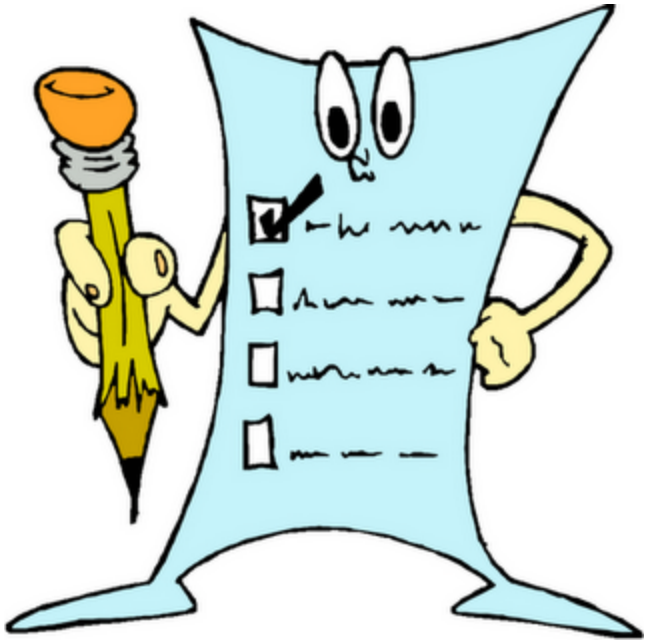
Software Testing

Materials adapted from Software Engineering by Ian Sommerville

Topics covered

- System testing
- Component testing
- Test case design
- Test automation



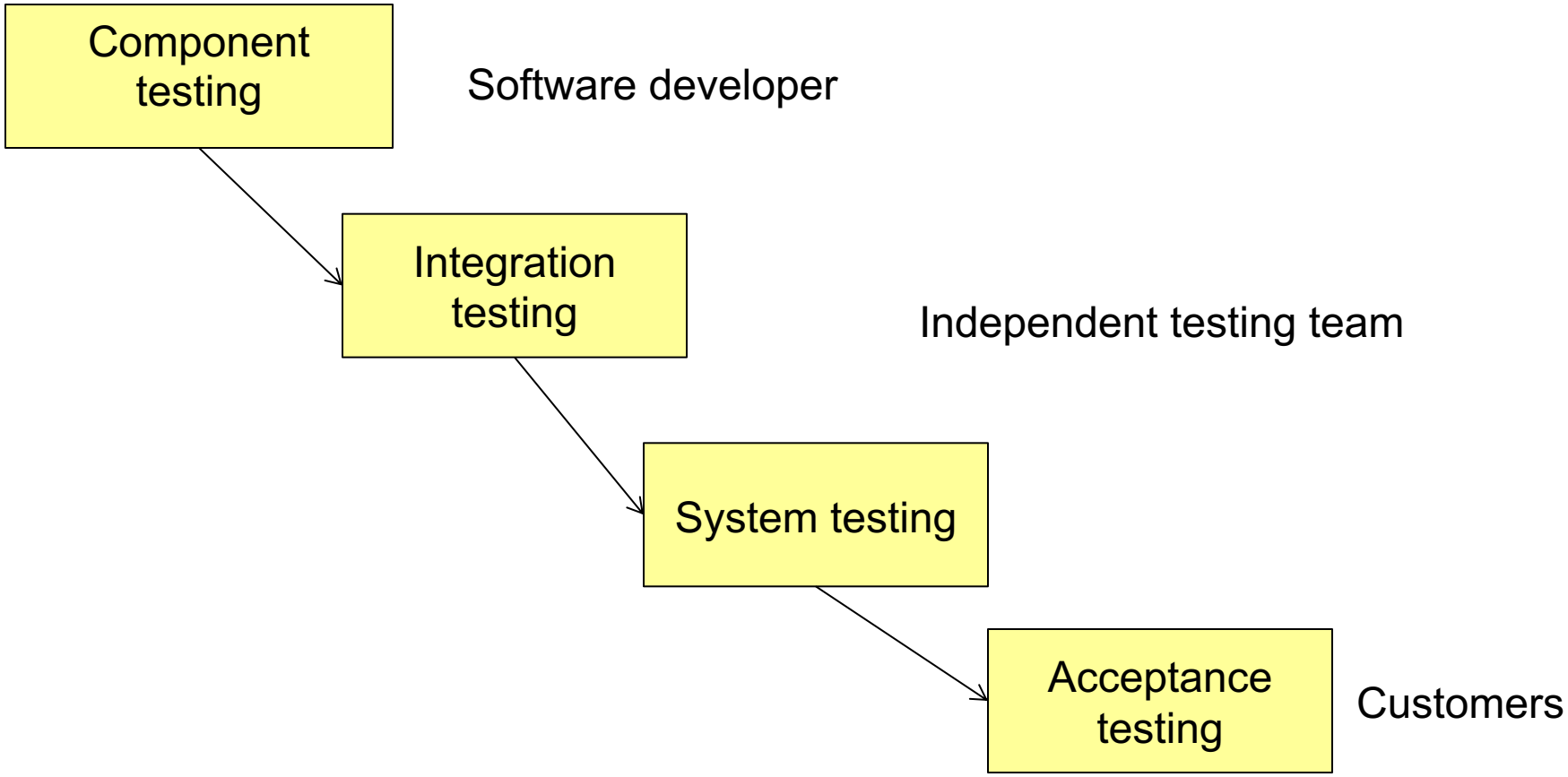
Testing Levels

- Component/unit testing
  - Testing of individual program components
  - Usually the responsibility of component developers
  - Tests are derived from the developer's experience
- System testing
  - Testing of groups of components integrated to create a system or sub-system
  - Typically the responsibility of an independent testing team
  - Tests are based on requirements and system specifications
- Acceptance testing
  - Testing the system as a whole to accept the system or not
  - Done by customers

Types of testing

- Functional testing
- Performance testing
  - Load testing
  - Stress testing
- Security testing
- Usability/interface testing
- API and service testing
- Privacy testing
- …

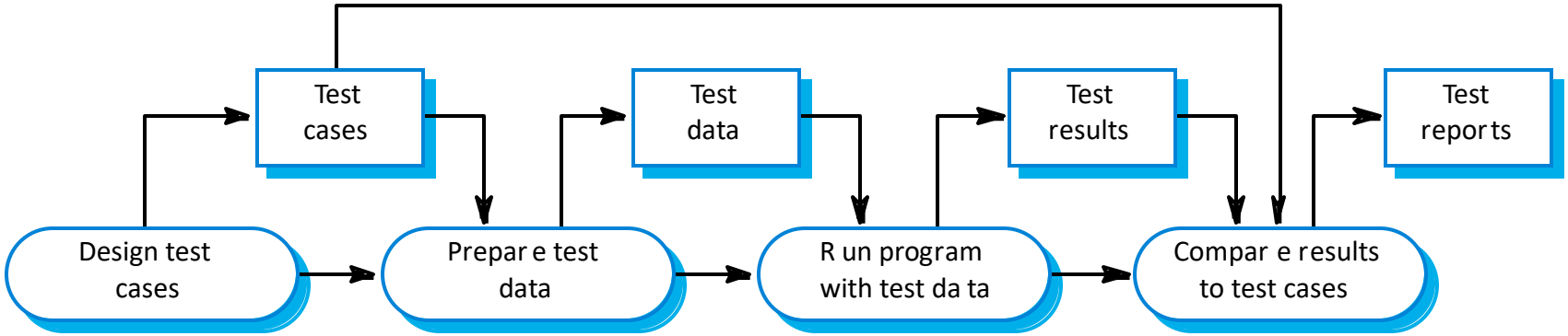
Testing phases



Testing goals

- Validation testing
  - Demonstrate to the developer and the system customer that the software meets its requirements
  - A successful test shows that the system operates as intended
- Defect testing
  - Discover faults or defects in the software where its behavior is incorrect or not in conformance with its specification
  - A successful test is a test that makes the system perform incorrectly
  - Tests show the presence not the absence of defects

The software testing process



Testing policies

- Only exhaustive testing can show a program is free from defects
  - But, exhaustive testing is impossible

- Testing policies define the approach to be used in selecting system tests:
  - All functions accessed through menus should be tested
  - Combinations of functions accessed through the same menu should be tested
  - Where user input is required, all functions must be tested with correct and incorrect input

System testing

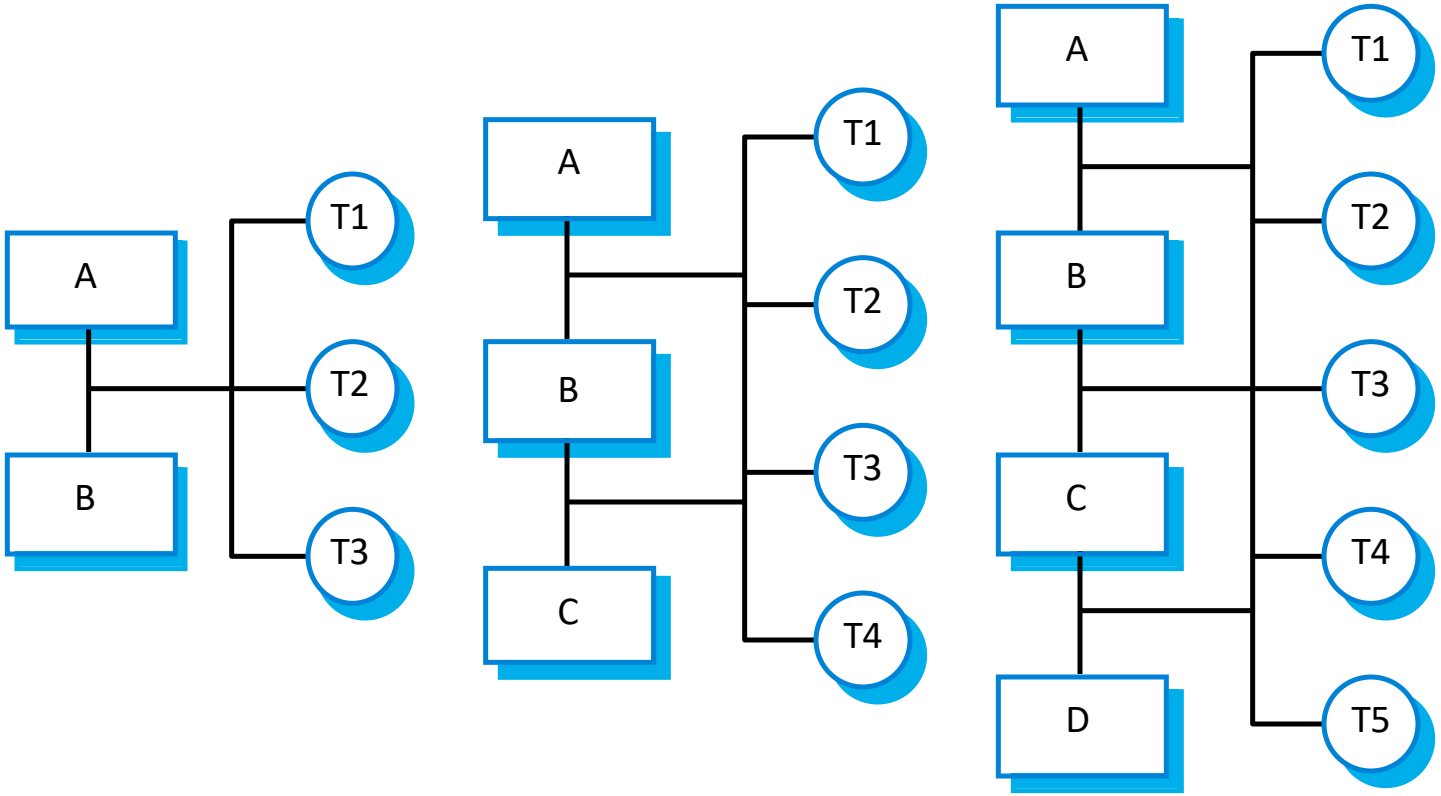
- Involves integrating components to create a system or sub-system
- May involve testing an increment to be delivered to the customer

- Two phases
  - Integration testing
    - System is tested as components are integrated
    - Testing for inter-connection between components
    - Normally black-box testing
  - Release testing
    - Testing the complete system to be delivered as a black-box

Integration testing

- Involves building a system from its components
  - and testing it for problems that arise from component interactions
- Top-down integration
  - Develop the skeleton of the system and populate it with components
- Bottom-up integration
  - Integrate infrastructure components then add functional components
- Systems should be incrementally integrated to avoid “big bang” integration

Incremental integration testing



Test sequence 1 Test sequence 2 Test sequence 3

Testing approaches

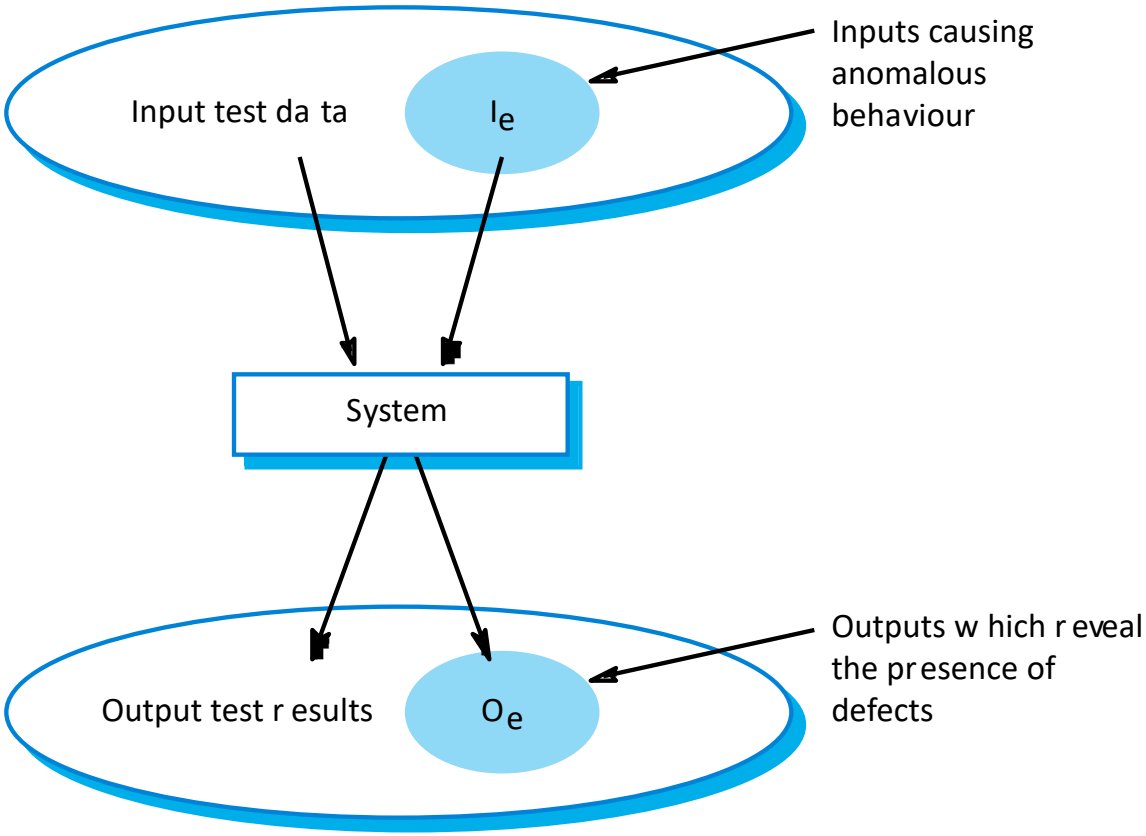
- Architectural validation
  - Top-down integration testing is better at discovering errors in the system architecture
- System demonstration
  - Top-down integration testing allows a limited demonstration at an early stage in the development
- Test implementation
  - Often easier with bottom-up integration testing
- Test observation
  - Problems with both approaches
  - Extra code may be required to observe tests

Release testing

- Process of testing a release of a system before distributing to customers
- Primary goal: increase the supplier's confidence that the system meets its requirements

- Release testing is usually black-box testing
  - Based on the system specifications, requirements, and testers' experience

Black-box testing

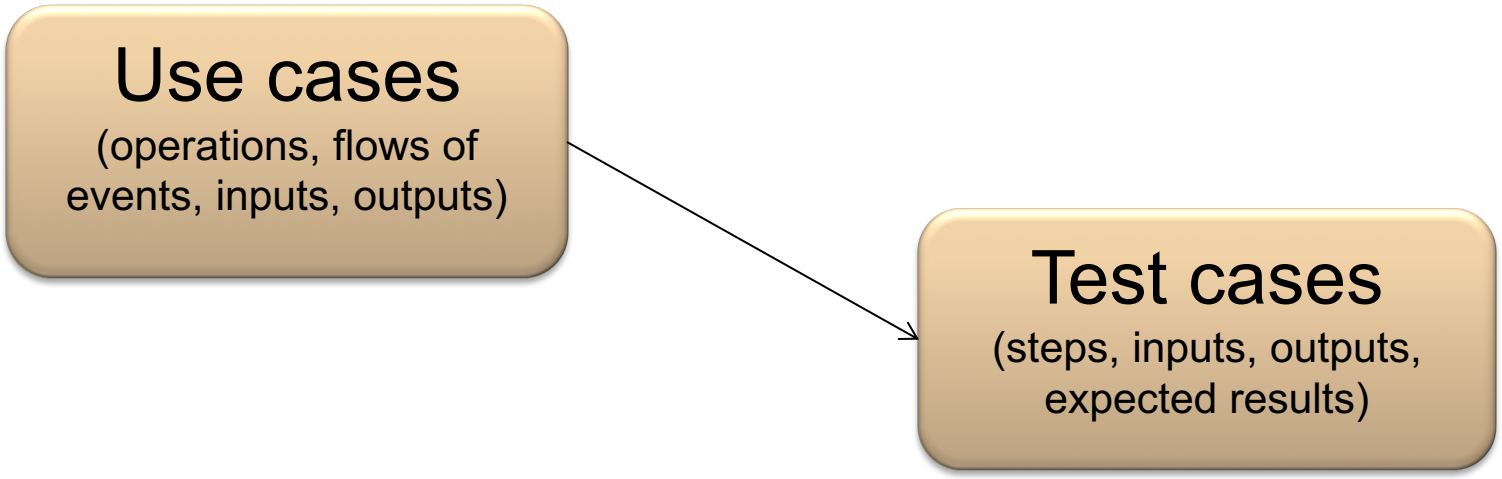


Some testing guidelines

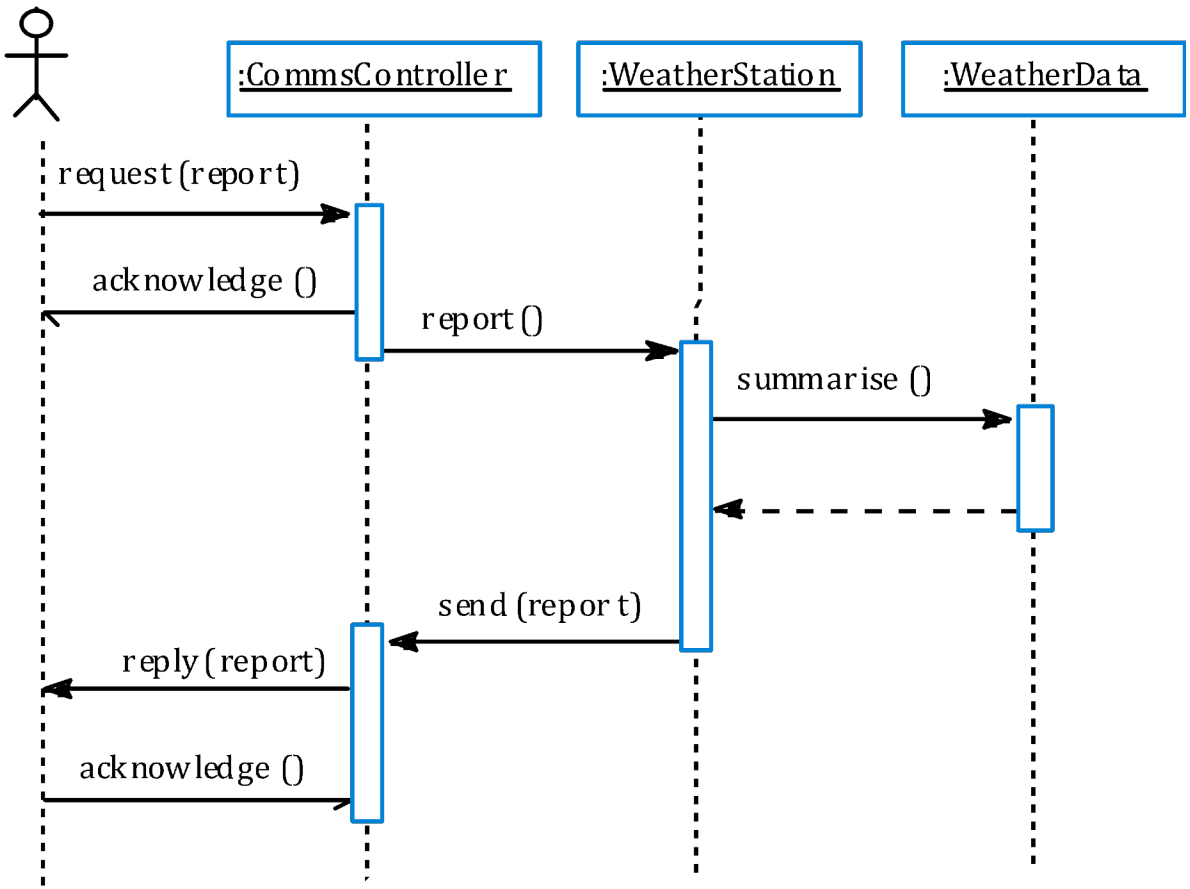
- Some guidelines for effective testing
  - Choose inputs that force the system to generate all error messages
  - Design inputs that cause buffers to overflow
  - Repeat the same input or input series several times
  - Force invalid outputs to be generated
  - Force computation results to be too large or too small

From use cases to test cases

- Test cases are designed as a basis for testing system
- Test cases can be created from use cases
- Using use cases to identify flows of events, operations, inputs and outputs for test cases



Collect weather data sequence chart



Topics covered

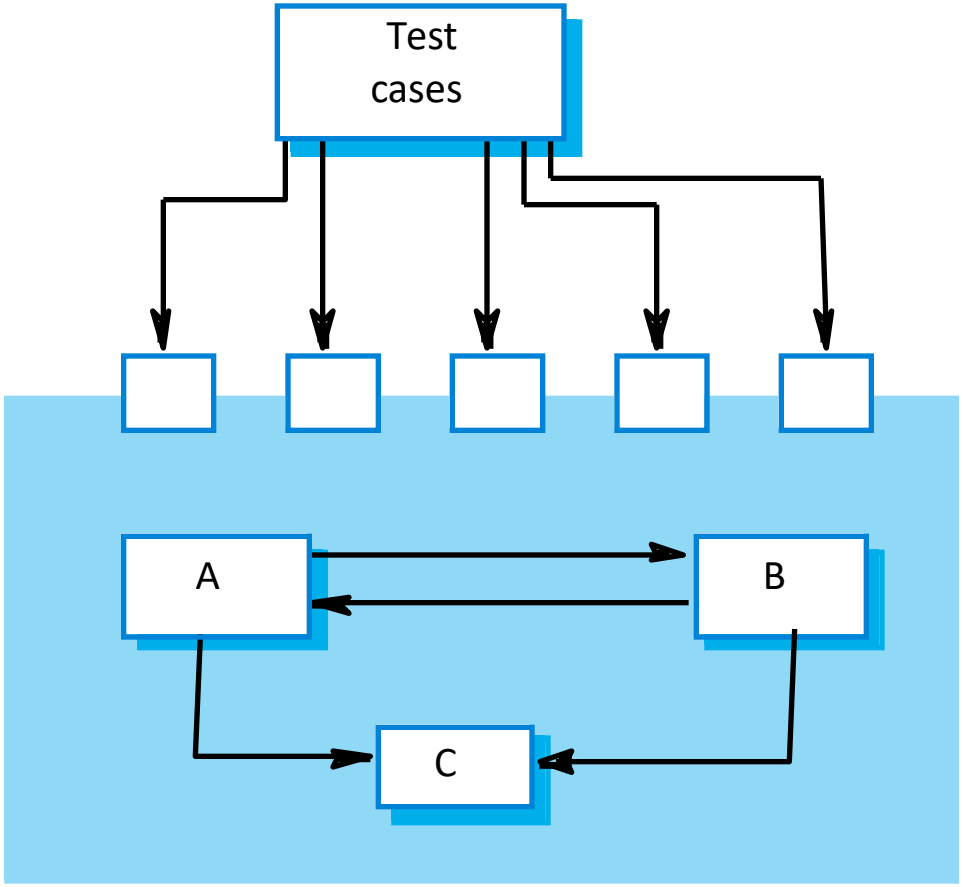
- System testing
- Component testing
- Test case design
- Test automation

Component testing

- Component or unit testing is the process of testing individual components in isolation

- Components may be:
  - Individual functions or methods within an object
  - Object classes with several attributes and methods
  - Composite components with defined interfaces used to access their functionality

Interface testing



Some interface types

- Parameter interfaces
  - Data passed from one procedure to another
- Shared memory interfaces
  - Block of memory is shared between procedures or functions.
- Procedural interfaces
  - Sub-system encapsulates a set of procedures to be called by other sub-systems
- Message passing interfaces
  - Sub-systems request services from other sub-systems

Topics covered

- System testing
- Component testing
- Test case design
- Test automation

Test case design

- Involves designing the test cases used to test the system
- Goal: create a set of tests that are effective in V&V

- Design approaches
  - Requirements-based testing
  - Partition testing
  - Structural testing

Requirements based testing

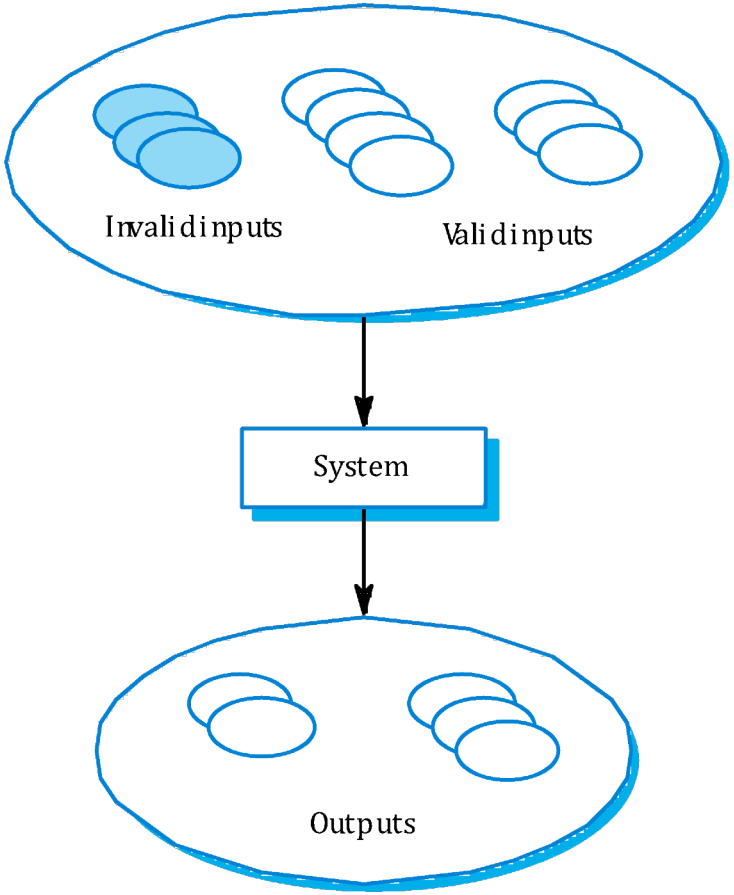
- A general principle of requirements engineering is that requirements should be testable

- Requirements-based testing is the most common testing technique
  - consider each requirement and derive a set of tests for that requirement

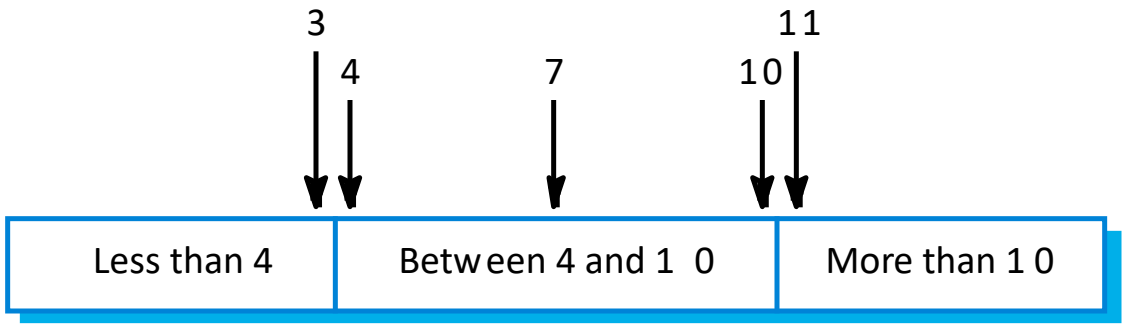
Partition testing

- Input data and output results often fall into different classes
  - Each of these classes is an equivalence partition where the program behaves in an equivalent way for each class member
  - Test cases should be chosen from each partition

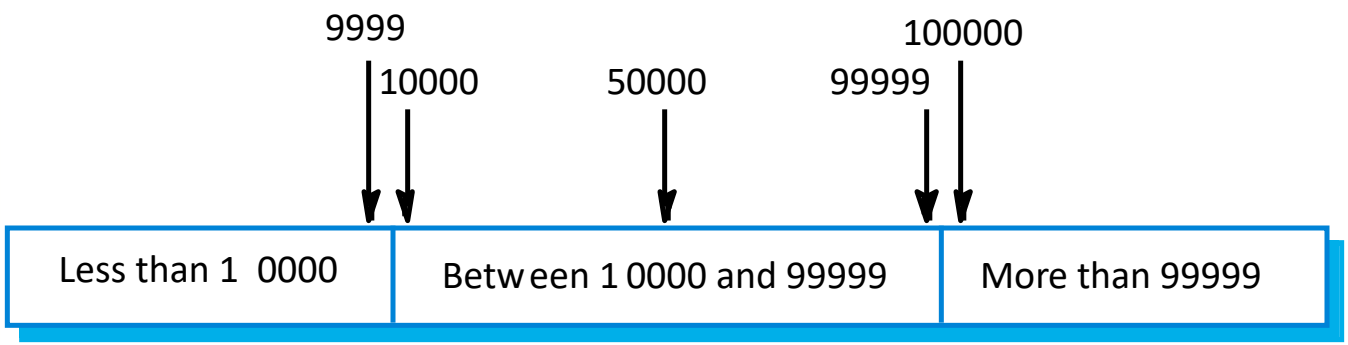
Equivalence partitioning



Equivalence partitions



Number of input values



Input values

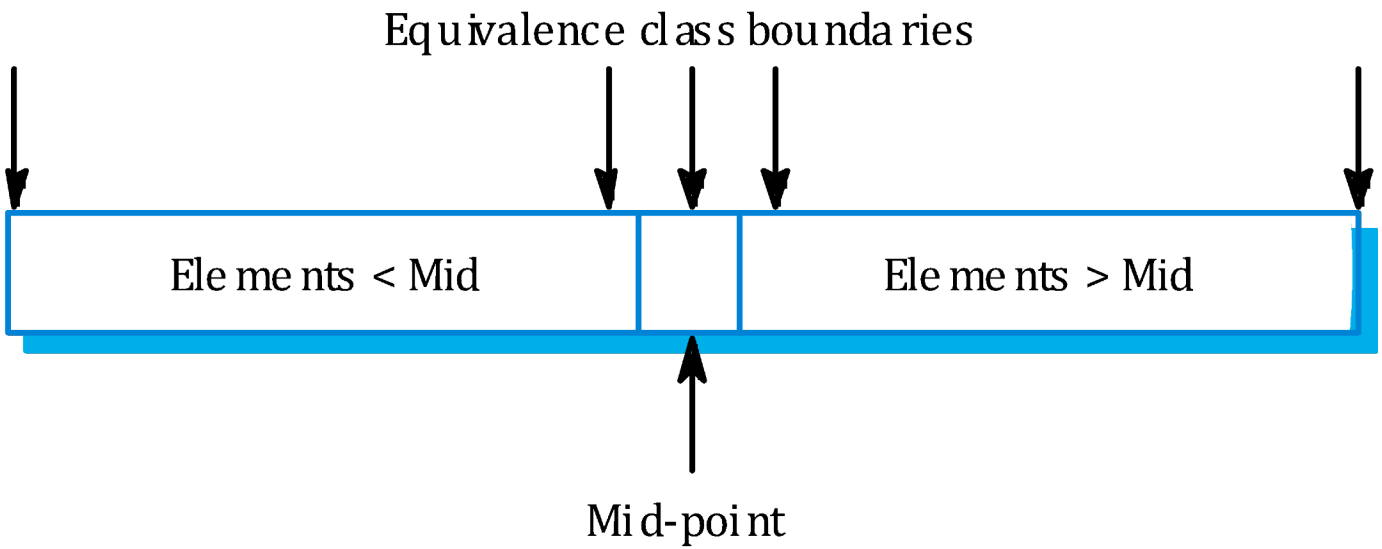
Testing guidelines (sequences)

- Test software with sequences which have only a single value
- Use sequences of different sizes in different tests
- Derive tests so that the first, middle and last elements of the sequence are accessed
- Test with sequences of zero length

Binary search - equiv. partitions

- Pre-conditions satisfied, key element in array
- Pre-conditions satisfied, key element not in array
- Pre-conditions unsatisfied, key element in array
- Pre-conditions unsatisfied, key element not in array
- Input array has a single value
- Input array has an even number of values
- Input array has an odd number of values

Binary search equiv. partitions



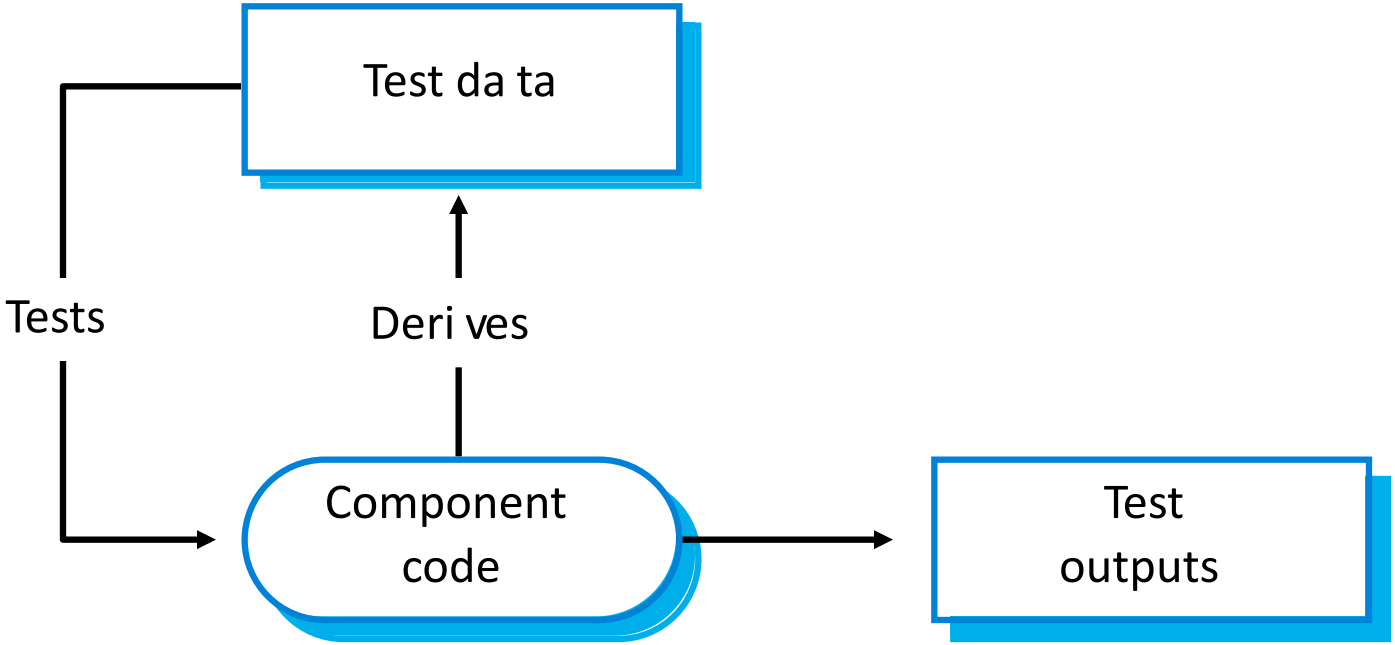
Binary search - test cases

Input array (T)	**Key** (Key)	**Output** (Found, L)
17	17	true, 1
17	0	false, ??
17, 21, 23, 29	17	true, 1
9, 16, 18, 30, 31, 41, 45	45	true, 7
17, 18, 21, 23, 29, 38, 41	23	true, 4
17, 18, 21, 23, 29, 33, 38	21	true, 3
12, 18, 21, 23, 32	23	true, 4
21, 23, 29, 33, 38	25	false, ??

Structural testing

- Sometime called white-box testing
- Derivation of test cases according to program structure
  - Knowledge of the program is used to identify additional test cases
- Objective is to exercise all program statements (not all path combinations)

Structural testing

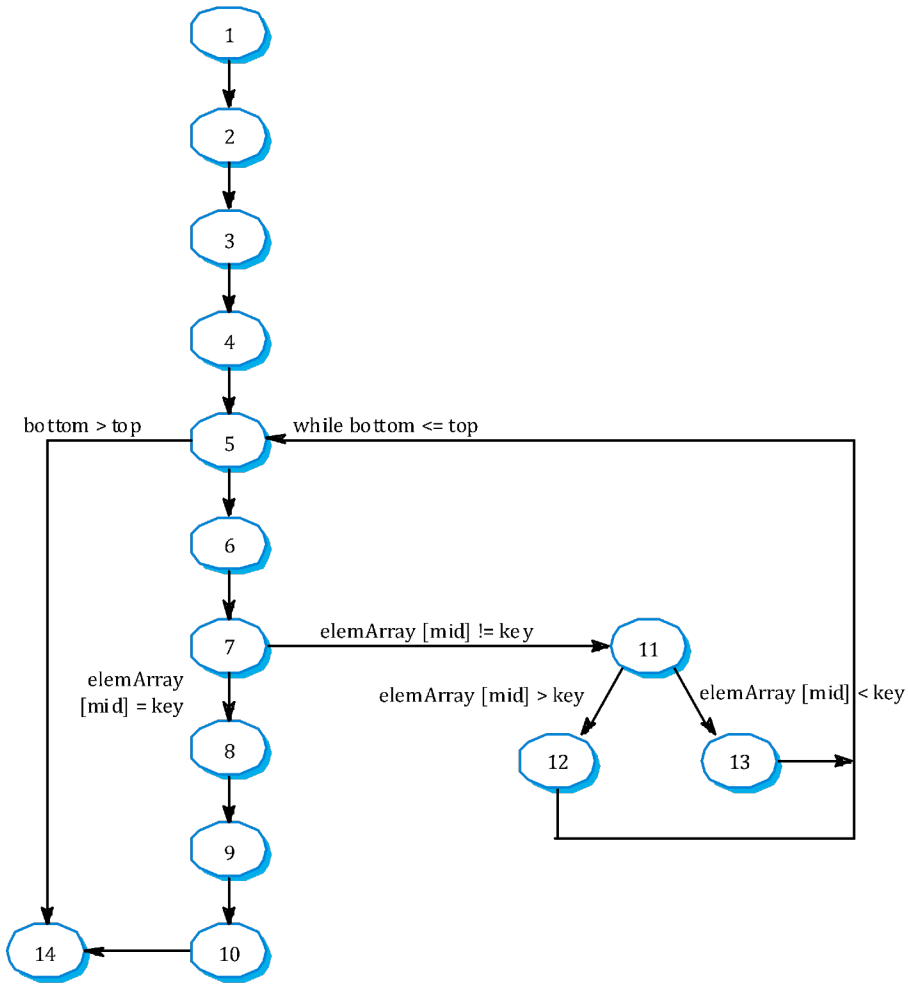


Path testing

- Objective is to ensure that **each path** through the program is executed at least once
- It includes
  - starting point
  - nodes representing program decisions
  - arcs representing the flow of control
- Statements with conditions are nodes in the flow graph

```
f (int i, int j) {
if (i == 10) {
        //some code here
} else {
        if (j == 5) {
        // some code here
        } else {
        // some code here
        }
}
//some code here
}
```

Binary search flow graph



How many test cases should be tested to cover all possible paths?

Topics covered

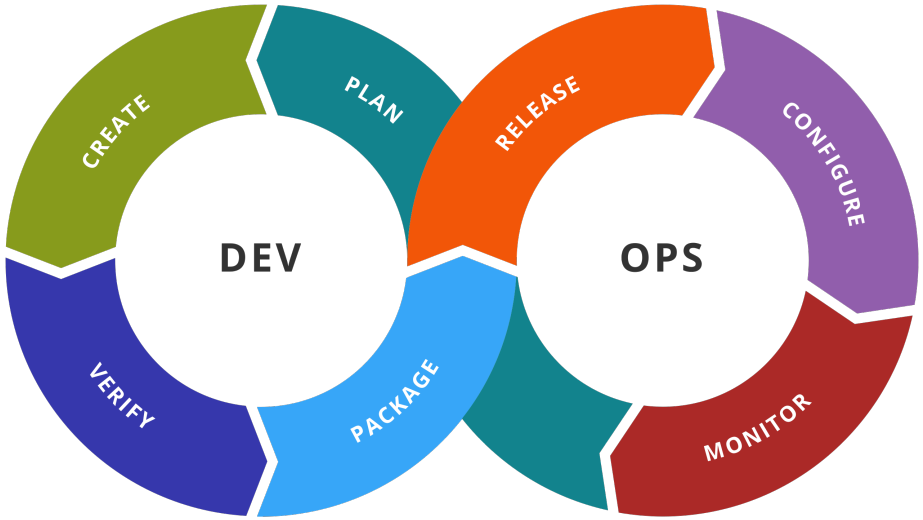
- System testing
- Component testing
- Test case design
- Test automation

Test automation

- Much testing is now done manually but it's expensive
- Test automation: testing is done automatically with support of tools
- Benefits
  - Reduce time required and total testing costs
  - Some tests cannot be done manually
- Many tools and frameworks
  - jUnit
  - Cucumber
  - Selenium
  - QuickTest Pro
  - Etc.

Test automation becomes essential

- Test automation is a must in DevOps practices
- It allows delivering quality software fast
- GUI and API/service testing



Key points

- Testing can show the presence of faults in a system
  - it cannot prove there are no remaining faults
- Component developers are responsible for component testing
- System testing is the responsibility of a separate team
- Integration testing is testing increments of the system
- Release testing involves testing a system to be released to a customer
- Use experience and guidelines to design test cases in defect testing

Key points

- Interface testing is designed to discover defects in the interfaces of composite components
- Equivalence partitioning is a way of discovering test cases
  - all cases in a partition should behave in the same way
- Structural analysis relies on analyzing a program and deriving tests from this analysis
- Test automation reduces testing costs by supporting the test process with a range of software tools