Lecture 4

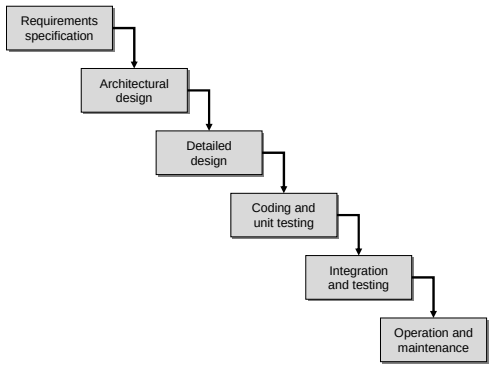
HCI in the software process

Chapter 6

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
- There are many models of the software life cycle we will look at the 2 main ones
  - Waterfall
  - Prototyping

The waterfall model



Activities in the life cycle

•Requirements specification

–designer and customer try 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

–Informal design and scenario based design will result in better requirements analysis

Detailed design

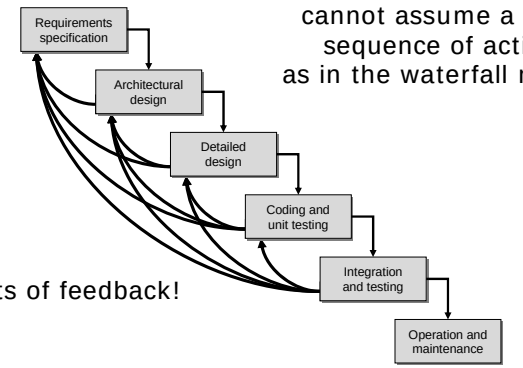
- Detailed design of the interface
- Move from informal to formal specification

- Separation of layers
  - A layered approach to software development will provide for more flexibility
    - Data
    - Logic
    - Interface

Testing

- Testing is not only about functionality of code
- Usability testing
  - There are some basics that are nearly always important
    - Layout
    - Language
    - Number of click/steps to perform task
    - Choose rather than remember
  - More detail in weeks 7 & 8

The life cycle for interactive systems

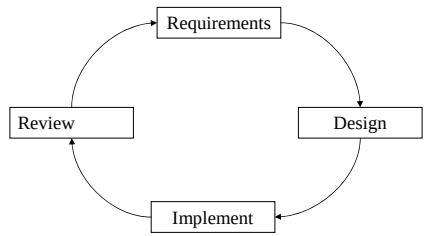


cannot assume a linear sequence of activities as in the waterfall model

lots of feedback!

Iterative design and prototyping

- Iterative design overcomes inherent problems of incomplete requirements



Prototyping, Management

- Prototypes
  - simulate or animate some features of intended system
  - different types of prototypes
    - Throw-away
      - probably best from quality perspective – prototype software is discarded
    - Incremental
      - Series of component products; this can be good
    - Or Evolutionary – the system evolves
- Management issues
  - Contract oriented – formal agreements of what will be delivered
  - Related, the activities tend to be temporally-bound *phases*
  - Can detract from the natural possibilities for iteration and feedback toward a highly usable design
    - Prototyping takes time; requires planning, budgeting

Techniques for prototyping

Storyboards

need not be computer-based

can be animated

Limited functionality simulations

some part of system functionality provided by designers

tools like HyperCard are common for these

Wizard of Oz technique – 'pay no attention to the man behind the curtain' Before programming a technology, have a person perform the function in a usability situation. 'Listening typewriter' was first prototyped this way

Warning about iterative design

design inertia – early bad decisions stay bad

diagnosing real usability problems in prototypes….

…. and not just the symptoms

Usability Engineering

- Introduce explicit usability engineering goals into the design process
  - *Usability specification* consists of measuring concept, measuring method, now (current) level, worst case (lowest acceptable performance), planned level and best case
  - Various measuring methods (see table 6.2); e.g., time to complete task
    - Note that *satisfaction* (e.g., via a rating scale) is an important type of usability measure

Problem with Usability Engineering

- Very definite and measurable which is good, but…
  - At early stage of the design it is often hard to tell what specific user actions and situations will be most important to overall success of the system
  - Might end up satisfying the usability specification but not actually getting usability (the assumption is that satisfying the specific measures is good, but it might not be sufficient)

Design Rationale

- The information that explains why a computer system is the way it is
  - Support communication among team members
  - Accumulates knowledge transferable across a set of products
  - Forces designers to think carefully about decisions
- In HCI, the design rationale is particularly important because there's seldom one 'best' design alternative
  - It's usually some sort of trade-off

IBIS

- 'Issue Based Information System' (IBIS)
  - A process-oriented design rationale
  - Document design decisions as a graph
- Root node is 'issue'
  - Various 'positions' are put forth as solutions to the issue
  - These have supporting and refuting 'arguments'
  - Issues can also have 'sub-issues' (see p.251)

Design space analysis

- QOC – question, option, criterion
  - Question - similar to Issue in IBIS
  - Option – similar to IBIS Position
  - *Criterion* addressed favourably by an Option are joined by a solid line; those that fail get dashed line
  - Best Option gets a box around it
  - Options can have Consequent Questions (and hence subgraphs)

Waterfall or Prototype

- Waterfall
  - Interaction paradigm 'standard' and well understood?
  - The problem is well understood?
  - Data centric systems
    - Information systems
    - Data warehouse
- Prototype
  - The interaction paradigm new or poorly understood?
  - The problem definition is incomplete or poorly defined?
  - Interface centric systems
    - Games
    - Modelling
    - Design tools

Waterfall or prototype

- It doesn't have to be a one or the other decision
- Many systems are a blend
  - With some parts are prototyped to elicit requirements
  - There isn't one 'best way'
  - Nor is there a 'silver bullet'
- But the less well understood the solution is (the higher the *risk*), the more iteration and prototyping is warranted