SOFTWARE QUALITY

CPTS 583

Software Quality Planning

Outline

- Software quality management
  - Quality planning
  - Product quality in relation to process quality
- Software quality plan
  - Elements
  - Planning steps
- Quality planning in practice
  - For small projects
  - Reduced plan versus no plan

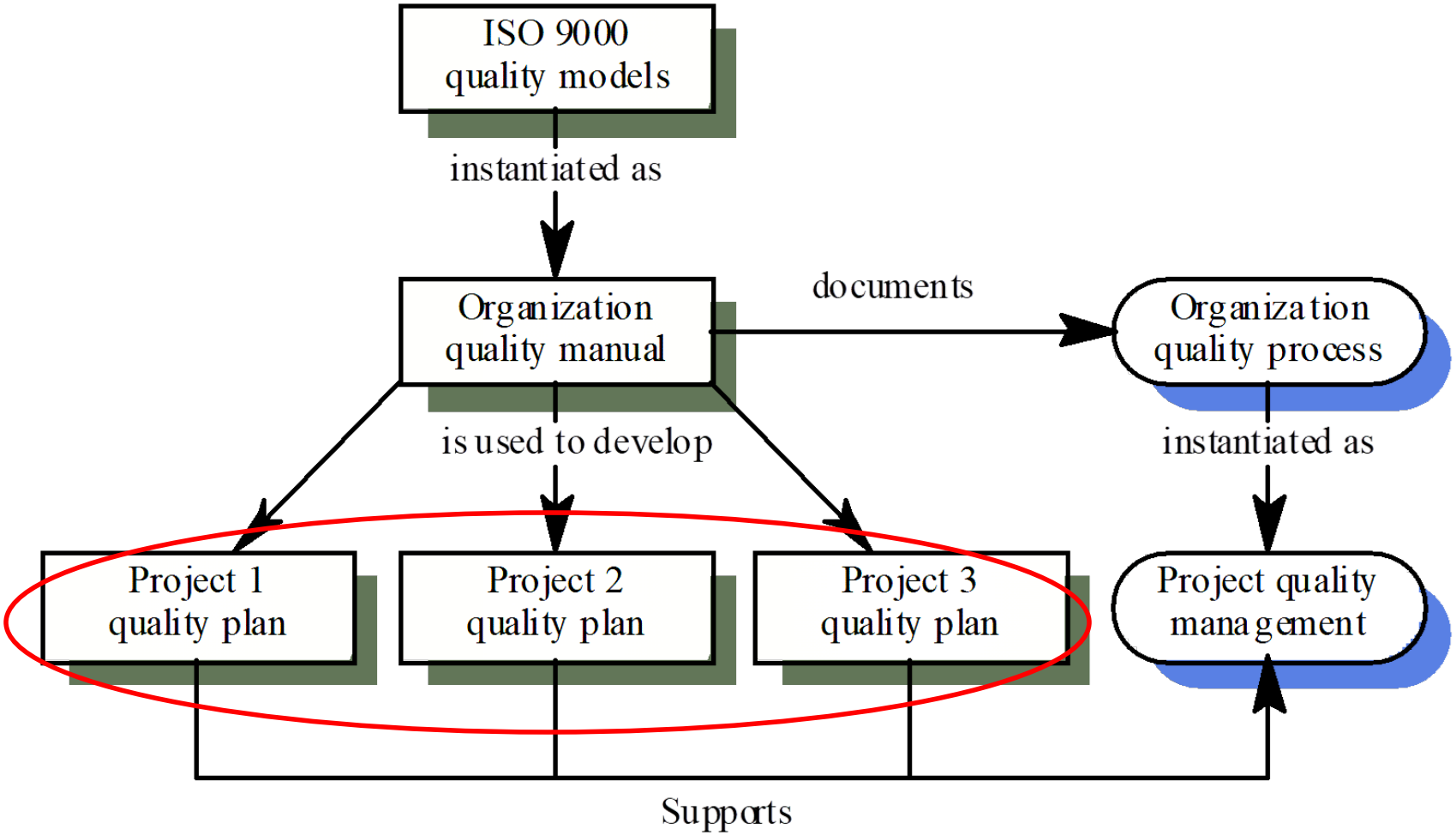
Software Quality Management

- Quality Planning
- Quality Assurance
- Quality Control

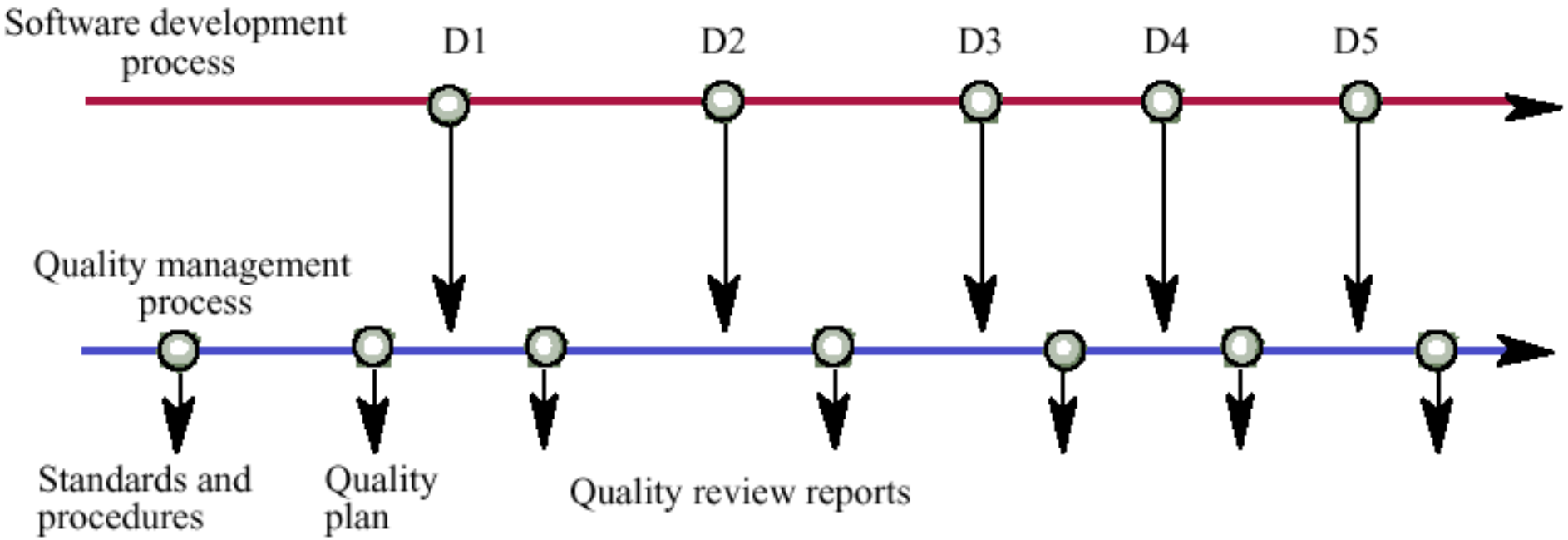
Quality Management Activities

- ***Quality planning***
  - selecting and modifying applicable quality standards and procedures for a particular project
- Quality assurance
  - establishing organizational quality standards and procedures
- Quality control
  - ensuring quality standards and procedures are followed by development team

Quality Management Activities



Quality Management along the way



Process and product quality

- The quality of a developed product is influenced by the quality of the production process
- Particularly important in software development as some product quality attributes are hard to assess
- Relationship between software processes and product quality: complex yet poorly understood

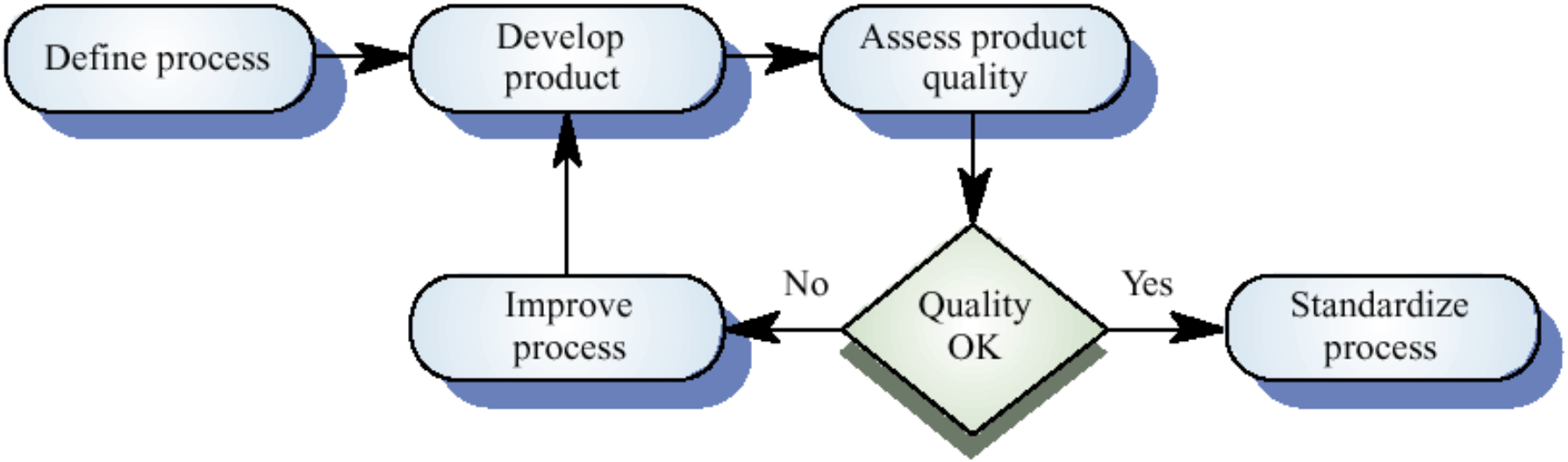
Process-based quality

- Straightforward link between process and product in manufactured goods

- More complex for software because:
  - The application of individual skills and experience is particularly important in software development
  - External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality

- Care must be taken not to impose inappropriate process standards

Process-based quality



Quality planning

- A quality plan sets out the desired product qualities and how these are assessed and define the most significant quality attributes
  - set out which organisational standards should be applied and, if necessary, define new standards
  - define the quality assessment process

Elements of a software quality plan

- ✓Product introduction/description
- ✓Quality goals
- ✓Review activities
- ✓Tests
- ✓Configuration tools/procedures

Quality Goals

Quality requirements of the developed software.

- Safety
- Security
- Reliability
- Resilience
- Robustness
- Understandability
- Testability
- Adaptability

- Modularity
- Complexity
- Portability
- Usability
- Accessibility
- Reusability
- Efficiency
- Learnability

Quality Goals

- Quantitative measures usually preferred to qualitative measures when choosing goals
  - easier to assess objectively during testing.
- Quality goals should reflect the major acceptance criteria found in the requirement's document (i.e. RFP)
  - correctness, reliability, robustness, maintainability….
- RFP is often used to measure successful achievement of the customer's quality requirements.

Product Quality Goals

- Example: *a warehouse safety monitoring system*

Quality requirements

- The system needs to work continuously
- The system should be highly user-friendly
- The system must be very reliable
- The system should provide highly quality service
- The System must be very efficient

Quality goals

- If the system fails, it must recover within <5 minutes
- A new user should be able to learn how to operate the system in 3 days
- The system must be working correctly 98% of the time
- The system should provide right results with 99% accuracy
- The system should respond to safety condition changes and send reports in 2 seconds

Process Quality Goals

Purpose: To (*characterize, evaluate, predict, monitor, etc.*) the (*process, product, model, metric, etc.*) in order to (*understand, plan, assess, manage, control, engineer, learn, improve, etc.*) it.

Example: To *evaluate* the *maintenance procedure* in order to *improve* it.

Perspective: Examine the (*cost, effectiveness, correctness, defects, changes, product measures, etc.*) from the viewpoint of the (*developer, manager, customer, etc.*)

Example: Examine the *effectiveness* from the viewpoint of the *customer*.

Environment: The environment consists of the following: process factors, people factors, methods, tools, constraints, etc.

Example: The maintenance staff are poorly motivated programmers who have limited access to tools.

Review Activities

- Design reviews (DR)
- Design inspections
- Managerial reviews
- Code inspections

Review Activities

- Scope – what does it cover
- Type – emphasis – managerial, technical, super detailed…
- Schedule – often based on previous reviews and outcomes
- Procedures – action lists; present and discuss
- Reviewers – who will participate in the review
- Responsibilities – what each reviewer would be supposed to do for the review; what documents would be needed, by when…

Tests

- Test Scope - unit, integration, system, subsystem….
- Type of test - may include computer-generated tests and their application via test suites, and more
- Test Schedule – prioritized and follow up
- Test procedure (for different types of tests…)
- Tester - Who is responsible for carrying out tests
  - Notification, time, date, materials, facilities, etc.
  - Different people responsible at different times

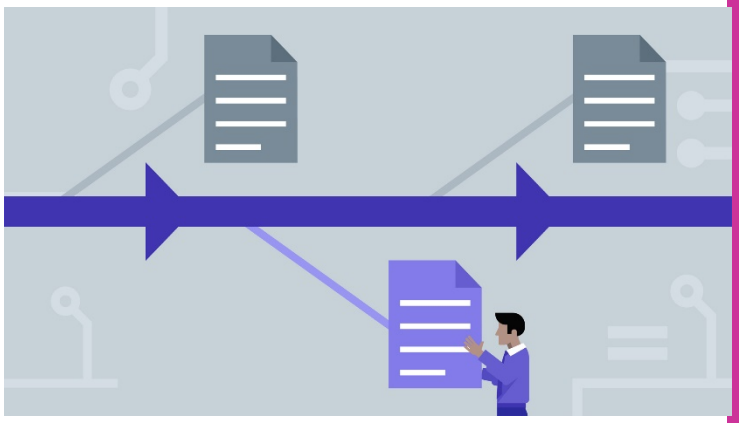
Tests

Acceptance Tests for External Software/Components

- run in parallel with internally-developed software tests
- in the plan, list
  - software/component purchased
  - software/component developed by subcontractors
  - customer-supplied software/component
- for each acceptance test
  - similar per-test info

Configuration tools / procedures

- Configuration Management
  - Tools to be used
    - E.g., version control tools
  - Procedures to be followed



Quality planning in practice

- Preparing plans can be a hassle
  - Too many
  - Too bureaucratic
- Agile versus heavy-weight planning
  - Avoid being "*plan-centric*"
  - Heavy-weight plan may be unnecessary or infeasible

Quality planning for small projects

- A project of short duration (e.g., 10 days)
- A project to be worked on by a small team (e.g., 3 professionals)
- A project that would not cost much even timeline failed to be observed
- ……

Quality planning for small projects

- Simplified/reduced quality plan
  - Quality goals

- Reduction is NOT dismissal
  - Advantages of quality planning
    - Even for small projects!

Quality planning: why not dismiss

- General advantages over 'no plan'

1. Gaining a more comprehensive / thorough understanding of quality assurance/control tasks
2. Assigning greater responsibility for meeting obligations
3. Easier to share control of the project and identify unexpected delays
4. Better understanding of requirements and timetable

Quality planning: why not dismiss

- Benefits for customers

1. smaller deviations from planned completion dates
2. smaller budget overruns
3. better control over development process – problems can be addressed locally
4. Fewer delay damages

Quality planning: why not dismiss

- Benefits for software business/organization

1. reduced risk of market loss
2. reduced risk of litigation (late arrival; non-compliance)
3. reduced risk of impairing a firm's reputation
4. reduced risk of requesting a budget supplement.

Quality planning: why not dismiss

- Problems with no plan

1. Product/process errors
2. cost overruns
3. finger pointing
4. missed dates
5. internal friction among cooperating parties

Summary

- Software quality management: overview
  - planning, assurance, control
  - difference and connection
- Quality planning strategies: what and how
  - Product and process introduction/description
  - Quality goals
  - Review activities
  - Software tests
  - Configuration management
- Practical issues in quality planning
  - Reducing, but not dismissing, the plan even for small projects