CMPT 473
Software Testing, Reliability and Security

Intro to Software Quality

Nick Sumner
Today

- Quiz
- Administrivia
- Overview
- Intro to Software Quality
Course Website

  - Policies
  - Schedule
  - Reading
  - Assignments
  - Course Project
  - ...

People

- Instructor: Nick Sumner
  - Office Hour: Thursday 5:30-6:30?
  - Office: ?
People

- Instructor: Nick Sumner
  - Office Hour: Thursday 5:30-6:30?
  - Office: ?

- TAs: Golnaz Gharachorlu
  Shreya Sadashiva Shetty
Grading

- 40% Assignments
- 30% Exam
- 30% Quizzes (Dropping the worst 2) & Homework
Assignments

- 4+ Assignments
Assignments

- 4+ Assignments

- Practical experience learning how to use tools & techniques
Assignments

• 4+ Assignments

• Practical experience learning how to use tools & techniques

• Assignments may be turned in late with a 20% penalty for each day.
Exams

- Two options for a final exam:
Exams

• Two options for a final exam:
  1) Written final exam
Exams

• Two options for a final exam:
  1) Written final exam
  2) Practical research project based final
    • Requires meeting with me by the middle of the semester to get a project approved (and to commit to a project)
Exams

• Two options for a final exam:
  1) Written final exam
  2) Practical research project based final
     • Requires meeting with me by the middle of the semester to get a project approved (and to commit to a project)

• Why two options?
Reading

- Not *required* to purchase any books for the course
Reading

- Not *required* to purchase any books for the course

- Books available online or through library for reading assignments
Reading

- Not *required* to purchase any books for the course
- Books available online or through library for reading assignments
- Still need to complete reading assignments before class for the day they are assigned
What is Software Quality?
What is Software Quality?

- Bad software costs developers and users *billions* of dollars a year, but why?
  - What is `good' software?
  - What is `bad' software?
What is Software Quality?

- Bad software costs developers and users *billions* of dollars a year, but why?
  - What is `good' software?
  - What is `bad' software?

- Try to identify 5-6 most important characteristics of `quality' software. (and examples)
It is a Matter of Perspective

Your role relative to the software changes what is important to you.

- End Users
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- End Users
  - Does the software have all desired features?
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- **End Users**
  - Does the software have all desired features?
  - Can it reliably produce correct results for good input?
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- **End Users**
  - Does the software have all desired features?
  - Can it reliably produce correct results for good input?
  - Does it securely & gracefully handle bad input?
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- **End Users**
  - Does the software have all desired features?
  - Can it reliably produce correct results for good input?
  - Does it securely & gracefully handle bad input?
  - Is it easy to use?
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- End Users
  - Does the software have all desired features?
  - Can it reliably produce correct results for good input?
  - Does it securely & gracefully handle bad input?
  - Is it easy to use?
  - Is it responsive?
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- **End Users**
  - Does the software have all desired features?
  - Can it reliably produce correct results for good input?
  - Does it securely & gracefully handle bad input?
  - Is it easy to use?
  - Is it responsive?
  - Does it integrate well with other software?
It is a Matter of Perspective

Your *role* relative to the software changes what is important to you.

- **End Users**
  - Does the software have all desired features?
  - Can it reliably produce correct results for good input?
  - Does it securely & gracefully handle bad input?
  - Is it easy to use?
  - Is it responsive?
  - Does it integrate well with other software?

What are the consequences of missing these goals?
It is a Matter of Perspective

- Operational / Deployment
It is a Matter of Perspective

- Operational / Deployment
  - Is the software secure from attacks that may compromise the IT infrastructure?
It is a Matter of Perspective

• Operational / Deployment
  – Is the software secure from attacks that may compromise the IT infrastructure?
  – Does the software appropriately use resources? (CPU, Memory, Disk Space, Bandwidth, ...)
It is a Matter of Perspective

- Operational / Deployment
  - Is the software secure from attacks that may compromise the IT infrastructure?
  - Does the software appropriately use resources? (CPU, Memory, Disk Space, Bandwidth, ...)

What are the consequences of missing these goals?
It is a Matter of Perspective

- Developers
It is a Matter of Perspective

- Developers
  - How easy is the software to adapt to changes in requirements?
It is a Matter of Perspective

• Developers
  – How easy is the software to adapt to changes in requirements?
  – Can the software be easily adapted to other systems?
It is a Matter of Perspective

- Developers
  - How easy is the software to adapt to changes in requirements?
  - Can the software be easily adapted to other systems?
  - How easy is the software to inspect and understand?
It is a Matter of Perspective

- Developers
  - How easy is the software to adapt to changes in requirements?
  - Can the software be easily adapted to other systems?
  - How easy is the software to inspect and understand?
  - Can components be easily examined gauged for quality?
It is a Matter of Perspective

- Developers
  - How easy is the software to adapt to changes in requirements?
  - Can the software be easily adapted to other systems?
  - How easy is the software to inspect and understand?
  - Can components be easily examined gauged for quality?

What are the consequences of missing these goals?
ISO/IEC 9126 groups them as:

1) Functionality
2) Reliability
3) Usability
4) Efficiency
5) Portability
6) Maintainability
ISO/IEC 9126 groups them as:

1) Functionality
2) Reliability
3) Usability
4) Efficiency
5) Portability
6) Maintainability

Almost:

End User
Operational
Developer
Going Deeper

- Reliability – Doesn't just mean not having faults!
Going Deeper

• Reliability – Doesn't just mean not having faults!
  – Avoid failure resulting from software faults
  – Maintain performance in the face of faults or attacks
  – Reestablish performance and data after a failure
Going Deeper

- Usability- Beyond adequate to polished
Going Deeper

- Usability- Beyond adequate to polished
  - Helps user understand whether the program meets their needs
  - Features & uses are easy to learn
  - Easy to operate & control
  - Liked!
Going Deeper

- Maintainability - Makes developers lives easier
Going Deeper

- Maintainability - Makes developers lives easier
  - Defects are easy to identify
  - Changes are easy to understand & don't affect other components
  - The software is easy to test
Measuring Quality
How Can We Measure Quality?

These characteristics may be vague or ill defined. So how do we do it?
How Can We Measure Quality?

These characteristics may be vague or ill defined. So how do we do it?

- **Planning**
  - Decide what criteria are most important
  - Form a plan to assess them, directly or indirectly
How Can We Measure Quality?

These characteristics may be vague or ill defined.

So how do we do it?

- Planning
- Process
  - Code reviews help to improve maintainability & reduce bugs
  - Regular monitoring
How Can We Measure Quality?

These characteristics may be vague or ill defined. So how do we do it?

- Planning
- Process
- Testing!!
  - Show that you meet the minimum agreed requirements
  - Observe defects and performance issues.
Quality Processes
Quality as a Process

The quality of software comes from the quality of the process used to build it.
Quality as a Process

The quality of software comes from the quality of the process used to build it.

In every aspect of development:
- Requirements
- Specification
- Design
- Implementation
- Testing
- Documentation
- Maintenance
Quality as a Process

How can we assist the process?
Quality as a Process

How can we assist the process?

- **Synthetic** - Tools and techniques structured to create better software
- **Analytical** - Tools and techniques that measure the quality of software
Quality as a Process

How can we assist the process?

- **Synthetic**: Tools and techniques structured to create better software
- **Analytical**: Tools and techniques that measure the quality of software
- **Manual**: Driven interactively by developers
- **Automated**: Function w/o developer intervention
# Quality as a Process

<table>
<thead>
<tr>
<th>Manual</th>
<th>Synthetic</th>
<th>Analytical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• design methodologies</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• prototyping</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• coding standards</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• templates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• documentation standards</td>
<td></td>
</tr>
<tr>
<td>Automated</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Quality as a Process

<table>
<thead>
<tr>
<th>Manual</th>
<th>Synthetic</th>
<th>Analytical</th>
</tr>
</thead>
</table>
|        | • design methodologies  
|        |   • prototyping  
|        |   • coding standards  
|        |   • templates  
|        |   • documentation standards  
|        | • walk-throughs  
|        |   • inspections  
|        |   • audits  
| Automated | | |
# Quality as a Process

<table>
<thead>
<tr>
<th></th>
<th>Synthetic</th>
<th>Analytical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>• design methodologies&lt;br&gt; • prototyping&lt;br&gt; • coding standards&lt;br&gt; • templates&lt;br&gt; • documentation standards</td>
<td>• walk-throughs&lt;br&gt; • inspections&lt;br&gt; • audits</td>
</tr>
<tr>
<td>Automated</td>
<td>• program generators&lt;br&gt; • compilers&lt;br&gt; • development environments</td>
<td></td>
</tr>
</tbody>
</table>
## Quality as a Process

<table>
<thead>
<tr>
<th></th>
<th>Synthetic</th>
<th>Analytical</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manual</strong></td>
<td>• design methodologies</td>
<td>• walk-throughs</td>
</tr>
<tr>
<td></td>
<td>• prototyping</td>
<td>• inspections</td>
</tr>
<tr>
<td></td>
<td>• coding standards</td>
<td>• audits</td>
</tr>
<tr>
<td></td>
<td>• templates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• documentation standards</td>
<td></td>
</tr>
<tr>
<td><strong>Automated</strong></td>
<td>• program generators</td>
<td>• model checkers</td>
</tr>
<tr>
<td></td>
<td>• compilers</td>
<td>• program verifiers</td>
</tr>
<tr>
<td></td>
<td>• development environments</td>
<td>• program checkers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• unit testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integration testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• system testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Synthetic**
  - design methodologies
  - prototyping
  - coding standards
  - templates
  - documentation standards

- **Analytical**
  - walk-throughs
  - inspections
  - audits

- **Automated**
  - program generators
  - compilers
  - development environments
Quality as a Process

Integrating these into the process is a core part of the quality plan
Quality as a Process

Integrating these into the process is a core part of the quality plan

- Set goals
- Establish practices
- Measure progress
So What Will We Do?

- We'll mostly consider how analytical tools and techniques can help ensure quality.
So What Will We Do?

- We'll mostly consider how analytical tools and techniques can help ensure quality.
- Starting next time, with TESTING.