

Usability Engineering
Dr. Debayan Dhar
Department of Design
Indian Institute of Technology, Guwahati


Module - 02
Lecture - 05
Usability In Software Development

Welcome to lecture 5 and the first lecture for module 2. In this lecture or in this module we are going to discuss about Usability in Software Development. It is essential for us to understand the emergence of software engineering, the emergence of software development, issues behind the various techniques and models that were that emerged during those early periods of software engineering. And how those issues became critical in order for usability to be considered as a significant characteristic as a significant quality attribute of software products. So, let us begin this lecture.

(Refer Slide Time: 01:37)

Software Development- Introduction

- Software development is a process guided by systematic methods and techniques.
- 1960s saw challenges of software development. Third-generation computer hardware enabled new applications were introduced, leading to software systems of much greater scale and complexity. This resulted in cost overruns, late delivery, and ineffective and unreliable systems. This led to software crisis. This crisis led to the emergence of software engineering as a professional discipline.

 Dr. Debayan Dhar
Department of Design

Now software development is a process guided by systematic methods and techniques. So, when we talk about software essentially what we refer to as a product in itself, but then we hardly do realize the amount of work the amount of processes that are involved in the development of these softwares.

So, in 1960s saw the emergence of challenges for software development and its primarily driven because the third generation of computer hardware enabled new applications were introduced into the market, new techniques technologies were introduced and this led to software systems of much greater scale and complexity and this resulted in cost overruns,

late delivery and ineffective and unreliable systems. Now, this led to the software crisis in late in early '70s and this crisis led to the emergence of software engineering as a professional discipline.

(Refer Slide Time: 03:04)

Software Engineering

- Software engineering was founded on the ideas of structured programming (Mills, 1971).
- Programmers first define the major structures of a software system—the database, the event handler, and the network server, and then recursively decompose (break down) each structure into substructures.
- The driving vision is to gain control over design activities by making the software development process explicit and systematic.

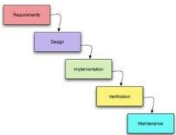
Dr. Dakshin Char
Department of Design

Now, software engineering was founded on the ideas of structured programming, programmers first define the major structures of a software system likely like the database, the event handler, the network server so on and so forth, and then recursively, they decomposed means broke down they used to break down each of these structures into substructures. So, the driving vision is to gain control over design activities by making software development process explicit and systematic.

(Refer Slide Time: 03:46)

Waterfall Model

- An early and influential model for software engineering was the waterfall: software development is organized into a series of modular phases, beginning with the analysis of functional requirements, and continuing through software design, implementation, testing, and maintenance.
- Each phase produces one or more documents that are handed off as a specification for the work of the next phase.
- The waterfall is a wonderful management tool, because projects are organized, tracked, and measured in terms of progress through the phases.



```
graph TD; A[Requirements] --> B[Design]; B --> C[Implementation]; C --> D[Testing]; D --> E[Maintenance];
```

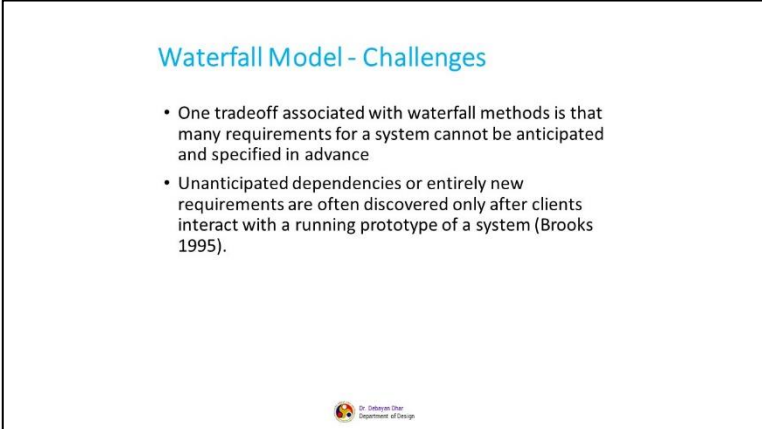
Dr. Dakshin Char
Department of Design

An early influential model for software engineering was the waterfall model. Software development is organized into a series of modular phases beginning with the analysis of

functional requirements. And continuing through software design, implementation, testing and maintenance. The figure that you can see in this slide would let you understand would provide you and in depth understanding about the process that are involved in waterfall model.

What you see here are the modular phases like requirements, design, implementation, verification. And maintenance this was the first known emergence of the modular concepts or series of modular phases that were focused on how software was developed. Each phase produces one or more documents that are handed off as a specification for the work of the next phase. The waterfall is a wonderful management tool because projects are organized, tracked and measured in terms of process through the phases.

(Refer Slide Time: 05:13)



The slide is titled "Waterfall Model - Challenges" in blue text. It contains two bullet points: "One tradeoff associated with waterfall methods is that many requirements for a system cannot be anticipated and specified in advance" and "Unanticipated dependencies or entirely new requirements are often discovered only after clients interact with a running prototype of a system (Brooks 1995)". At the bottom center, there is a small logo and the text "Dr. Siddhant Dhar, Department of Design".

Now, in spite of waterfall model being modular and providing a lot of benefits to software developers, it had its own challenges and one tradeoff associated with waterfall models is that many requirements for a system cannot be anticipated and specified in advance. So, essentially if you see the waterfall model, you would realize its a very linear process.


So, therefore, systems that are complex in nature, unanticipated dependencies or entirely new requirements are often discovered only after clients or end users interact with a running prototype of a system and therefore, this led to a lot of loss in resources in terms of time that were involved, in terms of the amount of other resources like manpower involved in development of the software, a wrong decision at an any initial stage led on to the entire process being carried on and can only be interpreted at the last phase of the development model.

What it means that if an early issue existed this would not be able, it is not possible for the design team to identify the issue unless and until they get it tested at the end user stage where testing with clients and users happen. And therefore, this became one of the major challenges of using waterfall model.

(Refer Slide Time: 06:57)

Prototyping and Iterative Development

- The waterfall model offers a framework for addressing the software crisis, but is simplistic. A strict linear flow of design specifications is unlikely to work for systems of nontrivial scale and complexity.
- A complement to structured development is prototyping, where designers develop one or more operational models to demonstrate a design idea.
- A prototype implements ideas that are abstract, making them concrete, viewable, and testable. By turning a proposal into a prototype, designers can test it for usefulness, feasibility, or other project concerns.

 Dr. Debayan Dhar
Department of Design

The waterfall model offers a framework for addressing the software crisis, but it is simplistic, it cannot handle challenges of complex systems modeling complex systems and moreover an issue arising at the initial stage could only be handled until you complete the entire process and testing results are out.


Therefore, a strict linear flow of design specifications is unlikely to work for systems of non-trivial scale and complexity and that is one of the reasons that led to the development of prototyping and iterative development. A complement to structure development is prototyping where designers develop one or more operational models to demonstrate a design idea.

A prototype implements ideas that are abstract making them concrete viewable and testable. By turning a proposal into a prototype, designers can test it for usefulness, feasibility or other project concerns. Now what we understand from this is that the primary requirement that drove the emergence of prototyping and iterative development was primarily because designers simply did not want to wait until the testing phase to identify issues with their concepts with their requirements they want to test early so, that the issues the early issues can be tackled and an entire cycle is not wasted.

(Refer Slide Time: 08:50)

Prototyping and Iterative Development

- Prototypes can be built at many phases; they can be used to evaluate requirements, high-level design, detailed software design, and so on (Boehm 1988). The feedback provided by prototyping is used to guide further development, but importantly, it may also be used to transform or reject aspects of the design.



Prototypes can be built at many phases; they can be used to evaluate requirements high level decision detailed software design and so on. The feedback provided by prototyping is used to guide further development, but importantly, it may also be used to transform or reject aspects of design. What we observe in history in software development is the early acceptance of an iterative process.


If you remember in our introduction lectures we discussed about iterative design. Design is a time bound activity and identifying requirements that are that reflect needs of actual users is important for the software getting success and therefore, quality of requirements that are extracted are of paramount importance for the design community. It is this important aspect that led to the initial rejection of waterfall model and the adoption of prototyping and iterative development models by software developers and designers.

(Refer Slide Time: 10:20)

Prototyping and Iterative Development

However, prototyping and iteration create their own problems:

- Process management can be difficult if the system evolves too rapidly to document and track the changes. Constant change can also undermine the integrity of a software design.
- One way to integrate prototyping with structured design is to view prototyping as a requirements analysis method-- prototypes are built as trial versions of a system that will be later discarded (Brooks 1995).

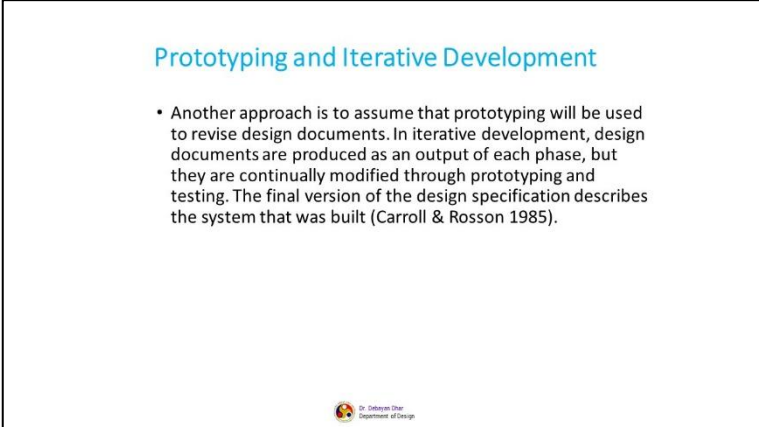


Now, prototyping and iteration has also their own problems. Process management can be difficult if the system evolves too rapidly to document and track the changes. Constant change can also undermine the integrity of a software design. One way to integrate prototyping with structure design is to view prototyping as a requirement analysis method.

Prototypes are built as trial versions of a system that will be later discarded the essence of prototyping development is test early. And it many a time it is difficult to track changes, to track the states of the systems that evolve too fast and therefore, it is highly difficult to document and track those changes.

These are some of these issues that created or that ensured that prototyping need to be visualized as a requirement analysis method. So, you test early you identify the requirement the need from the users come up with your concepts test early and that is how you do you build trial versions of a system, which based on the results of your user testing might be accepted or discarded at the later stage.

(Refer Slide Time: 12:11)



Prototyping and Iterative Development

- Another approach is to assume that prototyping will be used to revise design documents. In iterative development, design documents are produced as an output of each phase, but they are continually modified through prototyping and testing. The final version of the design specification describes the system that was built (Carroll & Rosson 1985).

Dr. Debayan Dhar
Department of Design

Another approach for prototyping is to assume that prototyping will be used to revise design documents; that means, decisions that were taken at the early phase of software development can be revised if we have prototyping and test results out in time or early in the phase. In iterative development design documents are produced as an output of each phase, but they are continually modified through prototyping and testing.


The final version of the design specification describes the system that was built right. So, in order to ensure that any wrong information does not get carried away any wrong way of defining the requirements. And a wrong feature design does not get carried away to the

final testing phase. Iterative development and prototyping can be used to ensure that quick results quick test data with users can be accessed in order to take a call on the design decisions at an early stage.

(Refer Slide Time: 13:33)

The Emergence of Usability

- Through the 1970s, it became clear that an important component of software engineering would be the user interface design. As more and more software was developed for interactive use, attention to the needs and preferences of *end users* intensified.
- In the 1960s, most software users were computing professionals who developed and maintained applications for their organizations. But as computing technology became more powerful, users became more diverse.

 Dr. Sahayam Dhar
Department of Design

This gradually led to the emergence of usability. Now in 1970s it became clear that an important component of software engineering would be the user interface design. And why so? Because developers understood that this is the medium through which our users are going to interact with the hardware and the software system.


This is the medium through which they would ensure that the tasks and goals are completed. As more and more softwares were developed for interactive views, attention to the needs and preferences of end users intensified. In 1960s, most software users were computing professionals who developed and maintained applications for their organizations. But as computing technology became more and more powerful, users became more diverse.

As end users became more diverse and less technical interactive systems came to be compared and evaluated with respect to usability. The quality of a system with respect to ease of learning, ease of use and user satisfaction. Three distinct perspectives that contributed to early views of usability and still being considered are human performance learning and cognition and collaborative activity.

(Refer Slide Time: 15:34)

The advent of Human Computer Interaction

- The increasing prominence of PCs in society made usability more visible. One important new user group consisted of cognitive scientists- psychologists, anthropologists, sociologists, and philosophers interested in how people solve problems and learn new things. Many of these scientists were starting to use computers for their own research activities. Their personal experiences often prompted research programs exploring how people learn to use and solve problems on computers.



Now, when usability entered the software development process it entered from both the ends. So, it entered from the requirements phase as well as the system testing phase. Marketing groups interviewed customers they analyzed competitive products to understand requirements. Quality assurance groups tested whether systems met design specifications regarding human performance with the system; this testing was often summarized as human factors evaluation.

(Refer Slide Time: 16:15)

Human Performance

- Unfortunately, the marketing groups rarely talked to people who would actually use the products, instead gathering requirements from management or from similar products. It was also a problem that quality assurance testing occurred at the very end of the development process. The usability evaluators might gather realistic data (performance on a fully functioning system), but the findings were too late to have impact on design.




Unfortunately, the marketing groups rarely talked to people who would actually use the products, instead gathering requirements from management or from similar products. These led to serious issues in defining actual need and requirement of end users. It was also a problem that quality assurance testing occurred at the very end of the development cycle or the process.

The usability evaluators might gather realistic data performance on a fully functioning system, but the findings were too late to have an impact on design. So, this tells that gradually there is a belief that is forming that we need usability experts or practitioners at an early phase of the software development life cycle so that we can get quality, actual requirements been defined and designed decisions being informed and directed so that the software development can fulfill the actual needs of the users.

(Refer Slide Time: 17:47)

Human Performance

- The scientific foundations for studies of human performance are psychology and industrial engineering. The emphasis is on optimal performance—simpler displays and commands, fewer keystrokes, and shorter execution times (Card, Moran, & Newell 1980).
- The specification and testing of human performance objectives demonstrated that usability could be studied empirically, and that it could play a role in software development. A first generation of experimental methods was developed, and important design variables were identified (e.g., display complexity and length of command strings).

 Dr. Dilipkumar Dhar
Department of Design


The scientific foundations for studies of human performance are psychology and industrial engineering. The emphasis for this is on optimal performance simpler displays and commands, fewer keystrokes, and shorter execution times.

The specification and testing of human performance objectives demonstrated that usability could be studied empirically, and that it could play a role in software development. A first generation of experimental methods were developed, and important design variables were identified like display complexity, length of command strings etcetera.

(Refer Slide Time: 18:47)

The advent of Human Computer Interaction

- In the early 1980s, new usability challenges emerged. Rapid learning and self-study became critical, and product development cycles were compressed. This placed a high premium on lightweight methods for improving system usability, including inspection methods based on guidelines or theory. New programming languages and tools were helping to streamline software development. Small and distributed software development organizations became common, and prototyping was used more and more to drive system development.




Now, this started the advent of what we call as the human-computer interaction, the emergence of human-computer interaction. In the early 1980s, new usability challenges emerged rapid learning and self-study became critical and product development cycles were compressed this placed a high premium on lightweight methods for improving system usability, including inspection methods based on guidelines, heuristics or theory.

New programming languages and tools were helping to streamlined software development. Small and distributed software development organizations became very very common and prototyping was used more and more to drive system development.

(Refer Slide Time: 19:57)

The advent of Human Computer Interaction

- The increasing prominence of PCs in society made usability more visible. One important new user group consisted of cognitive scientists- psychologists, anthropologists, sociologists, and philosophers interested in how people solve problems and learn new things. Many of these scientists were starting to use computers for their own research activities. Their personal experiences often prompted research programs exploring how people learn to use and solve problems on computers.




Now during the 1980s, the increasing prominence of personal computers in society made usability more visible. One important new user group consisted of cognitive scientists, psychologists, anthropologists, sociologists and philosophers became interested in how

people solve problems and learn new things. These scientists were starting to use computers for their own research activities their personal experiences often prompted research programs exploring how people learn to use and solve problems on computers.

(Refer Slide Time: 20:52)

The advent of Human Computer Interaction

- Many cognitive scientists felt that the field needed to study complex tasks. Soon spreadsheets, drawing programs, personal databases, and other applications were also in use. This new area of shared interest between computer science and cognitive science was called human-computer interaction (HCI).
- A significant early HCI project was GOMS (goals, operators, methods, and selection rules; Card, Moran, & Newell 1983). GOMS is used to analyze the goals, methods, and actions of routine human-computer interaction. This was an advance in human performance testing, because it addressed the mental activities that guide behavior.

 Dr. Sahayam Dhar
Department of Design

Now, many of these cognitive scientists felt that the field needed to study complex tasks. Soon spreadsheets, word processors, drawing programs, personal databases and other applications were also in use. This new area of shared interest between computer science and cognitive science was later called as human-computer interaction or HCI in short.

Now, a significant early HCI projects was focused on cognitive models or cognitive theory driven models like GOMS. GOMS means goals, operators, methods and selection rules proposed in 1983 for the first time by Card, Moran and Newell. We will discuss about GOMS later in the next lectures.


Now, GOMS is used to analyze the goals the methods and actions of routine human computer interaction. Routine human computer interaction means task that were routinely performed by your customers or your users. This was an advance in human performance testing; why?

Because for the first time the parameter that is mental activities guided were considered as a construct that guide human behavior that was the advent of consideration of mental operators, mental processes as parameters that influences human decision making, specifically in the context of software development.

(Refer Slide Time: 23:13)

Collaboration and Group Interaction

- In the 1990s, the scope of usability broadened to incorporate social and organizational aspects of system development and use.
- Usability came to include more emphasis on understanding the activities of users in the real world (Suchman 1987). This went beyond what is sometimes called task analysis; it involved detailed studies of work practices, roles, and concepts. Field methods were adapted from anthropology and sociology; usability engineers sometimes spent months at a work site collecting the data for requirements analysis. The descriptions such work produces are rich, but not very structured. The scenario-based framework offered a unique approach to working with work-oriented methods.




In late 1880s and in early 1990s, the scope of usability broadened and it broadened to incorporate social and organizational aspects of system development and use. Usability came to include more emphasis on understanding the activities of users in the real world this went beyond what is sometimes called as task analysis. It involved detailed studies of work practices, roles and concepts, field methods were adopted from anthropology and sociology.

Usability engineers sometimes used to spend months at works place or work site collecting data for requirements analysis for defining the users need. These descriptions such work procedures are rich but then they are not structured and therefore, lately we see the advent of scenario-based framework which offered a unique approach to working with work-oriented methods.


(Refer Slide Time: 24:56)

Collaboration and Group Interaction

- Through the 1990s, electronic mail became a pervasive communication tool, and other Internet tools such as newsgroups, multiuser domains, and real-time chat became more accessible. Communication and the coordination of work using networking software had powerful effects on organizations. The portrait of a solitary user finding and creating information on a PC became background to the portrait of groups working together in a variety of times and places.



<https://photos.state.gov/libraries/ocid/ocid/messaq/e.html>




Through the 1990s, electronic mail became a pervasive communication tool and other internet tools such as news groups, multi user domains and real time chat became more accessible. Communication and the coordination of work using networking software had powerful effects on organizations.

The portrait of a solitary user single user finding and creating information on a PC on a personal computer became background while the portrait of groups working together in a variety of times and places networking with each other in the form of completing collaboration and group interaction emerged.

(Refer Slide Time: 26:00)

Emergence of Usability Engineering

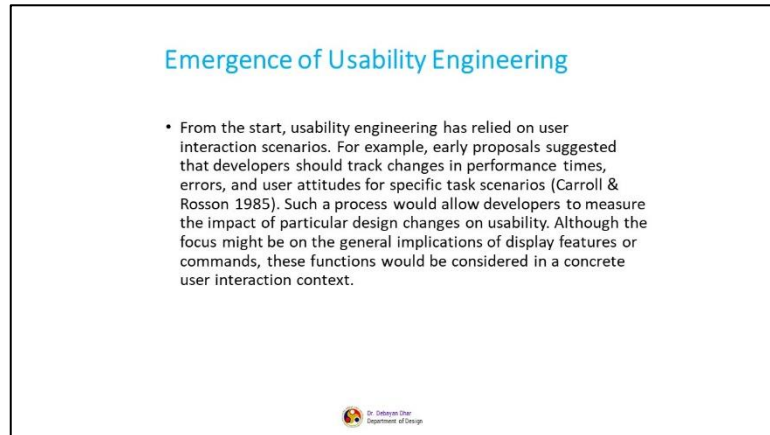
- The term usability engineering was coined by usability professionals from Digital Equipment Corporation (Good, et al. 1986). They used the term to refer to concepts and techniques for planning, achieving, and verifying objectives for system usability. The key idea is that measurable usability goals must be defined early in software development, and then assessed repeatedly during development to ensure that they are achieved (Bennett 1984; Gilb 1984).



Now, the term usability engineering is influenced by all these developments that happened through 1980s and the early 1990s. The term was first coined by usability professionals from Digital Equipment Corporation and they used the term to refer to concepts and techniques for planning, achieving and verifying objectives for system usability. Initially, the parameters were limited to planning, achieving and verifying objectives; that is what were focused essentially on planning, achieving and verifying objectives system usability.

The key idea here is that measurable usability goals must be defined early in software development because this would ensure how features or the software products are designed so, as to ensure that the requirements of your end users are met. So, the key idea here is that measurable usability goals must be defined early in software development life cycle and then assessed repeatedly during development to ensure that these objectives are met and achieved.

(Refer Slide Time: 27:37)



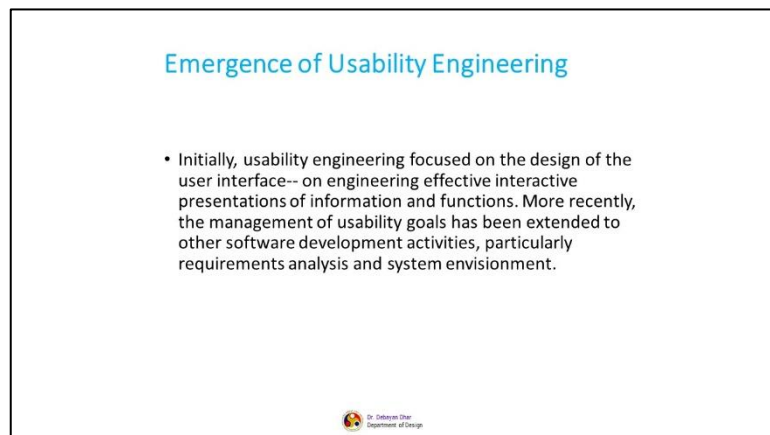
Emergence of Usability Engineering

- From the start, usability engineering has relied on user interaction scenarios. For example, early proposals suggested that developers should track changes in performance times, errors, and user attitudes for specific task scenarios (Carroll & Rosson 1985). Such a process would allow developers to measure the impact of particular design changes on usability. Although the focus might be on the general implications of display features or commands, these functions would be considered in a concrete user interaction context.

Dr. Debayan Dhar
Department of Design

From the start, usability engineering has relied extensively on user interaction scenarios. An example, early proposals suggested that developers should track changes in performance times, errors and user attitudes for specific task scenarios. Such a process would allow developers to measure the impact of particular design changes on usability. Although the focus might be on the general implications of display features or commands these functions would be considered in a concrete user interaction context.

(Refer Slide Time: 28:39)



Emergence of Usability Engineering

- Initially, usability engineering focused on the design of the user interface-- on engineering effective interactive presentations of information and functions. More recently, the management of usability goals has been extended to other software development activities, particularly requirements analysis and system envisionment.

Dr. Debayan Dhar
Department of Design

So, initially usability engineering focused on the design of the user interface, the medium that allowed the end users to communicate with the software and the hardware assembly to ensure their task or goal is reached. So, usability engineering initially focused on the design of the user interface. On engineering effective interactive presentations of information and functions and more recently the management of usability goals has been extended to other software development activities.

Particularly, requirement analysis and system envisionment. With this what we would do now is we would gradually discuss about the tools and techniques that are considered by usability practitioners human computer interaction designers specifically for defining the requirements of their end users. In the subsequent lecture we will discuss about this in detail we will explore some of the techniques and then we will also explore the detailed process of developing these software's from a human-centered perspective.