# Test and Analysis Activities within a Software Process

# Learning objectives

- Identify how quality is incorporated into the development process
- Build an overall picture of the quality process
- Identify the main characteristics of a quality process
  - visibility
  - anticipation of activities
  - feedback

# Software Qualities and Process

- Qualities cannot be added after development
  - Quality results from a set of inter-dependent activities
  - Analysis and testing are crucial but far from sufficient.
- Testing is not a phase, but a lifestyle
  - Testing and analysis activities occur from early in requirements engineering through delivery and subsequent evolution.
  - Quality depends on every part of the software process
- An essential feature of software processes is that software test and analysis is thoroughly integrated and not an afterthought

# The Quality Process

- Quality process: set of activities and responsibilities
  - focused primarily on ensuring adequate dependability
  - concerned with project schedule or with product usability
- The quality process provides a framework for
  - selecting and arranging activities
  - considering interactions and trade-offs with other important goals.

# Interactions and Tradeoffs

*example*

**high dependability vs. time to market**

- Mass market products:
  - better to achieve a reasonably high degree of dependability on a tight schedule than to achieve ultra-high dependability on a much longer schedule
- Critical medical devices:
  - better to achieve ultra-high dependability on a much longer schedule than a reasonably high degree of dependability on a tight schedule

# Properties of the Quality Process

- **Completeness**: Appropriate activities are planned to detect each important class of faults.
- **Timeliness**: Faults are detected at a point of high leverage (as early as possible)
- **Cost-effectiveness**: Activities are chosen depending on cost and effectiveness
  - cost must be considered over the whole development cycle and product life
  - the dominant factor is usually the cost of repeating an activity through many change cycles.

# Planning and Monitoring

- The quality process
  - Balances several activities across the whole development process
  - Selects and arranges them to be as cost-effective as possible
  - Improves early visibility
- Quality goals can be achieved only through careful planning
- Planning is integral to the quality process

# Process Visibility

- A process is visible to the extent that one can answer the question
  - How does our progress compare to our plan?
  - Example: Are we on schedule? How far ahead or behind?
- The quality process has not achieved adequate visibility if one cannot gain strong confidence in the quality of the software system before it reaches final testing
  - quality activities are usually placed as early as possible
    - design test cases at the earliest opportunity (not ``just in time'')
    - uses analysis techniques on software artifacts produced before actual code.
  - motivates the use of “proxy” measures
    - Ex: the number of faults in design or code is not a true measure of reliability, but we may count faults discovered in design inspections as an early indicator of potential quality problems

# A&T Strategy

- Identifies company- or project-wide standards that must be satisfied
  - procedures required, e.g., for obtaining quality certificates
  - techniques and tools that must be used
  - documents that must be produced

# A&T Plan

- A comprehensive description of the quality process that includes:
  - objectives and scope of A&T activities
  - documents and other items that must be available
  - items to be tested
  - features to be tested and not to be tested
  - analysis and test activities
  - staff involved in A&T
  - constraints
  - pass and fail criteria
  - schedule
  - deliverables
  - hardware and software requirements
  - risks and contingencies

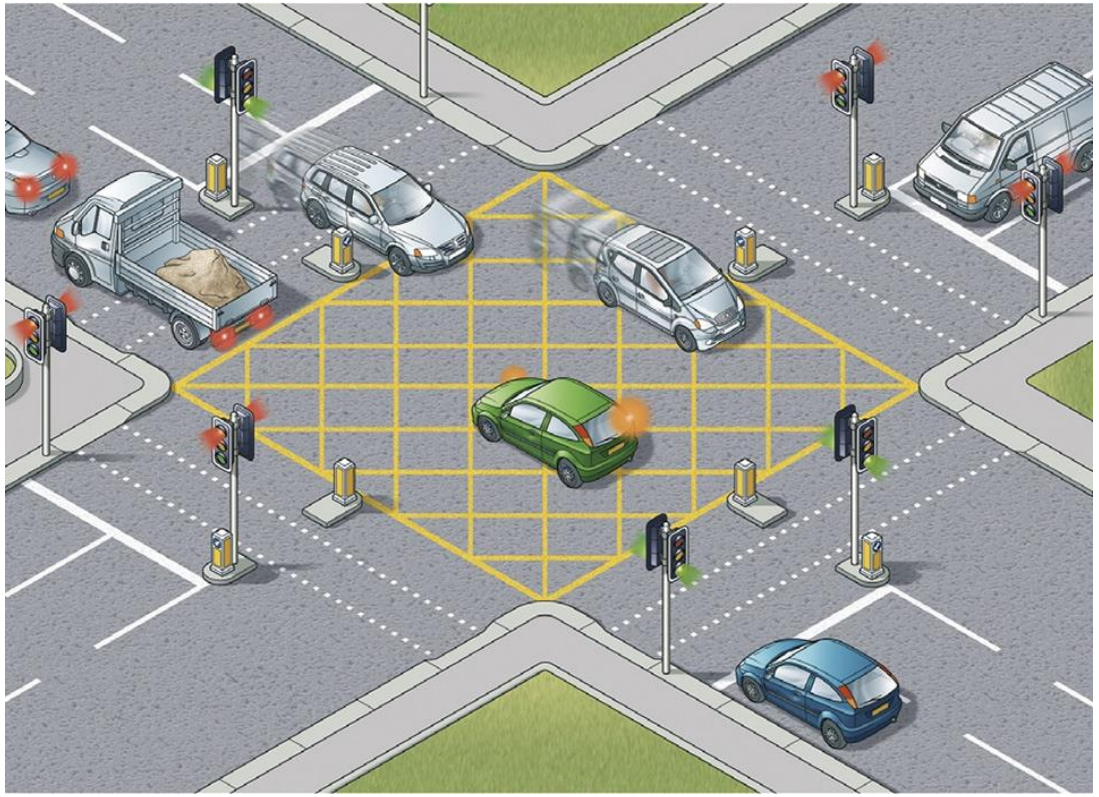
# Quality Goals

- Process qualities (visibility,....)
- Product qualities
  - internal qualities (maintainability,....)
  - external qualities
    - usefulness qualities:
      - usability, performance, security, portability, interoperability
    - dependability
      - correctness, reliability, safety, robustness

# Dependability Qualities

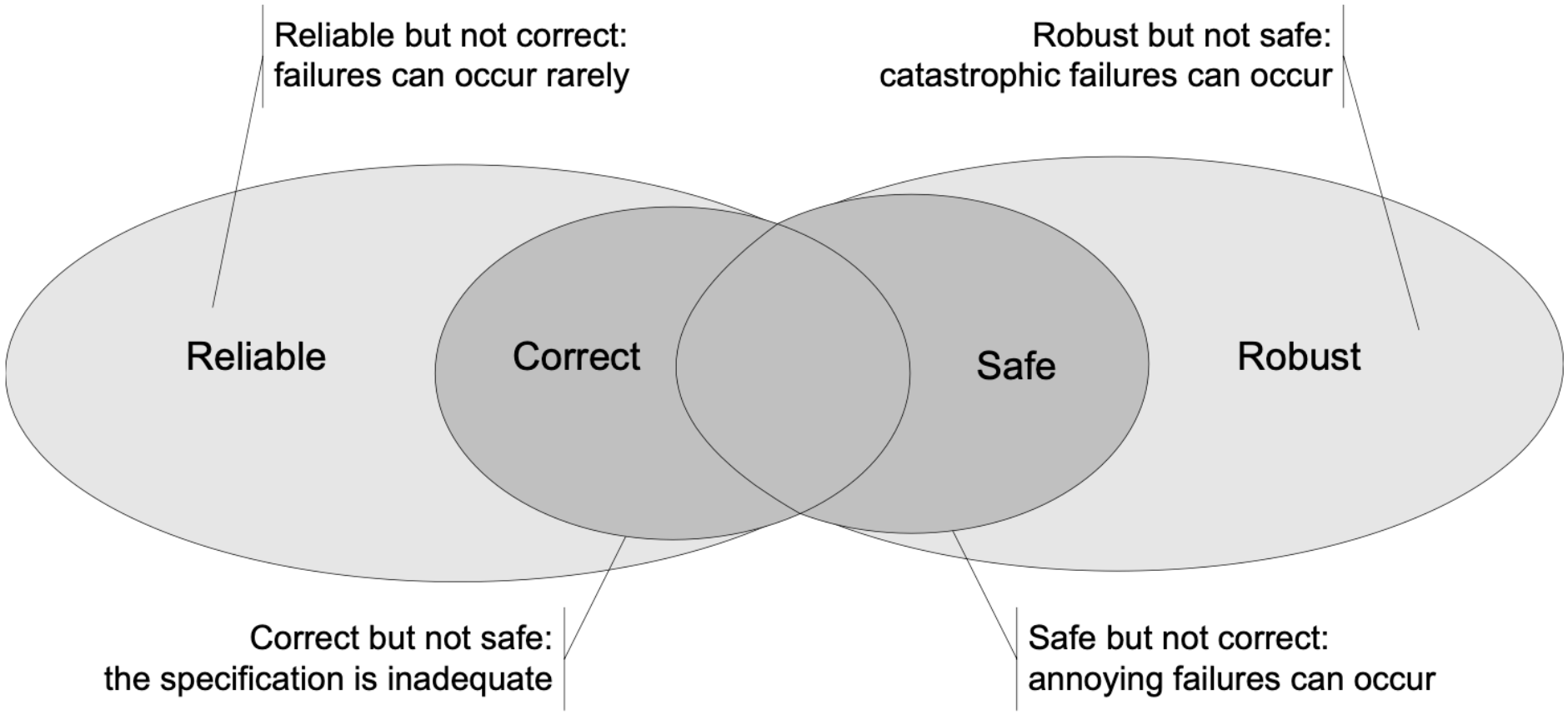
- Correctness:
  - A program is correct if it is consistent with its specification
    - seldom practical for non-trivial systems
- Reliability:
  - likelihood of correct function for some ``unit'' of behavior
    - relative to a specification and usage profile
    - statistical approximation to correctness (100% reliable = correct)
- Safety:
  - preventing hazards
- Robustness
  - acceptable (degraded) behavior under extreme conditions

# Example of Dependability Qualities



- **Correctness, reliability:** let traffic pass according to correct pattern and central scheduling
- **Robustness, safety:** Provide degraded function when possible; never signal conflicting greens.
    - Blinking red / blinking yellow is better than no lights; no lights is better than conflicting greens

# Relationships of Dependability Qualities



# Analysis

- analysis includes
  - manual inspection techniques
  - automated analyses
- can be applied at any development stage
- particularly well suited at the early stages of specifications an design

# Inspection

- can be applied to essentially any document
  - requirements statements
  - architectural and detailed design documents
  - test plans and test cases
  - program source code
- may also have secondary benefits
  - spreading good practices
  - instilling shared standards of quality.
- takes a considerable amount of time
- re-inspecting a changed component can be expensive
- used primarily
  - where other techniques are inapplicable
  - where other techniques do not provide sufficient coverage

# Automatic Static Analysis

- More limited in applicability
  - can be applied to some formal representations of requirements models
  - not to natural language documents
- are selected when available
  - substituting machine cycles for human effort makes them particularly cost-effective.

# Testing

- Formerly executed late in development but with Test-Driven Development can play a large part.
- Start as early as possible
- Early test generation has several advantages
  - Tests generated independently from code, when the specifications are fresh in the mind of analysts
  - The generation of test cases may highlight inconsistencies and incompleteness of the corresponding specifications
  - tests may be used as compendium of the specifications by the programmers

# Improving the Process

- Long lasting errors are common
- It is important to structure the process for
  - Identifying the most critical persistent faults
  - tracking them to frequent errors
  - adjusting the development and quality processes to eliminate errors
- Feedback mechanisms are the main ingredient of the quality process for identifying and removing errors

# Organizational factors

- Different teams for development and quality?
  - separate development and quality teams is common in large organizations
  - indistinguishable roles is postulated by some methodologies (extreme programming)
- Different roles for development and quality?
  - test designer is a specific role in many organizations
  - mobility of people and roles by rotating engineers over development and testing tasks among different projects is a possible option

# Example of Allocation of Responsibilities

- Allocating tasks and responsibilites is a complex job: we can allocate
  - Unit testing
    - to the development team (requires detailed knowledge of the code)
    - but the quality team may control the results (structural coverage)
  - Integration, system and acceptance testing
    - to the quality team
    - but the development team may produce scaffolding and oracles
  - Inspection and walk-through
    - to mixed teams
  - Regression testing
    - to quality and maintenance teams
  - Process improvement related activities
    - to external specialists interacting with all teams

# Allocation of Responsibilities and rewarding mechanisms: case A

- allocation of responsibilities
  - Development team responsible development  measured with LOC per person month
  - Quality team responsible for quality
- possible  effect
  - Development team tries to maximize productivity, without considering quality
  - Quality team will not have enough resources for bad quality products
- result
  - product of bad quality and overall project failure

# Allocation of Responsibilities and rewarding mechanisms: case B

- allocation of responsibilities
  - Development team responsible for both development and quality control
- possible effect
  - the problem of case A is solved
  - but the team may delay testing for development without leaving enough resources for testing
- result
  - delivery of a not fully tested product and overall project failure

# Summary

- Test and Analysis are complex activities that must be suitably planned and monitored
- A good quality process obeys some basic principles:
  - visibility
  - early activities
  - feedback
- aims at
  - reducing occurrences of faults
  - assessing the product dependability before delivery
  - improving the process