Techniques for Improving Software Quality

Software Quality
Quality Audit and Certification

Master in Computer Engineering

Roberto García (rgarcia@diei.udl.cat)
Introduction

- **Software quality assurance** is a planned and systematic **program of activities** designed to ensure that a system has the desired characteristics.

- In software quality, the best place to **focus** is on the **process**, incorporate **quality assurance components**.
Components of a Software Quality Program

• **Informal technical reviews.** Include *desk-checking* the design or the code or walking through the code with a few *peers*
  – Common practice *before* a *formal* review
Components of a Software Quality Program

• **Formal technical reviews.** Includes code inspections, peer reviews, customer reviews or external audits
  
  – When: the process goes through “quality gates”, periodic reviews that check quality before moving to next development stage

Requirements ➔ Design ➔ Development ➔ Deployment
Components of a Software Quality Program

• **Software-quality objectives.** Important to set explicit quality objectives, from among the external and internal characteristics
  – Without them, programmers might work to maximize characteristics different from the important ones
Components of a Software Quality Program

• Explicit quality-assurance activity. Organization must show programmers that quality is a priority and make the quality-assurance activity explicit
  – Usually, quality perceived as a secondary goal
  – Programmers that "complete" their programs quickly, are rewarded more than programmers looking for quality results
  – Consequently, programmers don't make quality their first priority
Components of a Software Quality Program

• **Testing strategy.** Develop a test strategy in conjunction with the product requirements, design and development
  – **Test:** compare what should the software do and what it actually does
  – Integrated into *development process*...
Testing Strategy
Components of a Software Quality Program

• **Software-engineering guidelines.** Guidelines should control the technical character of the software as it's developed
  – Guidelines apply to all software development activities, including problem definition, requirements development, architecture, coding and system testing
Components of a Software Quality Program

• **External audits.** Technical reviews performed by an **external team** to determine the status of a project or the quality of a product being developed.
Other components with quality effects

• They aren't explicitly quality-assurance activities but also affect software quality:
  – Change-control procedures
  – Measurement of results
  – Prototyping
Other components with quality effects

- **Change-control procedures.** Avoid **uncontrolled changes**, destabilize and degrade quality:
  - In requirements, disrupt design and coding
  - In design, code disagrees with requirements
  - In the code, inconsistencies and uncertainties about what has been reviewed and tested

- **Version Control Systems (VCS):**
  - Standalone (CVS, Subversion, Git,...)
  - Word processors (Microsoft Office, OpenOffice,...)
  - Content Management Systems (Drupal, Joomla,...)
  - Wikis (MediaWiki, TWiki, Trac (wiki+svn)...)
Other components with quality effects

• **Measurement of results.** Measure results of the quality-assurance **plan** to know whether it is working and perform controlled changes to improve it
  
  – Examples: error detection rates for each technique
Other components with quality effects

• **Prototyping.** The development of **realistic models** of a system's key functions
  – parts of a user interface to determine usability
  – critical calculations to determine execution time
  – typical data sets to check memory requirements

Prototyping can lead to better designs, better matches with user needs, and improved maintainability (Gordon and Bieman 1991)
Relative Effectiveness of Quality Techniques

• The various quality-assurance practices don't all have the same **effectiveness**

• Some practices are better at detecting defects than others, and **different methods** find **different kinds of defects**
  – Practices Effectiveness: percentage of defects they detect out of the total defects...
## Defect Detection Rates

<table>
<thead>
<tr>
<th>Defect-detection techniques</th>
<th>Rate</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Informal design reviews</td>
<td>25%</td>
<td>35%</td>
<td>40%</td>
</tr>
<tr>
<td>Formal design inspections</td>
<td>45%</td>
<td>55%</td>
<td>65%</td>
</tr>
<tr>
<td>Informal code reviews</td>
<td>20%</td>
<td>25%</td>
<td>35%</td>
</tr>
<tr>
<td>Formal code inspections</td>
<td>45%</td>
<td>60%</td>
<td>70%</td>
</tr>
<tr>
<td>Modeling or prototyping</td>
<td>35%</td>
<td>65%</td>
<td>80%</td>
</tr>
<tr>
<td>Personal desk-checking of code</td>
<td>20%</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>Unit test</td>
<td>15%</td>
<td>30%</td>
<td>50%</td>
</tr>
<tr>
<td>New function (component) test</td>
<td>20%</td>
<td>30%</td>
<td>35%</td>
</tr>
<tr>
<td>Integration test</td>
<td>25%</td>
<td>35%</td>
<td>40%</td>
</tr>
<tr>
<td>Regression test</td>
<td>15%</td>
<td>25%</td>
<td>30%</td>
</tr>
<tr>
<td>System test</td>
<td>25%</td>
<td>40%</td>
<td>55%</td>
</tr>
<tr>
<td>Low-volume beta test (&lt;10 sites)</td>
<td>25%</td>
<td>35%</td>
<td>40%</td>
</tr>
<tr>
<td>High-volume beta test (&gt;1,000 sites)</td>
<td>60%</td>
<td>75%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Source: Adapted from Programming Productivity (Jones 1986a), "Software Defect-Removal Efficiency" (Jones 1996), and "What We Have Learned About Fighting Defects" (Shull et al. 2002).
Defect Detection Rates

- Maximum modal rate 75%, average 40%
- Defect-detection methods work better in combination
  - Typical organization: test-heavy defect-removal approach → 85%
  - Leading organizations: combine techniques → 95%
- Human techniques (inspections, walk-throughs,...) better for certain errors than computer-based testing, and the opposite
  - For instance, code reading detects more interface defects and functional testing detected more control defects
Cost of Finding Defects

• **Some** defect-detection practices **cost more than others**
  – The most economical practices result in the least cost per defect found

• Most studies have found that inspections are cheaper than testing
Cost of Fixing Defects

- **The longer** a defect remains in the system, the **more expensive** it becomes to remove.
- An effective software-quality program includes a **combination of techniques** that apply to **all stages of development**.

**Recommendation:**
- Formal inspections of all requirements, all architecture and designs for system critical parts
- Modelling or prototyping
- Code reading or inspections
- Execution testing
Exercise

• For the proposed diagram for the Project Management Process of your company:

  1. Integrate into the process quality assurance components:
     • A measure/s of results of the quality assurance plan

  2. Indicate how you will make explicit the quality-assurance activity