perspective-based reading. *Experimental Software Engineering*, Spring 1996.

[2] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. Goal Question Metric paradigm. In *Encyclopedia of Software Engineering*, pages 528–532. John Wiley & Sons, 1994.

[3] Victor R. Basili and H. Dieter Rombach. The TAME project: Towards improvement-oriented software environment. *IEEE Transaction on Software Engineering*, 14(6):758–773, June 1988.

[4] Randolph G. Bias. The pluralistic usability walkthrough: Coordinated empathies. In Jakob Nielsen and Robert L. Mack, editors, *Usability Inspection Methods*, chapter 3, pages 63–76. John Wiley, 1994.

[5] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction.* Laurence Erlbaum Associates, 1983.

[6] Joseph S. Dumas and Janice C. Redish. *A Practical Guide to Usability Testing.* Ablex Publishing Corporation, Norwood, New Jersey, 1993.

[7] R. Jeffries, J. R. Miller, C. Wharton, and K. M. Uyeda. User interface evaluation in the real world: A comparison of four methods. In *Human Factors in Computing Systems, CHI'91 Conference Proceedings*, pages 261–266. ACM, 1991.

[8] C.-M. Karat, R. Campbell, and T. Fiegel. Comparison of empirical testing and walkthrough methods in user interface evaluation. In *CHI'92 Conference Proceedings*, pages 397–404. ACM, 1992.

[9] David E. Kieras. Towards a practical GOMS model methodology for user interface design. In J. Psotks, L. D. Massey, and S. Mutter, editors, *The handbook of human-computer interaction*, pages 135–158. North-Holland, Amsterdam, 1988.

[10] Thomas K. Landauer. *The Trouble with Computers: Usefulness, Usability, and Productivity.* The MIT Press, 1995.

[11] Jacob Nielsen and Victoria L. Phillips. Estimating the relative usability of two interfaces: Heuristic, formal, and empirical methods compared. In *Interchi'93 Conference Proceedings*, pages 214–221. ACM, April 1993.

[12] Jakob Nielsen. Heuristic evaluation. In Jokob Nielsen and Robert Mack, editors, *Usability Inspection Methods*, chapter 2, pages 25–62. John Wiley, 1994.

[13] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Interchi'93 Conference Proceedings*, pages 206–213. ACM, April 1993.

[14] Jokob Nielsen. *Usability Engineering.* Academic Press, Inc., San Diego, California, 1993.

[15] Donald A. Norman. *The Design of Everyday Things.* Basic Books, New York, 1st doubleday/currency edition, 1988.

[16] Adam A. Porter, Lawrence G. Votta, Jr., and Victor R. Basili. Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transaction on Software Engineering*, June 1995.

[17] Ben Shneiderman. *Designing the User Interface : Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 2nd edition, 1992.

[18] M. Sweeney, M. Maguire, and B. Shackel. Evaluating user-computer interaction: a framework. *International Journal of Man-Machine Studies*, 38:689–711, 1993.

[19] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. The cognitive walkthrough method: A practitioner's guide. In Jakob Nielsen and Robert L. Mack, editors, *Usability Inspection Methods*, chapter 5, pages 105–140. John Wiley & Sons, Inc., 1994.