HUMAN-COMPUTER INTERACTION THIRD EDITION DIX FINLAY ABOWD BEALE

Chapter 6

HCI in the software process

1

HCI in the software process

- Software engineering and the design process for interactive systems
- Usability engineering, as a promoter of using explicit criteria to judge the success of a product in terms of its usability
- Iterative design and prototyping, as a means to incorporate crucial customer feedback early in the design process
- Design rationale, as a means to record decisions made and their context

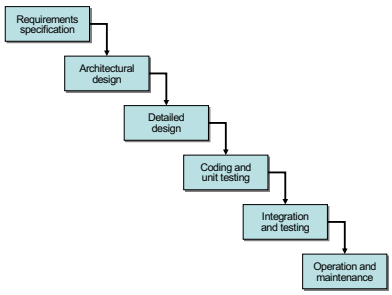
2

The software lifecycle

- Software engineering is the discipline for understanding the software design process, or life cycle
- Designing for usability occurs at all stages of the life cycle, not as a single isolated activity

3

The waterfall model



4

Activities in the life cycle

Requirements specification
: designer and customer try to capture what the system is expected to provide; can be expressed in natural language or more precise languages, such as a task analysis would provide

Architectural design
: high-level description of how the system will provide the services required; factor system into major components of the system and how they are interrelated; needs to satisfy both functional (services provided by the system) and nonfunctional (e.g. reliability, efficiency, etc.) requirements

Detailed design
: refinement of architectural components and interrelations to identify modules to be implemented separately; the refinement is governed by the nonfunctional requirements

5

Activities in the life cycle (cnd)

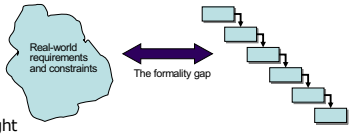
Coding and unit testing
: generating executable code from a detailed design on a per component basis; after coding, the component can be tested to verify that it performs correctly; possible to generate code automatically from sufficiently specific design

Integration and testing
: the individually implemented and tested components are integrated to form the complete application, as described in the architectural design; further testing is done at global level; certification may also be needed

Maintenance
: after product release, any further work on the system (e.g. correction of errors, revisions to satisfy additional user requirements) is considered maintenance and provides feedback to all of the other activities in the life cycle

6

Verification and validation



Verification
designing the product right
Validation
designing the right product

The formality gap
validation will always rely to some extent on subjective means of proof, as the real world is inherently ambiguous
Management and contractual issues
design in commercial and legal contexts, where factors such as time constraints and economic forces are important

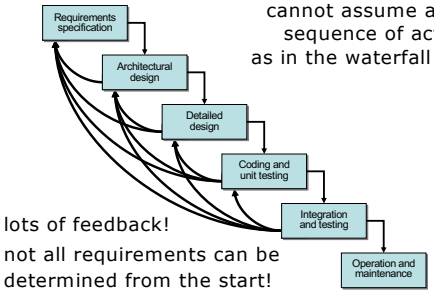
7

Interactive systems and the software life cycle

- The first software systems were mostly data-processing ones with low user interactivity; usability issues were not important
- Modern software applications have high level of interactivity; about 50% of the designer's time is spent on designing code for user interfaces
- Very often, users don't know all the tasks that they will perform before they have started using the system and becoming familiar with it

8

The life cycle for interactive systems



cannot assume a linear sequence of activities as in the waterfall model

lots of feedback!

not all requirements can be determined from the start!

9

Usability engineering

The ultimate test of usability based on measurement of user experience

Usability engineering demands that specific usability measures be made explicit as requirements

Usability specification
- usability attribute/principle
- measuring concept
- measuring method
- now level/ worst case/ planned level/ best case

Problems
- usability specification requires a level of detail that may not be possible early in design
- satisfying a usability specification does not necessarily satisfy usability itself (e.g. do fewer explicit actions make an undo operation easier?)

10

Part of a usability specification for a VCR

Attribute: Backward recoverability	
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program
Now level:	No current product allows such an undo
Worst case:	As many actions as it takes to program-in mistake
Planned level:	A maximum of two explicit user actions
Best case:	One explicit cancel action

11

Criteria to determine the measuring method for a usability attribute

1. Time to complete a task
2. Per cent of task completed
3. Per cent of task completed per unit time
4. Ratio of successes to failures
5. Time spent in errors
6. Per cent or number of errors
7. Per cent or number of competitors better than it
8. Number of commands used
9. Frequency of help and documentation use
10. Per cent of favorable/unfavorable user comments
11. Number of repetitions of failed commands
12. Number of runs of successes and of failures
13. Number of times interface misleads the user
14. Number of good and bad features recalled by users
15. Number of available commands not invoked
16. Number of regressive behaviors
17. Number of users preferring your system
18. Number of times users need to work around a problem
19. Number of times the user is disrupted from a work task
20. Number of times user loses control of the system
21. Number of times user expresses frustration or satisfaction

12

Possible ways to set measurable levels in a usability specification

Set levels with respect to information on:

1. an existing system or previous version
2. competitive systems
3. carrying out the task without use of a computer system
4. an absolute scale
5. your own prototype
6. user's own earlier performance
7. each component of a system separately
8. a successive split of the difference between best and worst values observed in user tests

13

ISO usability standard 9241

Adopts traditional usability categories:

- Effectiveness
  - can you achieve what you want to?
- Efficiency
  - can you do it without wasting effort?
- Satisfaction
  - do you enjoy the process?

14

Some metrics from ISO 9241

Usability objective	Effectiveness measures	Efficiency measures	Satisfaction measures
Suitability for the task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of power features used	Relative efficiency compared with an expert user	Rating scale for satisfaction with power features
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for ease of learning
Error tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

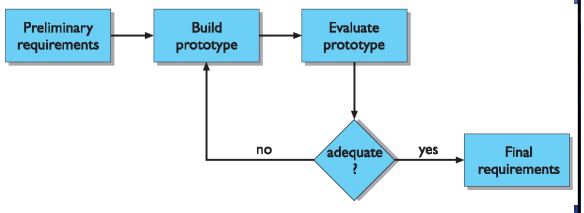
15

Iterative design and prototyping

- Iterative design overcomes inherent problems of incomplete requirements
- Prototypes
  - simulate or animate some features of intended system
  - different types of prototypes
    - throw-away: prototype built and tested, design knowledge used to build the final product, but the actual prototype is discarded
    - incremental: final product is built as separate components, one at a time; each new release includes one more component
    - evolutionary: prototype serves as the basis for the next iteration of design; the actual system evolves from a limited initial version to a final release; operation and maintenance activities are included
- Management issues
  - time
  - planning
  - non-functional features
  - contracts

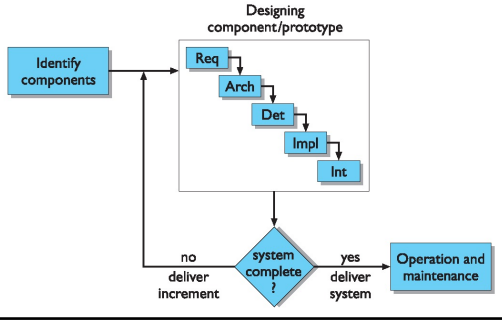
16

Throw-away prototyping within requirements specification



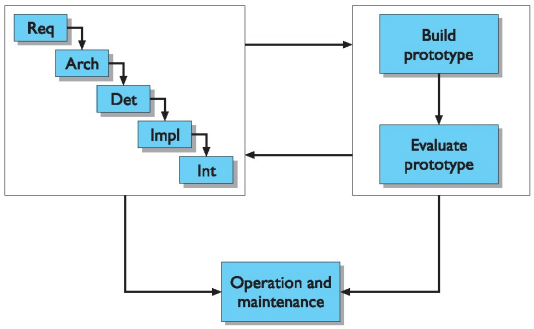
17

Incremental prototyping within the life cycle



18

Evolutionary prototyping throughout the life cycle



19

Management issues

Time
- building prototypes takes time and especially a throw-away prototype may be seen as wasting precious time; hence do *rapid prototyping* but no rushed evaluation

Planning
- managers may not have the experience to assess time/cost overheads in building prototypes

Non-functional issues
- such as reliability, safety, response time, are precisely the ones that are sacrificed in prototyping; so how real is eventually the prototype we have built?

Contracts
- prototypes cannot form the basis for legal contracts, as the latter are affected by many managerial and technical issues

20

Techniques for prototyping

Storyboards
- need not be computer-based, just snapshots of intended interfaces (as in a series of panels in the film industry)
- can be animated using graphical drawing packages or be enhanced with annotations and scripts describing interactions

Limited functionality simulations
- some part of system functionality provided by designers
- tools like HyperCard, Macromedia, etc. are common for this
- *Wizard of Oz* technique, where the prototype provides limited functionality enhanced by human intervention (e.g. an accountant uses a simple prototype and an observer (wizard) translates the accountant's commands to more complex functionality)

High-level programming support
- languages such as HyperTalk and environments such as UIMS (user interface management system) allow the independent development of the interface prototype from the underlying functionality

21

Warning about iterative design

The ideal model of iterative design, in which a prototype is designed, evaluated and modified until the best possible design is achieved in a given project time, is appealing, but…

Design inertia
- early bad decisions at the very beginning may remain bad, despite a number of iterations

Make sure you understand the reason for a problem and not just detect a symptom
- if a user has difficulty setting correctly the time in an appliance's control panel, is it because the buttons need a better design or because he things the time as being a 12-hour model whereas the appliance uses a 24-hour one?

22

Design rationale

Design rationale is information that explains why a computer system is the way it is

Benefits of design rationale
- communication throughout life cycle, providing in the future understanding of the critical decisions that were made
- reuse of design knowledge across products in similar situations by possibly different design teams
- enforces design discipline, by forcing the designer to deliberate for the decisions taken
- presents arguments for design trade-offs (e.g. use visible buttons or hidden sub-menu items)
- organizes potentially large design space, by indicating all alternatives that have been investigated
- capturing contextual information, that justifies the particular design decisions

23

Design rationale (ctd)

Types:
- Process-oriented
  - preserves order of deliberation and decision-making
  - intended to reduce stress on the generalization of design knowledge for use between different products
- Structure-oriented
  - emphasizes post hoc structuring of considered design alternatives
- Two examples:
  - Issue-based information system (IBIS)
  - Design space analysis

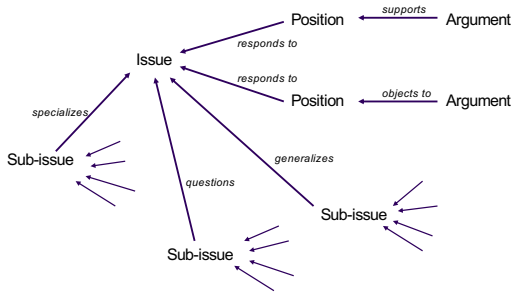
24

Issue-based information system (IBIS)

- Basis for much of design rationale research
- Process-oriented
- Main elements:
  - issues
    - hierarchical structure with one 'root' issue
  - positions
    - potential resolutions of an issue
  - arguments
    - modify the relationship between positions and issues
- gIBIS is a graphical version

25

Structure of gIBIS



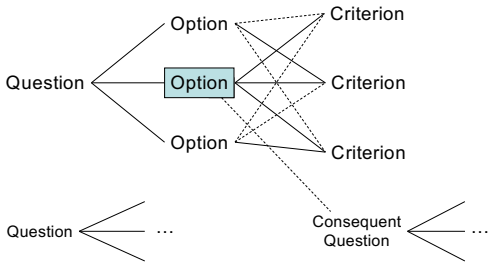
26

Design space analysis

- Structure-oriented
- QOC – hierarchical structure:
  - questions (and sub-questions)
    - represent major issues of a design
  - options
    - provide alternative solutions to the question
  - criteria
    - the means to assess the options in order to make a choice
  - the key here is deciding the right questions and the correct criteria to judge the options
- DRL – Decision Representation Language, similar to QOC with a larger language and more formal semantics
  - decision problems instead of questions
  - alternatives instead of options
  - goals instead of criteria

27

The QOC notation



28

Psychological design rationale

- To support task-artefact cycle in which user tasks are affected by the systems they use
- Aims to make explicit consequences of design for users (rather than capturing the designer's intentions)
- Designers identify tasks system will support
- Scenarios are suggested to test task
- Users are observed using the first version of the system
- Psychological claims of system made explicit
- Negative aspects of design can be used to improve next iteration of design
- Such a documentation of the psychological design rationale makes the designer more aware of the natural evolution of user tasks, in improving later designs

29

Summary

The software engineering life cycle
- distinct activities and the consequences for interactive system design

Usability engineering
- making usability measurements explicit as requirements

Iterative design and prototyping
- limited functionality simulations and animations

Design rationale
- recording design knowledge
- process vs. structure

30