chapter 6

HCI in the software process
Software Engineering

• Software design process or life cycle called Software Engineering
  – that addresses the management and technical issues of the development of software systems.
  – The software life cycle is an attempt to identify every activity that occur in software development from the initial concept formation for a software system up until its eventual phasing out and replacement.
  – These activities must then be ordered in time in any development Project
The waterfall model

- Requirements specification
- Architectural design
- Detailed design
- Coding and unit testing
- Integration and testing
- Operation and maintenance
Activities in the life cycle

- **Requirements specification**
  - Designers try to capture what functionality is expected from the system
  - “HOW” is not concern of this activity.
  - Can be expressed in natural language or more precise languages like Unified Modeling Language.
  - begins at the start of product development and the requirements are from the customer’s perspective
  - It involves eliciting information from the customer about the work environment, or domain, in which the final product will function.
Activities in the life cycle

- **Architectural Design:**
  - A decomposition of the system that allows independent development of separate components
  - These components will later be integrated
  - Satisfying both functional and non-functional requirements after integration

- **Detailed Design:**
  - Designer is responsible to ensure complete functionality of the component(s) provided by the architectural design phase.
Activities in the life cycle

• **Coding & Testing:**
  - Transform detailed design in a programming language.
  - Component testing

• **Integration & Testing:**
  - Completed components should be tested individually
  - they must be integrated as described in the architectural design.
  - Integration testing is performed to ensure correct behavior and acceptable use of any shared resources e.g. memory
Activities in the life cycle

- **Maintenance:**
  - involves the correction of errors in the system which are discovered after release
  - System can be revised to satisfy requirements that were not realized during previous development.
  - maintenance provides feedback to all of the other activities in the life cycle
Verification and validation

• Verification
  – Are you building the product right?
  – Software must conform to its specification

• Validation
  – Are you building the right product?
  – Software should do what the user really requires

• Validation & Verification relies on some proof
  – Subjective means of proof i.e. User signed Requirements/system design
  – Management contracts
    • Time
    • Economical issues
    • Recourses e.g. Manpower, hardware etc
The life cycle of interactive systems cannot assume a linear sequence of activities as in the waterfall model

- lots of feedback!
- The actual design process is **iterative**
ISO usability standard 9241

Usability has following categories:

- **effectiveness**
  - can you achieve what you want to?

- **efficiency**
  - can you do it without wasting effort?

- **satisfaction**
  - do you enjoy the process?
## ISO usability standard 9241

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

- Iterative design overcomes inherent problems of incomplete requirements

- Prototypes
  - Simulate or animate some but not all features of intended system

- Types of prototypes:
  - throw-away/Rapid Prototyping
  - Incremental Prototyping
  - Evolutionary Prototyping
Throw Away/Rapid Prototyping

- The prototype is built and tested.
- It is likely to *inform* the final solution, but the prototype itself will not become part of the final solution.
Techniques of Prototyping

• **Storyboards:**
  – is a graphical depiction of the interface without accompanying system functionality.
  – can be animated, If not animated, storyboards usually include annotations and scripts indicating how the interaction will occur.

• **Limited functionality simulation:**
  – some part of system functionality provided by designers
  – tools like HyperCard & Wizard of Oz technique
Case Study

- A software solution is required to display information about incoming and outgoing flights to the public at Palm Island Airport. The airport terminal will have a number of large display screens (approximately 42" in size, measured horizontally) for viewing by the public. The display screens will provide information about flight arrivals and departures at Palm Island Airport.

- It is envisaged that this information will require more than one page. Therefore, each page will be required to display for not less than 10 seconds, and not more than 15 seconds, before moving onto the next page in the cycle. Each time a page loads, it must show the latest information available at that time. Airport branding and the current time and date must also be shown on all pages that are visible to the public.

- The layout should be designed in such a way that members of the public can quickly find the appropriate flight and check its status. Any flight that has been cancelled must stand out. The information should be readable from a distance of not less than four metres. At peak times, passengers are likely to seek information about one of five flights arriving and/or departing (potentially ten flights in total: up to five arrivals; up to five departures).

- The solution also requires that administration staff be able to: insert, edit and delete flight information. The administration user-interface must be intuitive and efficient. The system must also allow more than one administrator to update flight information simultaneously (not necessarily details of the same flight). When an administrator completes an update, that information must be available to the public at the next screen (or page) refresh.
Case Study (contd)

- The public information necessary about flights arriving is:
  - Flight Number
  - Where the flight is coming from (eg airport or city)
  - Expected time of arrival
  - Current status of flight (eg en-route, landed, delayed, cancelled, diverted, contact operator)
  - Gate of arrival
- The public information necessary about flights departing is:
  - Flight Number
  - Destination of flight (eg airport or city)
  - Expected time of departure
  - Current status of flight (eg As scheduled, gate open, boarding, last call, gate closed, airborne, delayed, cancelled, contact operator)
  - Departure gate
- The client would also like to make this information accessible to members of the public when using one of the internet connected information kiosks situated around the airport, and for users visiting the Palm Island Airport website. Legislation must be strictly adhered to as applicable (eg as relevant from copyright, data protections, accessibility and disability discrimination, etc)