Usability Basics for Software Developers

Xavier Ferré and Natalia Juristo, *Universidad Politécnica de Madrid*
Helmut Windl, *Siemens AG, Germany*
Larry Constantine, *Constantine & Lockwood*

In recent years, software system usability has made some interesting advances, with more and more organizations starting to take usability seriously.[1] Unfortunately, the average developer has not adopted these new concepts, so the usability level of software products has not improved.

This tutorial examines the relationship between usability and the user interface and discusses how the usability process follows a design–evaluate–redesign cycle. It also discusses some management issues an organization must face when applying usability techniques.

Contrary to what some might think, usability is not just the appearance of the user interface (UI). Usability relates to how the system interacts with the user, and it includes five basic attributes: learnability, efficiency, user retention over time, error rate, and satisfaction. Here, we present the general usability process for building a system with the desired level of usability. This process, which most usability practitioners apply with slight variation, is structured around a design-evaluate-redesign cycle. Practitioners initiate the process by analyzing the targeted users and the tasks those users will perform.

Clarifying usability concepts

According to ISO 9241, Part 11, usability is "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."[2] This definition ties a system's usability to specific conditions, needs, and users—it requires establishing certain levels of usability based on the five basic attributes.

Usability engineering defines the target usability level in advance and ensures that the software developed reaches that level. The term was coined to reflect the engineering approach some usability specialists take.[3] It is "a process through which usability characteristics are specified, quantitatively and early in the development process, and measured throughout the process."[4] Usability is an issue we can approach from multiple viewpoints, which is why many different disciplines, such as psychology, computer science, and sociology, are trying to tackle it. Unfortunately, this results in a lack of standard terminology. In fact, the term usability engineering is not universally accepted—other terms used include usage-centered design, contextual design, participatory design, and goal-directed design. All these philosophies adhere to some extent to the core issue of usability engineering: evaluating usability with real users from the first stages of development.

Usability attributes

We can't define usability as a specific as-

0740-7459/00/$10.00 © 2001 IEEE

pect of a system. It differs depending on the intended use of the system under development. For example, a museum kiosk must run a software system that requires minimum training, as the majority of users will use it just once in their lifetime. Some aspects of usability—such as efficiency (the number of tasks per hour)—are irrelevant for this kind of system, but ease of learning is critical. However, a bank cashier's system would require training and would need to be highly efficient to help reduce customer queuing time.

Because usability is too abstract a term to study directly, it is usually divided into the attributes we mentioned at the beginning of the article:[5]

- *Learnability*: How easy it is to learn the main system functionality and gain proficiency to complete the job. We usually assess this by measuring the time a user spends working with the system before that user can complete certain tasks in the time it would take an expert to complete the same tasks. This attribute is very important for novice users.
- *Efficiency*: The number of tasks per unit of time that the user can perform using the system. We look for the maximum speed of user task performance. The higher system usability is, the faster the user can perform the task and complete the job.
- *User retention over time*: It is critical for intermittent users to be able to use the system without having to climb the learning curve again. This attribute reflects how well the user remembers how the system works after a period of nonusage.
- *Error rate*: This attribute contributes negatively to usability. It does not refer to system errors. On the contrary, it addresses the number of errors the user makes while performing a task. Good usability implies a low error rate. Errors reduce efficiency and user satisfaction, and they can be seen as a failure to communicate to the user the right way of doing things.
- *Satisfaction*: This shows a user's subjective impression of the system.

One problem concerning usability is that these attributes sometimes conflict. For example, learnability and efficiency usually influence each other negatively. A system must be carefully designed if it requires both high learnability and high efficiency—for example, using accelerators (a combination of keys to perform a frequent task) usually solves this conflict. The point is that a system's usability is not merely the sum of these attributes' values; it is defined as reaching a certain level for each attribute.

We can further divide these attributes to precisely address the aspects of usability in which we are most interested. For example, *performance in normal use* and *advanced feature usage* are both subattributes of efficiency, and *first impression* is a subattribute of satisfaction. Therefore, when analyzing a particular system's usability, we decompose the most important usability attributes down to the right detail level.

Usability is not only concerned with software interaction. It is also concerned with help features, user documentation, and installation instructions.

Usability and the user interface

We distinguish between the visible part of the UI (buttons, pull-down menus, checkboxes, background color, and so forth) and the interaction part of the system to understand the depth and scope of a system's usability. (By interaction, we mean the coordination of the information exchange between the user and the system.) It's important to carefully consider the interaction not just when designing the visible part of the UI, but also when designing the rest of the system.

For example, if a system must provide continuous feedback to the user, the developers need to consider this when designing the time-consuming system operations. They should design the system so it can frequently send information to the UI to keep the user informed about the operation's current status. The system could display this information as a percentage-completed bar, as in some software installation programs.

Unfortunately, it is not unusual to find development teams that think they can design the system and then have the "usability team" make it usable by designing a nice set of controls, adding the right color combination, and using the right font. This approach is clearly wrong. Developers must consider user interaction from the beginning of the development process. Their understanding of the interaction will affect the final product's usability.

It's important to carefully consider the interaction not just when designing the visible part of the user interface, but also when designing the rest of the system.

Usability testing alone is not enough to output a highly usable product, because it uncovers but does not fix design problems.

Usability in software development

The main reason for applying usability techniques when developing a software system is to increase user efficiency and satisfaction and, consequently, productivity. Usability techniques, therefore, can help any software system reach its goal by helping the users perform their tasks. Furthermore, good usability is gaining importance in a world in which users are less computer literate and can't afford to spend a long time learning how a system works. Usability is critical for user system acceptance: If users don't think the system will help them perform their tasks, they are less likely to accept it. It's possible they won't use the system at all or will use it inefficiently after deployment. If we don't properly support the user task, we are not meeting user needs and are missing the main objective of building a software system.

For a software development organization operating in a competitive market, failure to address usability can lead to a loss of market share should a competitor release a product with higher usability. Also, a software product with better usability will result in reduced support costs (in terms of hotlines, customer support service, and so forth).

Even if a system is being used, it does not necessarily mean it has a high level of usability. There are other aspects of a software product that condition its usage, such as price, possibility of choice, or previous training. In addition, because users are still more intelligent than computers, it is usually the human who adapts to the computer in human–computer interaction. However, we shouldn't force the user to adapt to software with poor usability, because this adaptation can negatively influence efficiency, effectiveness, and satisfaction. Usability is a key aspect of a software product's success.

The usability process

As we mentioned, a system's usability depends on the interaction design. Therefore, we must deal with system usability throughout the entire development process. Usability testing alone is not enough to output a highly usable product, because usability testing uncovers but does not fix design problems. Furthermore, usability testing has been viewed as similar to other types of software quality assurance testing, so developers often apply the techniques late in the development cycle—when major usability problems are very costly, if not impossible, to fix. Therefore, it is crucial to evaluate all results during the product development process, which ultimately leads to an iterative development process. A pure waterfall approach to software development makes introducing usability techniques fairly impossible.

All software applications are tools that help users accomplish certain tasks. However, before we can build usable software tools—or, rather, design a UI—we need information about the people who will use the tool:

- Who are the system users?
- What will they need to accomplish?
- What will they need from the system to accomplish this?
- How should the system supply what they need?

The usability process helps user interaction designers answer these questions during the analysis phase and supports the design in the design phase (see Figure 1).

There are many usability methods—all essentially based on the same usability process—so we have abstracted a generic usability process from the different approaches to usability mentioned earlier. We hope this makes it easier for the reader to understand the different usability techniques we will be describing.

Usability analysis phase

First, we have to get to know the users and their needs, expectations, interests, behaviors, and responsibilities, all of which characterize their relationship with the system.

User analysis. There are numerous approaches for gathering information about users, depending on each individual system under development and the effort or time constraints for this phase. The main methods are *site visits*, *focus groups*, *surveys*, and *derived data*.

The primary source for user information is site visits. Developers observe the users in their working environment, using the system to be replaced or performing their tasks manually if there is no existing tool. In addition, developers interview the users to understand their motivation and the strategy behind their actions. A well-known method