Syllabus

Text book

Syllabus
Chapters 14-23 of Pressman: Software Quality Management

Lab
- Design patterns
- Spring application framework
- Scrum (agile software development)
- Unit testing
- Mercurial (software revision control)
Grading

- Final: 50%
- Term project: 40%
- Lab work and quizzes: 10%
Project

- Project: Team-based, 5-6 students per team
- Each team should deliver a production-quality software implementation of their chosen project by the end of the semester
- March 18: Find your team
- March 25: Find your project idea (TA must approve it)
  - You can suggest your own ideas
Project (2)

Deliverables (tentative)

- Two weeks after the project idea has been approved:
  - Specifications documents (Scrum’s *user stories*)
- By the end of the semester:
  - An integrated and working project
    - NOTE: if the project was not integrated nor working, no other deliverables will be accepted!
  - Unit tests (all of them should pass)
  - Design documents: UML class diagram and sequence diagrams, and other types of diagrams if needed
  - List of known bugs (if any)
- Periodic:
  - Every two weeks: Sprint backlogs (*one page* of *paper* per team, *i.e.* not electronic, not on a CD, not by e-mail)
    - NOTE: sprint backlogs must include which team member was assigned which task!
Ethics (very important)

Any act of dishonesty, cheating, code stealing, “copy and paste”, will not be tolerated:

▶ 0 grades out of 50 immediately!
▶ Put on a black list and will be watched closely in any other subjects
▶ Will be denied of any collaboration opportunity (with me at least), whether in graduation project, post graduate studies, or extra-curricular research activities
▶ This is for dishonesty in assignments, or the term project
▶ Punishment applies equally to the person/team who cheated, and the person/team who allowed others to cheat from him/them (if any)

Students who did not contribute to the team work at all will receive 0 out of 40!
Chapter 14: Quality Concepts
Key Concepts

- Importance of quality management
- Defining quality
- Quality dimensions and factors
- Software quality dilemma
- Cost of quality
Why is Quality Important
Downtime

- Defective code responsible for 45% of computer system downtime in 2001
- US companies in 2001 lost 100 billion dollars in lost productivity and repairs...
- ... not including the cost of losing angry customers
Latency

- Amazon found every 100ms of latency cost them 1% in sales
- Google found an extra 500ms in search page generation time dropped traffic by 20%
- A broker could lose $4 million per millisecond if their electronic trading platform is 5 milliseconds behind the competition
Inaccuracy

A bug in hospital software killed 5 people and developed serious complications in 15 others by a radiation overdose (year 2000 in Panama)

- They were accused of second degree murder! (Life imprisonment)
Inaccuracy

A bug in hospital software killed 5 people and developed serious complications in 15 others by a radiation overdose (year 2000 in Panama)

- They were accused of second degree murder! (Life imprisonment)
► 3-4 defects per 1000 lines of code is enough to make the software perform badly
► Most programmers inject 1 error per 10 lines of code
Meskimen’s Law

“Theres never time to do it right, but always time to do it over again”
Some Benefits

- Amount of rework is reduced
- Lower costs
- Reduced time-to-market
What is Quality
Relative vs. Absolute

- Relative: It is easy to tell whether product A > product B with respect to some feature/measure X
- Absolute: Not so easy...
Pragmatically

Transcendental view: recognizable
User view: meets end user's specific goals
Manufacturer's view: conforms to specifications
Product view: functions and features of the product
Value-based view: how much the customer is willing to pay
Quality of Design
The degree to which the design meets the functions and features specified in the requirements model

Quality of Conformance
The degree to which the implementation follows the design and the resulting system meets its requirements and performance goals

BUT...

User Satisfaction
Quality is important, but if the user isn't satisfied, nothing else really matters.
If a software product provides substantial benefit to its end users, they may be willing to tolerate occasional reliability or performance problems.
Quality of Design
The degree to which the design meets the functions and features specified in the requirements model

Quality of Conformance
The degree to which the implementation follows the design and the resulting system meets its requirements and performance goals

BUT...

User Satisfaction
Quality is important, but if the user isn't satisfied, nothing else really matters.
If a software product provides substantial benefit to its end users, they may be willing to tolerate occasional reliability or performance problems
Software Quality

“An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.”

- Ineffective software process leads to project chaos which leads to poor quality
- Useful product: meets requirements, reliable and error-free, easy to use
- Added value for producers: less maintenance, fewer bug fixes, reduced customer support
# Quality Dimensions and Factors

<table>
<thead>
<tr>
<th>Garvin’s Quality Dimensions</th>
<th>Not specific to software</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>McCall’s Quality Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categorizes software quality factors into</td>
</tr>
<tr>
<td>- Product revision, Product transition, Product operation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISO 9126 Quality Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six key quality attributes</td>
</tr>
<tr>
<td>- Functionality, Reliability, Usability, Efficiency, Maintainability, Portability</td>
</tr>
</tbody>
</table>

Targeted Quality Factors

<table>
<thead>
<tr>
<th>Question-based factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Intuitiveness, Efficiency, Robustness, Richness</td>
</tr>
</tbody>
</table>
## Quality Dimensions and Factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>Garvin’s</th>
<th>McCall’s</th>
<th>ISO 9126</th>
<th>Targeted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Usability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Efficiency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Maintainability</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Portability</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

### Quantitative Measurable for Software Quality

**Later in Ch. 23**
The Software Quality Dilemma
- Neglect quality at all → you lose
- Spend too much resources on quality → you lose too
Good Enough Software

“Good Enough” Release

- Most features will have high quality
- Other **uncommon** features may have known bugs
- → Most users will be happy
- Fix the bugs in the next version

Disadvantages

- May damage small companies’ reputation
- May not work for all application domains (like embedded vehicle software or avionics)
The Cost of Quality

### Costs to assure quality
- **Prevention**
  - quality planning
  - complete requirements and design models
  - test planning
  - training
- **Appraisal**
  - technical reviews
  - data collection and metrics evaluation
  - testing and debugging

### Costs for lack of quality
- **Internal failure (prior to shipment)**
  - repair
  - repair side effects
  - failure mode analysis
- **External failure (after shipment)**
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work
  - reputation damage
Cost

Figure: The relative costs to finding and repairing one error or defect: early vs. late (industry average cost)
Thank you!