Quality

- The *American Heritage Dictionary* defines **quality** as
  - “a characteristic or attribute of something.”

- For software, two kinds of quality may be encountered:
  - **Quality of design** encompasses requirements, specifications, and the design of the system.
  - **Quality of conformance** is an issue focused primarily on implementation.
  - **user satisfaction** = compliant product + good quality + delivery within budget and schedule
Software Quality

Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

Cost of Quality

- **Prevention costs** include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- **Internal failure costs** include
  - rework
  - repair
  - failure mode analysis
- **External failure costs** are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work
Role of the SQA Group-I

- Prepares an SQA plan for a project.
  - The plan identifies
    - evaluations to be performed
    - audits and reviews to be performed
    - standards that are applicable to the project
    - procedures for error reporting and tracking
    - documents to be produced by the SQA group
    - amount of feedback provided to the software project team

- Participates in the development of the project’s software process description.
  - The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.
Role of the SQA Group-II

- Reviews software engineering activities to verify compliance with the defined software process.
  - identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
- Audits designated software work products to verify compliance with those defined as part of the software process.
  - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
  - periodically reports the results of its work to the project manager.
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure.
- Records any noncompliance and reports to senior management.
  - Noncompliance items are tracked until they are resolved.

Why SQA Activities Pay Off?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Cost to Find and Fix a Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req.</td>
<td>0.75</td>
</tr>
<tr>
<td>Design Code</td>
<td>1.00</td>
</tr>
<tr>
<td>Test</td>
<td>1.50</td>
</tr>
<tr>
<td>System Test</td>
<td>3.00</td>
</tr>
<tr>
<td>Field Use</td>
<td>10.00</td>
</tr>
<tr>
<td>Overall</td>
<td>60.00-100.00</td>
</tr>
</tbody>
</table>

log scale

These courseware materials are to be used in conjunction with Software Engineering: A Practitioner’s Approach, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005.
Reviews & Inspections

... there is no particular reason why your friend and colleague cannot also be your sternest critic.

Jerry Weinberg

What Are Reviews?

- a meeting conducted by technical people for technical people
- a technical assessment of a work product created during the software engineering process
- a software quality assurance mechanism
- a training ground
What Reviews Are Not

- A project summary or progress assessment
- A meeting intended solely to impart information
- A mechanism for political or personal reprisal!

The Players

- review leader
- standards bearer (SQA)
- producer
- maintenance oracle
- recorder
- user rep
- reviewer
Conducting the Review

1. be prepared—evaluate product before the review
2. review the product, not the producer
3. keep your tone mild, ask questions instead of making accusations
4. stick to the review agenda
5. raise issues, don’t resolve them
6. avoid discussions of style—stick to technical correctness
7. schedule reviews as project tasks
8. record and report all review results

Review Options Matrix

<table>
<thead>
<tr>
<th></th>
<th>IPR</th>
<th>WT</th>
<th>IN</th>
<th>RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>trained leader</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>agenda established</td>
<td>maybe</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>reviewers prepare in advance</td>
<td>maybe</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>producer presents product</td>
<td>maybe</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>“reader” presents product</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>recorder takes notes</td>
<td>maybe</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>checklists used to find errors</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>errors categorized as found issues list created</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>issues list created</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>maybe</td>
</tr>
<tr>
<td>team must sign-off on result</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>maybe</td>
</tr>
</tbody>
</table>

*IPR—informal peer review  WT—Walkthrough  IN—Inspection  RRR—round robin review
**Sample-Driven Reviews (SDRs)**

- SDRs attempt to quantify those work products that are primary targets for full FTRs.

To accomplish this …

- Inspect a fraction $a_i$ of each software work product, $i$. Record the number of faults, $f_i$, found within $a_i$.
- Develop a gross estimate of the number of faults within work product $i$ by multiplying $f_i$ by $1/a_i$.
- Sort the work products in descending order according to the gross estimate of the number of faults in each.
- Focus available review resources on those work products that have the highest estimated number of faults.

**Metrics Derived from Reviews**

- inspection time per page of documentation
- inspection time per KLOC or FP
- inspection effort per KLOC or FP
- errors uncovered per reviewer hour
- errors uncovered per preparation hour
- errors uncovered per SE task (e.g., design)
- number of minor errors (e.g., typos)
- number of major errors (e.g., nonconformance to req.)
- number of errors found during preparation

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner’s Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005.
Statistical SQA

Collect information on all defects
Find the causes of the defects
Move to provide fixes for the process

... an understanding of how to improve quality ...

Six-Sigma for Software Engineering

- The term “six sigma” is derived from six standard deviations—3.4 instances (defects) per million occurrences—implying an extremely high quality standard.
- The Six Sigma methodology defines three core steps:
  - Define customer requirements and deliverables and project goals via well-defined methods of customer communication
  - Measure the existing process and its output to determine current quality performance (collect defect metrics)
  - Analyze defect metrics and determine the vital few causes.
  - Improve the process by eliminating the root causes of defects.
  - Control the process to ensure that future work does not reintroduce the causes of defects.
Software Reliability

- A simple measure of reliability is mean-time-between-failure (MTBF), where
  \[ \text{MTBF} = \text{MTTF} + \text{MTTR} \]
- The acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to-repair, respectively.
- **Software availability** is the probability that a program is operating according to requirements at a given point in time and is defined as
  \[ \text{Availability} = \left( \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \right) \times 100\% \]

Software Safety

- **Software safety** is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.
- If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.
ISO 9001:2000 Standard

- ISO 9001:2000 is the quality assurance standard that applies to software engineering.
- The standard contains 20 requirements that must be present for an effective quality assurance system.
- The requirements delineated by ISO 9001:2000 address topics such as:
  - management responsibility, quality system, contract review, design control, document and data control, product identification and traceability, process control, inspection and testing, corrective and preventive action, control of quality records, internal quality audits, training, servicing, and statistical techniques.