Course goals

For students
- Programming experience on Tablet PC
- UI and Design experience
- Work in team
- Develop an application for an external customer

Course goals

For Richard Anderson
- Build undergraduate expertise in Tablet PC development
- Prototype of TPC capstone
- Ugrad curriculum shift

Course goals

For Chris Mason [I’m making these up]
- Explore approaches to diagnostic tools
- Build ties with UW CSE
- Get to know UW Programs and what students can accomplish
- Artifacts to show to Schindler
  - Justify time spent with UW
  - Suggest R&D Directions for Schindler

Team organization

- Classic software teams
  - Program manager
  - Developers (Dev lead + devs)
  - Test
  - Documentation/UI
- Other models
  - Fad of the day

Waterfall model (McConnell)

System specification
- Requirements Analysis
- Architectural Design
- Detailed Design
- Coding and Debugging
- Unit testing
- System testing
- Maintenance
<table>
<thead>
<tr>
<th>Requirements</th>
<th>Challenges of requirements gathering (Kulak, Guiney)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Gather and document the functions that the application should perform for the users in the users' language and from the users' perspective”</td>
<td>Finding out what users need</td>
</tr>
<tr>
<td>Requirements should neither constrain nor define methods of implementation</td>
<td>Documenting users' needs</td>
</tr>
<tr>
<td></td>
<td>Avoiding premature design assumptions</td>
</tr>
<tr>
<td></td>
<td>Resolving conflicting requirements</td>
</tr>
<tr>
<td></td>
<td>Eliminating redundant requirements</td>
</tr>
<tr>
<td></td>
<td>Reducing overwhelming volume</td>
</tr>
<tr>
<td></td>
<td>Traceability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>User requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of interactions</td>
<td>Requirements from the user's point of view</td>
</tr>
<tr>
<td>Text details</td>
<td>Expressed in the user's language</td>
</tr>
<tr>
<td>Example</td>
<td>Based on understanding of user's application</td>
</tr>
<tr>
<td>Authenticate User</td>
<td>Does not define implementation</td>
</tr>
<tr>
<td>○ Actors: User, Unauthorized user</td>
<td>How do we get them???</td>
</tr>
<tr>
<td>○ Summary: Users request entry to the system, valid credentials allow access</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements gathering</th>
<th>Understanding use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand application from users perspective</td>
<td>Not asking users to define the application</td>
</tr>
<tr>
<td>○ An application which doesn't match needs won't be purchased, or won't be used</td>
<td>Observations, Interviews, Examination of artifacts, Focus Groups</td>
</tr>
<tr>
<td>Building for a specific customer</td>
<td>Ethnography</td>
</tr>
<tr>
<td>Building a widely used application, getting requirements from representative users</td>
<td>○ Branch of anthropology dealing with the scientific description of individual cultures</td>
</tr>
</tbody>
</table>
Field observations

- Protocols developed in many academic fields
- Event based
- Narrative

What do you do with the data?

- Define user experience of application
- Application must support the process
- Efficient handling of common cases
- Ability to handle exceptional cases (which aren’t all that exceptional!)
- Develop feature lists

User requirements

- Business Requirements
- Use Cases
- User Requirements
- Functional Requirements

Software project failures

- Software projects have a reputation for failure
  - Probably well deserved
  - Many examples of massive cost over runs, release delays and cancellations

Project failure

- Not delivering working program on targeted date
  - Overrun on time/budget
  - Under delivery of functionality or quality

All to common case

- Project starts out fine, with a few minor changes in requirements, delays of supporting activities and changes in personnel
- Coding proceeds at a good rate with most modules almost working at the point when the system is to integrated
Then everything goes wrong

- Integration reveals incompatibility between components
- Integration reveals severe bugs in components
- Unexpected hardware or software change
- And a few random disasters
  - Source code lost, key people directed to other tasks, sudden changes in requirements or schedule

What happens next

- Devs code like hell
  - Fixing and patching bugs
  - Significant changes in architecture or functionality on-the-fly
- Test and documentation held up
  - “The build is broken – I can’t do anything”
- Long hours
  - Negative team dynamics
  - Damage control activities

Day of reckoning

- Substandard product shipped
  - “It’s just version 1.0 – we can issue an upgrade”
- Schedule shifts
- Project cancelled or downgraded

Classic Mistakes

- McConnell, *Rapid Development*
  - People related mistakes
  - Process related mistakes
  - Product related mistakes
  - Technology related mistakes

People issues (high level)

- Personnel management
  - Functioning team
- Relationship with customer
- Management issues
  - Management support and competence

People related mistakes

- Motivation
- Weak personnel
- Problem employees
- Heroics
- Adding people to a late project
- Crowded offices
- Friction between dev and customers
- Unrealistic expectations
- Lack of sponsorship
- Lack of stakeholder buy-in
- Lack of user input
- Politics over substance
- Wishful thinking
### Process issues (high level)
- Accurate planning
  - Realistic scheduling
  - Contingency planning
- Paying attention to all stages of product development

### Process related mistakes
- Optimistic schedules
- Insufficient risk management
- Contractor failure
- Insufficient planning
- Abandonment of planning under pressure
- Wasted time in "fuzzy front end"
- Shortchanged upstream activities
- Inadequate design
- Shortchanged QA
- Insufficient management controls
- Premature convergence
- Omitting necessary tasks from estimates
- Planning to catch up later
- Code-like-hell programming

### Product related mistakes
- Requirements gold-plating
- Feature creep
- Developer gold-plating
- Push-me, pull-me negotiation
  - Adding new tasks when schedule slips
- Research-oriented development

### Technology related mistakes
- Silver-bullet syndrome
- Overestimating savings from new tools or methods
- Switching tools in the middle of a project
- Lack of automated source-code control