Quality Management
Goals of the Unit

• Understand the importance of quality management in software development projects

• Learn the main techniques to manage quality in projects

• Learn the main techniques to manage quality of project deliverables

• Understand the differences between software testing and quality management
Software Quality Assurance
Software Quality Assurance

**Software quality assurance** is the planned and systematic application of activities to ensure **conformance** of software life cycle processes and **products** to **requirements, standards, and procedures**.
Comments

• The definition applies both to the process and the products

• Quality assurance (like many other activities) is planned and systematic

• Conformance is required w.r.t. all the elements characterizing the software operational environment
Quality Assurance Process

• **Quality planning**, which identifies the relevant standard and practices and the way to implement them.

• **Quality assurance**, which focuses on ensuring that the project applies and follows the quality standards identified at the previous step.

• **Quality control**, which ensures that the products respect the quality standards identified during the planning phase.
Quality Planning
Quality Planning

• Goal: ensure the goals of quality management are met in a project

• Means:
  – Identification of constraints and quality goals in scope
  – Identification of standards and procedures to be applied
  – Identification of techniques to be applied
  – Allocation of resources (time, people, budget) to quality assurance activities
  – Roles and responsibilities

• Output: quality assurance planning document
Comments

• Quality needs to be balanced with the other project constraints (e.g. time and costs)

• Not all systems are equally critical: NASA, for instance distinguishes eight different classes of software systems

• The quality assurance team should be independent from the development team

• Different “levels” of independence:
  – different roles in the project
  – different structures in the organization
  – independent organizations
Quality Assurance & Quality Control
Quality Assurance

- **Goal:** ensure that the *project* applies and follows the quality standards
- **Main tool:** quality audits
- **Triggers:** time, milestones, or critical events in the project (according to the quality plan)
- **Quality audits include**
  - Inspections
  - Reviews
  - Walkthroughs
- **Output:**
  - Main findings and recommendations
Audit Meeting Organization

• Audit and review meetings are held to assess the status of a product or project

• Three “conflicting” roles:
  – The **auditors**: analysis of products and project documentation
  – The **project members**, responsible of providing clarifications and explain choices and project status
  – The **moderator**, who ensures the agenda is followed and the meeting environment remains productive
Auditing Process Structure

• Definition of the goals and boundaries of the audit
• Identification of the auditing committee (independence, competence, professionalism)
• Distribution of all the relevant material
• Preparation of the auditing by the auditors
• Auditing meeting
• Preparation of the final report
Signs of Troublesome Projects

• According to NASA signs of troublesome projects include:
  – Frequent changes in milestones
  – Unexplained fluctuations in personnel
  – Continued delays in software delivery
  – Unreasonable number of non conformance reports or change requests.
Quality Control

• **Goal:** ensure that the products respect the quality standards identified during the planning phase

• **Main tools:**
  – Inspections
  – Analyses
  – Testing

• **Triggers:** milestones or critical events in the project (according to the quality plan)

• **Output:**
  – List of non-conformance reports
Quality Control

• Quality control of software systems is extremely difficult, because:
  – of the enormous number of states a software system can be in (exhaustive testing is impractical/impossible)
  – the operating environment is unpredictable
  – discontinuity: little changes in inputs can cause enormous changes in outputs
  – non functional requirements can be difficult to assess (consider, e.g., maintainability, usability)
  – test automation can be difficult or very costly (consider, e.g., testing a GUI)
  – today’s systems are composed by using different technologies (e.g., HTML/CSS, Javascript, PHP, WebServer, OS)
Quality Control Techniques for Software

• **Walkthroughs and code inspections:**
  - the system is analyzed by an independent team

• **Analyses:**
  - **static checkers** verify the correct use of certain syntactic constructs (e.g., no assignments in conditions)
  - **dynamic checkers** verify anomalies and suspect situations by executing instrumented code
  - **code metrics** are collected to measure other quality characteristics (e.g., comments/lines; unit test coverage)
  - **formal verification** (theorem proving, model checking) proves properties about abstract representations of the system

• **Testing:**
  - tests are performed on a system to verify the behavior under specific circumstances
Establishing a Metrics Program
Establishing a Metrics Program

• A **metrics collection program** quantitatively assesses how the project goals are being achieved

• **Process metrics:** measure different characteristics of a project

• **Product metrics:** measure different characteristics of a project product

• Trends often more important than point-wise numbers

• Better if automated
Product Metrics: Size

- **Size oriented metrics:**
  - Source lines of code
  - Number of classes

- **Function oriented metrics:**
  - Function Points
  - Object Points

- **Comments:**
  - Size metrics can be automatically collected
  - The count of SLOC, however, is “controversial” (see next slide)
  - Function oriented metrics requires trained personnel for their collection
Product Metrics: Complexity

- Cyclomatic complexity
- Coupling between objects
- Depth of inheritance
- Fan-in, fan-out
Product Metrics: Quality Metrics

• **Ratio between lines of comments and lines of codes** (Indication of the maintainability of a system)

• **Cumulative number of open issues** (It measures whether the project is “converging”)

• **Error density**, that is, the number of errors found per source line of code. (It helps understand whether the development process has some systematic faults.)