COMP 311: Software Testing and Quality Assurance

Testing End to End

Week 4-5

Characteristics of Software Quality
Categories of tests
Metrics
Testing during maintenance

Software Quality Characteristics ISO/IEC 9126:2001

- Functionality
  - Suitability, Accuracy, Interoperability, Compliance, Security
- Reliability
  - Maturity, Recoverability, Fault tolerance
- Usability
  - Learnability, Understandability, Operability
- Efficiency
  - Resource efficiency, Time efficiency
- Maintainability
  - Stability, Analysability, Changeability, Testability
- Portability
  - Installability, Replaceabiltiy, Adaptability, Conformance

Functionality Characteristics

- Suitability
  - The essential Functionality characteristic: the appropriateness (to specification) of the functions of the software.
- Accuracy
  - Correctness of the functions
    Example: an ATM may provide a cash dispensing function but is the amount correct?
- Interoperability
  - A given software component or system does not typically function in isolation:
    the ability of a software component to interact with other components or systems
- Compliance
  - Where appropriate certain industry or government laws and guidelines need to be complied with, such as SOX
    This subcharacteristic addresses the compliant capability of software
- Security
  - Relates to unauthorized access to the software functions

❖ From http://www.sqa.net/iso9126.html

Reliability Characteristics

- Maturity
  - Relates to frequency of failure of the software
- Recoverability
  - Ability to bring back a failed system to full operation, including data and network connections
- Fault tolerance
  - The ability of software to withstand and recover from component, or environmental, failure

❖ From http://www.sqa.net/iso9126.html

Usability Characteristics

- Understandability
  - Determines the ease of which the systems functions can be understood, relates to user mental models in Human Computer Interaction methods
- Learnability
  - Learning effort for different users, such as novice, expert, casual …
- Operability
  - Ability of the software to be easily operated by a given user in a given environment.

❖ From http://www.sqa.net/iso9126.html

Efficiency Characteristics

- Time behaviour
  - Characterizes response times for a given throughput, such as transaction rate
- Resource behaviour
  - Characterizes resources used. Examples: memory, cpu, disk and network usage...

- From http://www.sqa.net/iso9126.html

Maintainability Characteristics

- Analyzability
  - Ability to identify the root cause of a failure within the software
- Changability
  - Amount of effort to change a system
- Stability
  - Sensitivity to change of a given system that is the negative impact that may be caused by system changes
- Testability
  - Effort needed to verify (test) a system change

- From http://www.sqa.net/iso9126.html

Portability Characteristics

- Adaptability
  - Ability of the system to change to new specifications or operating environments
- Installability
  - Effort required to install the software
- Conformance
  - Similar to compliance for functionality, but relates to portability
  - Example: Open SQL conformance relates to portability of database uses
- Replaceability
  - Characterizes the *plug and play* aspect of software components:
    how easy is it to exchange a given software component within a specified environment?

- From http://www.sqa.net/iso9126.html

Categories of testing

Functional and non-functional

Static and dynamic

Black box and white box

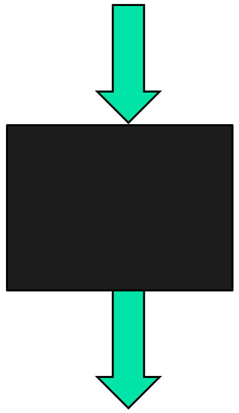
Functional testing

- Does system meet functional specification?
  - ISO 9126 quality characteristics:
    suitability, interoperability, securty, accuracy, complaince
- Applies to all four levels of testing
  - Unit Integration System Acceptance
- Tests can be based on
  - Requirements
    - Tasks or use cases focus on user interaction with system
  - Business processes
    - Business process modeling (BPM) focus on workflows

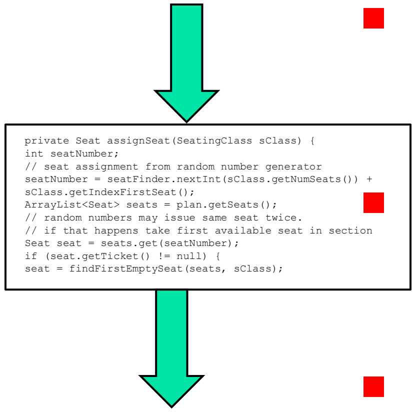
Non-functional testing

- Applies to quality characteristics:
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability
- Recall the 6th principle of testing
  - **Testing is context dependent**
  - Test for different qualities with very different tests
- Is performance / stress / load testing of Web applciations functional or non-functional?

Black and white box testing



- **Black box**
  - Dynamic tests
  - Make no assumptions how code is written
  - Verify results for different test data
  - Functional and non-functional characteristics



```
private Seat assignSeat(SeatingClass sClass) {
int seatNumber;
// seat assignment from random number generator
seatNumber = seatFinder.nextInt(sClass.getNumSeats()) +
sClass.getIndexFirstSeat();
ArrayList<Seat> seats = plan.getSeats();
// random numbers may issue same seat twice.
// if that happens take first available seat in section
Seat seat = seats.get(seatNumber);
if (seat.getTicket() != null) {
seat = findFirstEmptySeat(seats, sClass);
```

- **White box**
  - Evaluating documents
    - Requirements, Design, Plan …
  - Evaluating/analyzing code
    - Standards, readability, comments, clarity
  - Structural testing or coverage

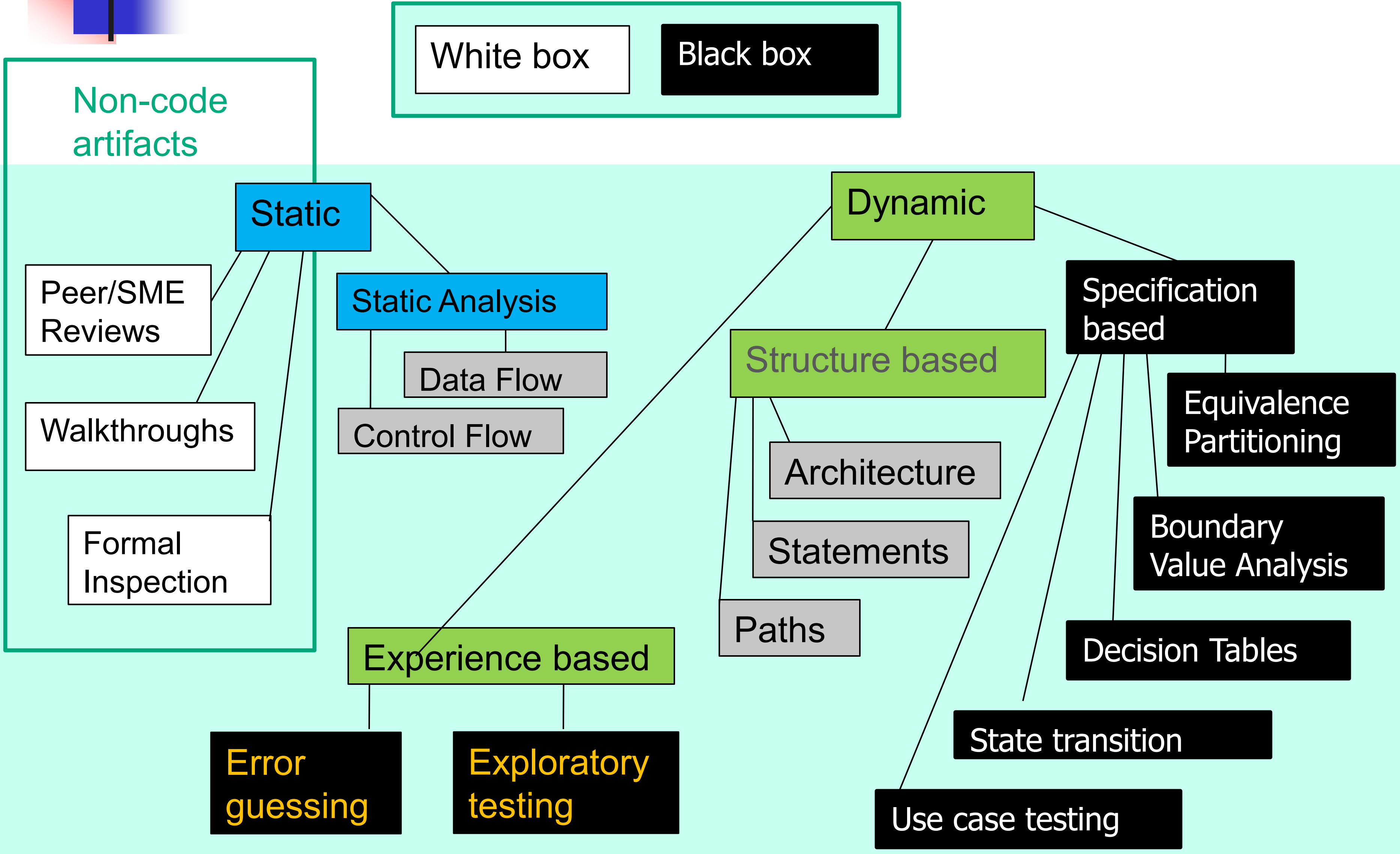
Black Box Testing

- Tests focus on external behavior
  - Specify and set up **preconditions**
    - Prepare test data
  - Perform an action
    - Typically user input or user-triggered event
    - Could be initiated by other systems
  - Compare results to **postconditions**
    - Output
    - Changes to persistent data
    - Internal state of the system
  - A deviation from expected result is a potential defect

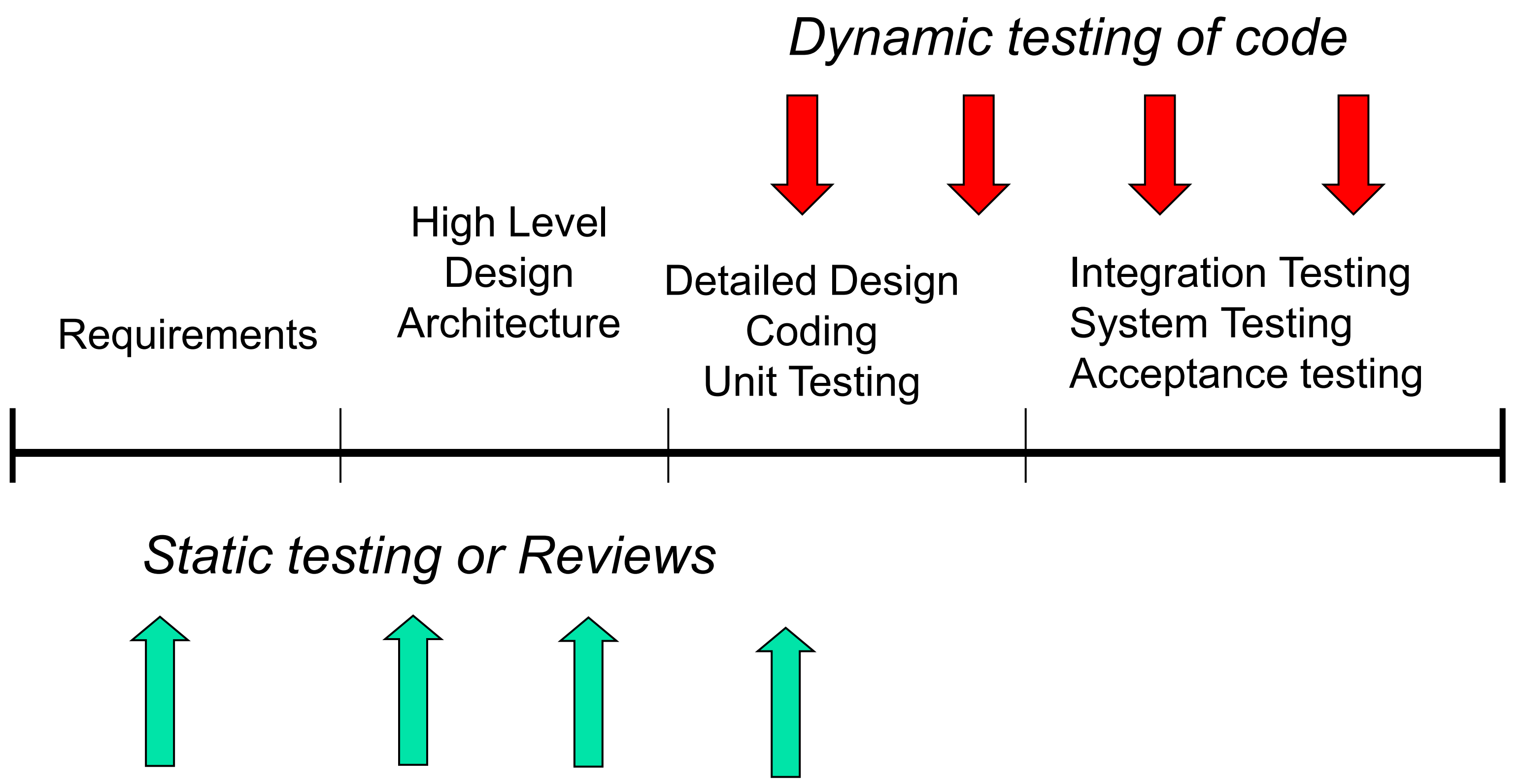
Testing all artifacts

- **Static tests**
  - Test without running code
  - Applies to documents as well as code
- **Dynamic tests**
  - Test by running code
  - Put the system into action
    - Requires well defined:
      - Test environment
      - Version or configuration of code to test
      - Test tools or test harness and script

Overview of testing techniques



Testing in the development life cycle



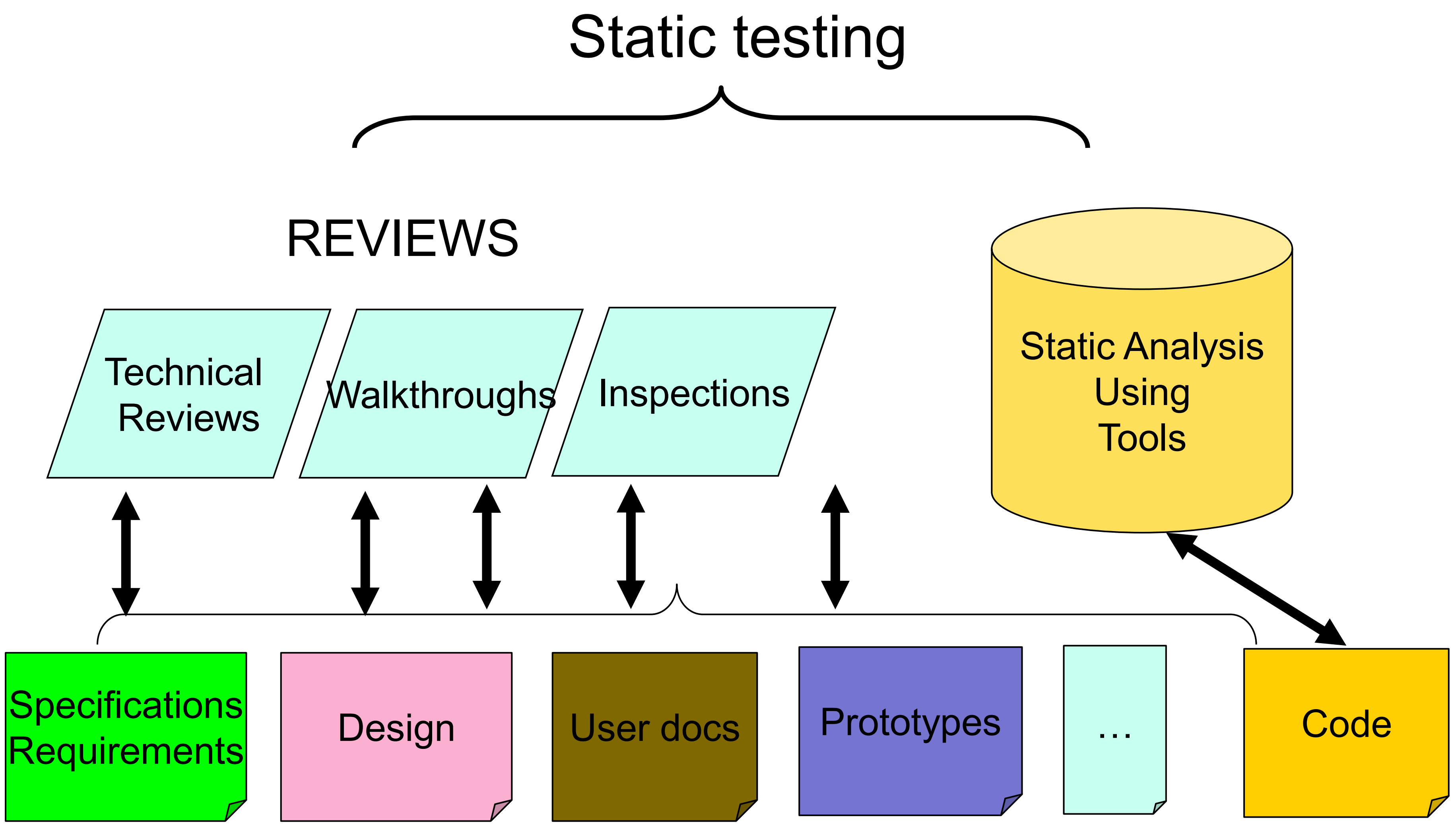
Dynamic testing of code
High Level Design Architecture
Detailed Design Coding Unit Testing
Requirements
Integration Testing System Testing Acceptance testing
Static testing or Reviews

Benefits of static testing

- Can start early, before there is code
  - Work products are examined

    Including plans and processes and tools
- Early validation
  - Reduced cost of rework
  - Relatively cheap improvement in quality
- Communication, information exchange, education
  - Team participation in review can include users
  - Established channel for feedback
- Increased awareness of quality issues

White Box Tests



The ultimate goal of testing code

- Black box
  - Functional: ensure software meets all requirements
    - Reducing bugs/defects to low risk or acceptable levels
  - Non-functional
    - Performance, robustness …
  - The unattainable goal:
    - All possible inputs and combinations of inputs
    - All possbile user actions in all possible orders
- White box
  - Maximize code coverage
    - All statements executed/All paths through code followed
    - Monitor creation, change in value/state, destruction of objects
  - Non-functional characteristis
    - Maintainability, portability, efficiency, …

A first look at test metrics

- How thoroughly is component or system tested
- Coverage usually answers as a %
  - White Box: code coverage
    - Has every line of code been exercised?
    - Has every path through the code been followed?
    - Has every part of every document been reviewed?
  - Black Box: test coverage
    - How many tests were identified? How many were run?
    - How many tests were successful?
    - How many defects were found? What is the rate of new defects?
- How meaningful are these metrics:
  - Do test objectives capture user requirements and expectations?
  - Do test criteria measure test objectives?
  - Does the test environment accurately replace real world use?

Testing during maintenance

70-90% of the lifespan of the software

Testing after general availability

After installation

- Maintainability is the ease with which a software product may be changed to:
  - Correct defects
  - Modified to meet new requirements
  - Refactored to make future maintenance easier
  - Adapted to a new environment
- Generally, the maintenance testing process is similar to development testing process
  - May last much longer
  - Often carried out by different resources

Triggers for maintenance testing

- Planned modifications
  - May be release based
  - May relate to environment change such as new database, OS upgrade, migration
- Ad-hoc
  - Defects requiring quick solution
  - Often hard to take a structured approach
  - Often implemented as patches later replaced by robust fix

Effectives maintenance testing

- Usually testing in mainenance is testing changes
  - Confirmation test
    - Change has desired effect: typically does a fix remove a defect
  - Regression test
    - Set of changes has no undesired side effect: don’t break code
- Impact analysis is important
  - What may be affected by a change?
  - Use risk analysis to focus areas of testing?
- Common challenges
  - Missing or inadequate specifications / documentation
  - Missing test cases or history of updates
  - Poor maintainability quality characteristics of code

What do you think?

- Does development stop when maintenance starts?
- How do fixes to defects found by customers and fixed by mainentance get fed back into products?
- Must development and post-production – maintenance/support/operations – be separate silos within the organization?