



International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Usability Issues in Software Development Lifecycle

R.Madhavan*

Department Of CSE,
Joesuresh Engineering College, Tamilnadu, India

Dr.K.Alagarsamy

Computer Centre
Maduraikamaraj University, Tamilnadu, India

Abstract: Usability is one of the key quality attributes in software development. It is a factor present in almost all software quality assurance models. There is a close analogy between the terms usability and quality. Usability is an important quality factor for interactive software systems. The usability attributes which contribute to quality of use will include the style and properties of the user interface, the dialogue structure, and the nature of the functionality. The aim of this paper is to improve software quality by finding ways to integrate usability with software quality measurements throughout the life cycle.

Keywords: User requirements, design, software architecture, SDLC, software quality.

I. Introduction

Usability is the extent to which specific users can use a product to their satisfaction in order to effectively and efficiently achieve specific goals in a specific context of use. A number of studies have pointed out a wide range of benefits stemming from usability: it improves productivity and raises team morale, reduces training and documentation costs, improves user productivity, increases e-commerce potential, etc. Additionally, and contrary to what one might think, the cost/benefit ratio of usability is highly worthwhile. Some authors have state that every dollar spent on usability offers a return of \$30.25. There are also studies for e-commerce sites that show that a 5% improvement in usability could increase revenues by 10% to 35%. Accordingly, usability is moving up the list of strategic factors to be dealt with in software development. In spite of its relevance usability is still insufficient in most software systems. The most widespread view in SE is that usability is chiefly related to the user interface. Therefore, the good design practice that separates the UI from the core functionality would be sufficient to support the development of a usable software system. This view implies that dealing with usability could be put off until the later stages of software system development (commonly during system evaluation), as the changes required to improve this quality attribute should not involve a lot of rework. On the contrary, usability places static and dynamic constraints on several software components. Separating the user interface from the core functionality is not sufficient to support usability and other tactics must be put in place in the software development if a usable product is to be delivered to the customer. The implications of the usability features for design materialize as the creation of special items (components, responsibilities, interactions, classes, methods, etc.) that affect both the presentation and application layer of the software system architecture. Therefore, addressing usability features at the end of the construction process will involve major rework. To avoid this, software usability should be dealt with proactively at requirements and design time instead of retroactively after testing.

II. Usability Basics

Usability refers to how well users can learn and use a product to achieve their goals. It also refers to how satisfied users are with that process.

Usability measures the quality of a user's experience when interacting with a product or system, including:

- Websites
- Software applications
- Mobile technologies
- Any user-operated device

It is important to realize that usability is not a single, one-dimensional property of a user interface. Usability is a combination of factors including:

- Intuitive design: a nearly effortless understanding of the architecture and navigation of the site
- Ease of learning: how fast a user who has never seen the user interface before can accomplish basic tasks
- Efficiency of use: How fast an experienced user can accomplish tasks
- Memorability: after visiting the site, if a user can remember enough to use it effectively in future visits
- Error frequency and severity: how often users make errors while using the system, how serious the errors are, and how users recover from the errors

- Subjective satisfaction: If the user likes using the system

III. Importance of Usability

It's impossible to list off of the different types of usability issues that may need to be addressed with a specific product, especially since not all measures apply to all products. But for an information system (at least for software or a website) issues such as platform compatibility, consistency, content wording, error prevention and help features, navigation, feedback, security, text readability, and the use of color all fall under the usability "umbrella" and are all fair game for a Usability Expert to evaluate. Nevertheless, the quick list of example questions asked above should be a good indicator of why good usability is so important -- it can make the difference between a successful software install or an unwanted hard drive "cleaning," a safe plane ride or a disaster, a perfect piece of plain white toast or a char-broiled square of something that used to be Wonder Bread. It's the difference between effectively helping users to accurately complete their tasks and frustrating them because they can't do what they need to do. It affects users, developers, and managers - if you don't take care of usability, people won't buy your software, fly your plane, use your product, or surf your website.

IV. User requirements analysis

Poorly specified user requirements are one of the most significant factors behind IT project failure. A successful product or system requires a proper understanding of both user and organisational requirements.

Typical requirements gathering and analysis methods include:

- *Surveys* – both open-ended and focused surveys, conducted electronically or on paper.
- *Interviews* – typically conducted face-to-face, but also over the telephone if deemed more appropriate.
- *Focus groups* – whilst being poorly suited to evaluating a product, focus groups are useful for discussing possible user requirements and brainstorming ideas.
- *Field studies* – observing the end-user situation and the environment in which a new product or system will be used is often extremely useful in understanding user needs.
- *Evaluation of an existing product* – provides a range of useful information (even competitor products can be tested). Usability evaluation reveals and clarifies good and bad aspects of current solutions - valuable input to new design work.
- *Task analysis* – a deeper analysis of users work with a system, useful for analysing how user's work tasks should be supported by functionality in a system.
- *User personas and usage scenarios* – concrete and illustrative data about typical users, their characteristics, usage situation, tasks and goals. Particularly useful in supporting early user interface design work.
- *Formulation of usability goals and overall design criteria* – help focus and steer the design process, supporting the evaluation of early concepts, prototypes and final designs.

V. Usability and software design

Designers make an impact analysis of user data collected during evolutionary delivery to estimate the effectiveness of design techniques in meeting product goals. In usability engineering, design techniques are usually ideas developed after watching people use computer systems. Estimating the effectiveness of a set of design techniques for meeting a set of usability attributes helps to economically focus engineering effort on key issues. Impact analysis tables contain percentage estimates of the contribution of each technique to the planned levels for each usability attribute. Impact analysis tables list product attributes and proposed design techniques in a matrix. Each entry in the table estimates the percentage that this technique will contribute toward meeting the planned level of this attribute. To design products that satisfy their target users, a deeper understanding is needed of their user characteristics and product properties in development related to unexpected problems users face. These user characteristics encompass cognitive aspect, personality, demographics, and use behavior. The product properties represent operational transparency, interaction density, product importance, frequency of use and so on.

VI. Usability and Software Architecture

A software architecture description such as a decomposition of the system into components and relations with its environment may provide information on the support for particular quality attributes. Specific relationships between software architecture (such as - styles, -patterns etc) and quality attributes (maintainability, efficiency) have been described by several authors. For example the architectural pattern layers and the positive effect this pattern may have on exchangeability and the negative effect it may have on efficiency

A. Usability Patterns

A number of usability patterns have been identified that should be applied during the design of a system's software architecture, rather than during the detailed design stage. This set of patterns has been identified from various cases in industry, modern software, literature surveys as well as from existing (usability) pattern collections.

Some examples:

- Actions on multiple objects - Actions need to be performed on objects, and users are likely to want to perform these actions on two or more objects at one time.
- Multiple views - The same data and commands must be potentially presented using different human-computer interface styles for different user preferences, needs or disabilities.
- User profiles - The application will be used by users with differing abilities, cultures, and tastes.

Unlike the design patterns, architecturally sensitive patterns do not specify a specific design solution in terms of objects and classes. Instead, potential architectural implications that face developers looking to solve the problem the architecturally sensitive pattern represents are outlined. For example, to facilitate actions on multiple objects, a provision needs to be made in the architecture for objects to be grouped into composites, or for it to be possible to iterate over a set of objects performing the same action for each. Actions for multiple objects may be implemented by the composite pattern or the visitor pattern.

(Positive) relationships have been defined between the elements of the framework that link architectural sensitive usability patterns to usability properties and attributes. These relationships have been derived from our literature survey. The usability properties in the framework may be used as requirements during design. For example, if a requirements species, "the system must provide feedback", we use the framework to identify which usability patterns may be implemented to fulfill these properties by following the arrows in Figure 1. Our assessment technique uses this framework to analyze the architecture's support for usability.

VII. Usage profiles

A usage profile represents the required usability of the system by means of a set of usage scenarios. Usability is not an intrinsic quality of the system. According to the ISO definition, usability depends on:

- The users - who is using the product? (System administrators, novice users)
- The tasks - what are the users trying to do with the product? (Insert order, search for item X)
- The context of use - where and how is the product used? (Helpdesk, training environment)

Usability may also depend on other variables, such as goals of use, etc. However in a usage scenario only the variables stated above are included. A usage scenario is defined as "an interaction (task) between users, the system in a specific context of use". A usage scenario specified in such a way does not yet specify anything about the required usability of the system. In order to do that, the usage scenario is related to the four usability attributes defined in our framework. For each usage scenario, numeric values are determined for each of these usability attributes. The numeric values are used to determine a prioritization between the usability attributes. For some usability attributes, such as efficiency and learnability, tradeoffs have to be made. It is often impossible to design a system that has high scores on all attributes. A purpose of usability requirements is therefore to specify a necessary level for each attribute. For example, if for a particular usage scenario learnability is considered to be of more importance than other usability attributes (maybe because of a requirement), then the usage scenario must reflect this difference in the priorities for the usability attributes. The analyst interprets the priority values during the analysis phase to determine the level of support in the software architecture for the usage scenario.

VIII. Quality Factors for Usability

Some quality factors are very important when performing usability testing. usability is subjective and not all requirements for usability can be documented clearly. However focusing on some of the quality factors given below help in improving objectivity in usability testing are as follows.

A. Comprehensibility The product should have simple and logical structure of features and documentation. They should be grouped on the basis of user scenarios and usage. The most frequent operations that are performed early in a scenario should be presented first, using the user interfaces. When features and components are grouped in a product, they should be based on user terminologies, not technology or implementation.

B. Consistency A product needs to be consistent with any applicable standards, platform look-and-feel, base infrastructure, and earlier versions of the same product. Also, if there are multiple products from the same company, it would be worthwhile to have some consistency in the look-and-feel of these multiple products. User interfaces that are different in different operating systems underlying, services irritates the users since they need to become comfortable with different templates and procedures for using each of them. For example, if an operating system uses a template for error message with a set of icons, error number, error message and link to the related documentation, following the same format will make users more comfortable with a product on that operating system. Unless there is a major problem with the user interface of a product, the current interface and usage should be consistent with the earlier versions of the same product. Following some standards for usability helps in meeting the consistency aspect of the usability.

C. Navigation This helps in determining how easy it is to select the different operations of the product. An option that is buried very deep requires the user to travel to multiple screens or menu options to perform the operation. The number of mouse clicks, or menu navigations that is required to perform an operation should be minimized to improve usability. When users get stuck or get lost, there should be an easy option to abort or go back to the previous screen or to the main menu so that the user can try a different route.

D. Responsiveness How fast the product responds to the user request is another important aspect of usability? This should not be confused with performance testing. Screen navigations and visual displays should be almost immediate after the user

selects an option or else it could give an impression to the user that there is no progress and cause him or her to keep trying the operation again. Whenever the product is processing some information, the visual display should indicate the progress and also the amount of time left so that the users can wait patiently till the operation is completed. Adequate dialogs and popups to guide the users also improve usability. Responsiveness should not be mistaken for too many popups and dialog boxes. Too many notifications within a short interval works only to reduce usability and may, in fact, slow down responsiveness.

IX. Technology & usability

Technologies play a key role for usability trends and implementation. Users' expectation changes with technologies - as more sophisticated technologies become available, user expectation increases. The following section discusses some common technology enablers for software usability:

A. Templates

Consistency is one of the key quality attributes of usability. Templates play a large role in creating consistent looking applications/products, helping developers to choose and adopt a common information flow across the product.

Website templates are very affordable and they save you a lot of effort and time when you want to create a new layout for your application. Many technologies support template driven framework and it can be effective to bring consistency to the product. To avoid the risk of adopting the wrong template though, it is essential to do proper information architecture for your product - understand the domain, target audience and typical usage of the product. Using this information architecture as a base, the flow of information can be defined and accordingly a template chosen.

B. Cascading Style Sheets

Cascading Style Sheets is the holy grail for the 'look and feel' of software. Style sheets have brought in many possibilities including the way to isolate the presentation styles from the content. CSS offers endless possibilities of customization by the end user for the same application or product. It is a great way to isolate the presentation styles till the final delivery without impacting the business functionalities. This allows the design team to work on better presentation styles even as the product is getting built. It also offers customers the ability to customize software to match their corporate or personal style.

C. Rich User Experience tools/technologies

Speed, interactivity and collaboration are some of the newer requirements of web applications. Some of the technologies that make these possible are Ajax, DOM, and RIA and so on. Ajax and DOM makes dynamic updates to websites quick by only allowing a select portion of the system (with required info) to be updated so that entire pages need not be refreshed. This reduces data download from web servers and makes the whole process speedy. All browsers support Ajax implementation; using third party Ajax libraries usually abstracts the difference in Ajax library implementation and gives standard interfaces. Adopting Web 2.0 attributes to any software, be it enterprise or consumer, can improve the aesthetics, facilitate better communication and information sharing, interoperability and collaboration. Today, various technology stacks supports Web 2.0 attributes and choosing the proper one can make the user experience better. Adding relevant wikis, forums and discussion threads where needed can drive collective collaboration and participation.

One of the technologies that have evolved over course of time is JavaScript – which is now suited to today's interactive applications on the web. The frameworks created in JavaScript promote reusability and sophisticated ones like EXTJS, JQuery, etc provide a good level of abstraction for RIA capabilities, without much complexity involved, and are portable across browsers.

X. Usability in the Software lifecycle

From a user-centered perspective, usability evaluations should start early in the development process and occur repeatedly throughout the design cycle, not just when the product is completed.

Usability testing methods can be adapted for any of the general phases detailed below in the software or technology product development process. Usability evaluation at each phase is a critical part of ensuring that the product will actually be used and be effective for its intended purpose(s).

Phase 1 – Concept Exploration

Processes:

- Ideas for the software/technology product to help users solve problems or accomplish goals are developed.
- What the product needs to do (requirements analysis) and how the product will be used (task analysis) are explored.
- Platform capabilities and constraints are considered.
- The Usability focus is on getting the product to work.

Usability evaluations in Phase 1 can yield information about:

- the need for the product
- the needs for particular functions/features of the product.

Phase 2-Demonstration and validation

Processes:

- The concept is implemented.
- Product requirements and constraints are refined and prioritized in order to identify or select those elements critical to the prototype.

- The key market is defined.
- The Usability focus is on having the right functions.

Usability evaluations in Phase 2 can yield evidence of the success or failure of the design concept.

Phase 3 – Detailed design and construction

Processes:

- The required software elements are defined and integrated into the prototype.
- Further context analysis is done to determine if any significant changes have occurred in the users' workplace since Phase 1.

Usability evaluations in Phase 3 can be used to determine whether or not the software is ready to go into production, or if further development and testing is needed.

Phase 4 –Production and Operation

Processes:

- Plans for integrating and testing the product in the "real world" environments for which it is intended are developed.
- Plans for "final" product delivery and installation are developed.
- Product maintenance and support plans are developed.
- Processes for disposing of the product when it is no longer viable are defined.

Usability evaluations in Phase 4:

- Can guide product revisions, marketing and delivery plans, and the support services required.
- Lead to product updates that require another round of usability evaluation thus repeating the usability engineering life cycle. At every stage of the software product life cycle, usability is critical.

XI. Conclusion

Usability is finally getting the attention it deserves, from the software community at large, as it becomes essential for the growth and survival of software products in a competitive environment. Better the usability of software, better are its chances of success. Usability has indeed become a true differentiator for software products and wise product managers will ensure they get the best interaction and usability engineering teams to work on their product.

References

- [1] B. Myers and M.B. Rosson, Survey on User Interface Programming, *Proc. CHI'92, ACM Press*, 195-202.
- [2] J. Nielsen. *Usability Engineering*, Academic Press, Cambridge, MA.
- [3] Folmer, E. & Bosch, J. (2003). Architecting for usability; a survey
- [4] Stoll, P., Bass, L., Golden, E., & John, B. E. (2009) Supporting usability in product line architectures. *Proceedings of the 13th International Software Product Line Conference* (San Francisco, CA, August 24-28, 2009).
- [5] Golden, E., John, B. E., Bass, L. (2005) Quality vs. quantity: Comparing evaluation methods in a usability-focused software architecture modification task. *Proceedings of the 4th International Symposium on Empirical Software Engineering* (Noosa Heads, Australia, November 17-18 2005).