

# Verification and Validation

---

- ◆ Assuring that a software system meets a user's needs

# Objectives

---

- ◆ To introduce software verification and validation
- ◆ To describe the stages of the testing process
- ◆ To explain the importance of test planning
- ◆ To describe various complementary testing strategies

# Topics covered

---

- ◆ The testing process
- ◆ Test planning
- ◆ Testing strategies

# Verification vs validation

---

- ◆ Verification:
  - "Are we building the product right"
- ◆ The software should conform to its specification
- ◆ Validation:
  - "Are we building the right product"
- ◆ The software should do what the user really requires

# The V & V process

---

- ◆ Is a whole life-cycle process - V & V must be applied at each stage in the software process.
- ◆ Has two principal objectives
  - The discovery of defects in a system
  - The assessment of whether or not the system is usable in an operational situation.

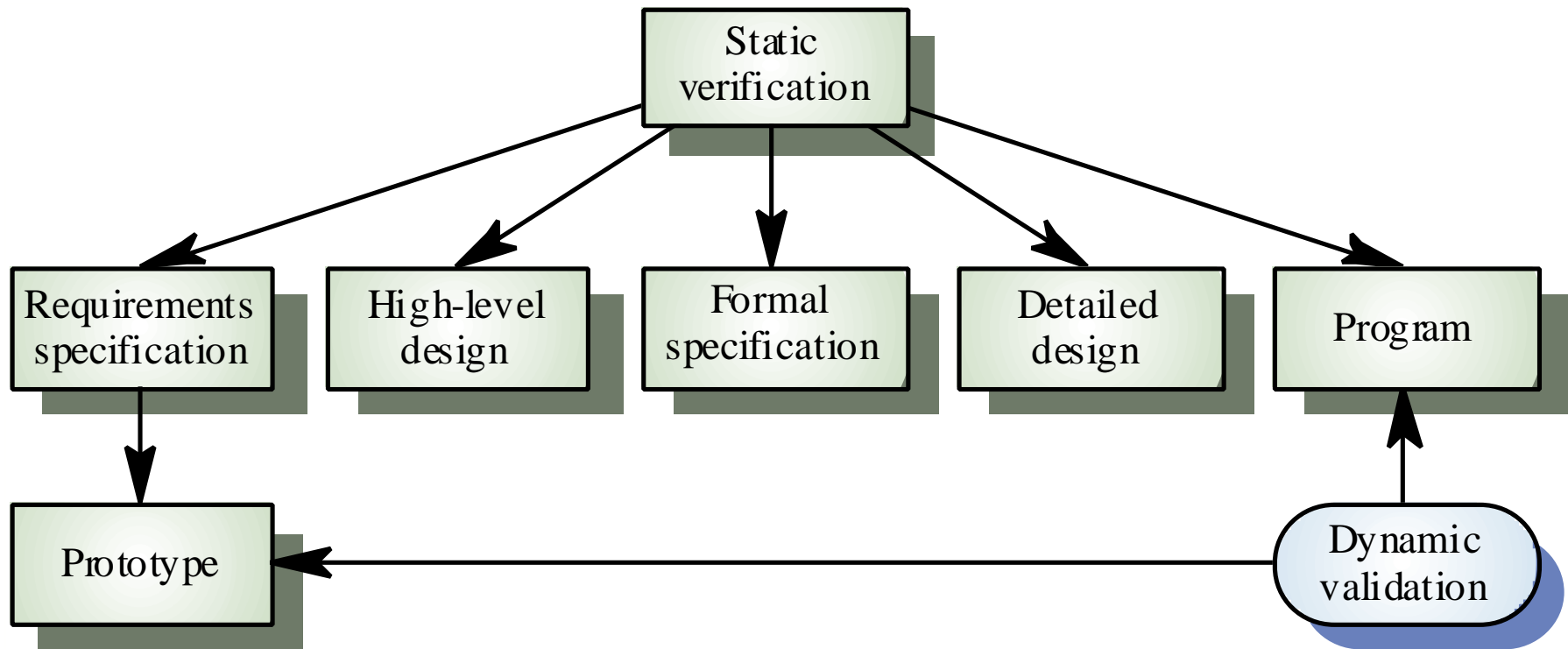
# Dynamic and static verification

---

- ◆ *Dynamic V & V* Concerned with exercising and observing product behaviour (testing)
- ◆ *Static verification* Concerned with analysis of the static system representation to discover problems

# Static and dynamic V&V

---



# Program testing

---

- ◆ Can reveal the presence of errors NOT their absence
- ◆ A successful test is a test which discovers one or more errors
- ◆ Only validation technique for non-functional requirements
- ◆ Should be used in conjunction with static verification

# Types of testing

---

## ◆ Statistical testing

- tests designed to reflect the frequency of user inputs. Used for reliability estimation.
- Covered in Chapter 18 - Software reliability.

## ◆ Defect testing

- Tests designed to discover system defects.
- A successful defect test is one which reveals the presence of defects in a system.
- Covered in Chapter 23 - Defect testing

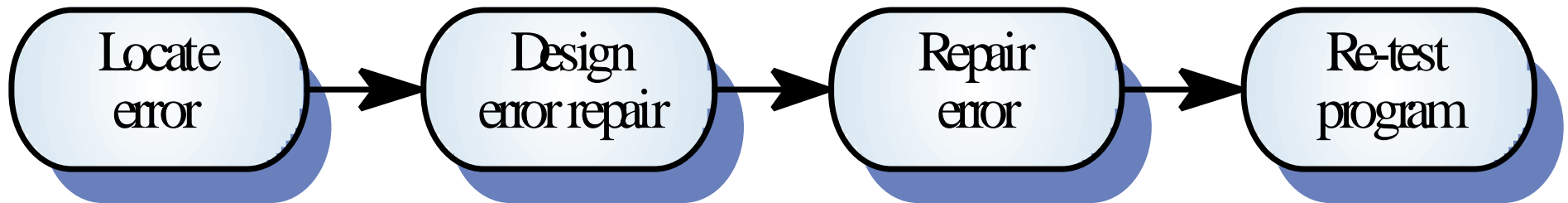
# Testing and debugging

---

- ◆ Defect testing and debugging are distinct processes
- ◆ Defect testing is concerned with confirming the presence of errors
- ◆ Debugging is concerned with locating and repairing these errors
- ◆ Debugging involves formulating a hypothesis about program behaviour then testing these hypotheses to find the system error

# The debugging process

---



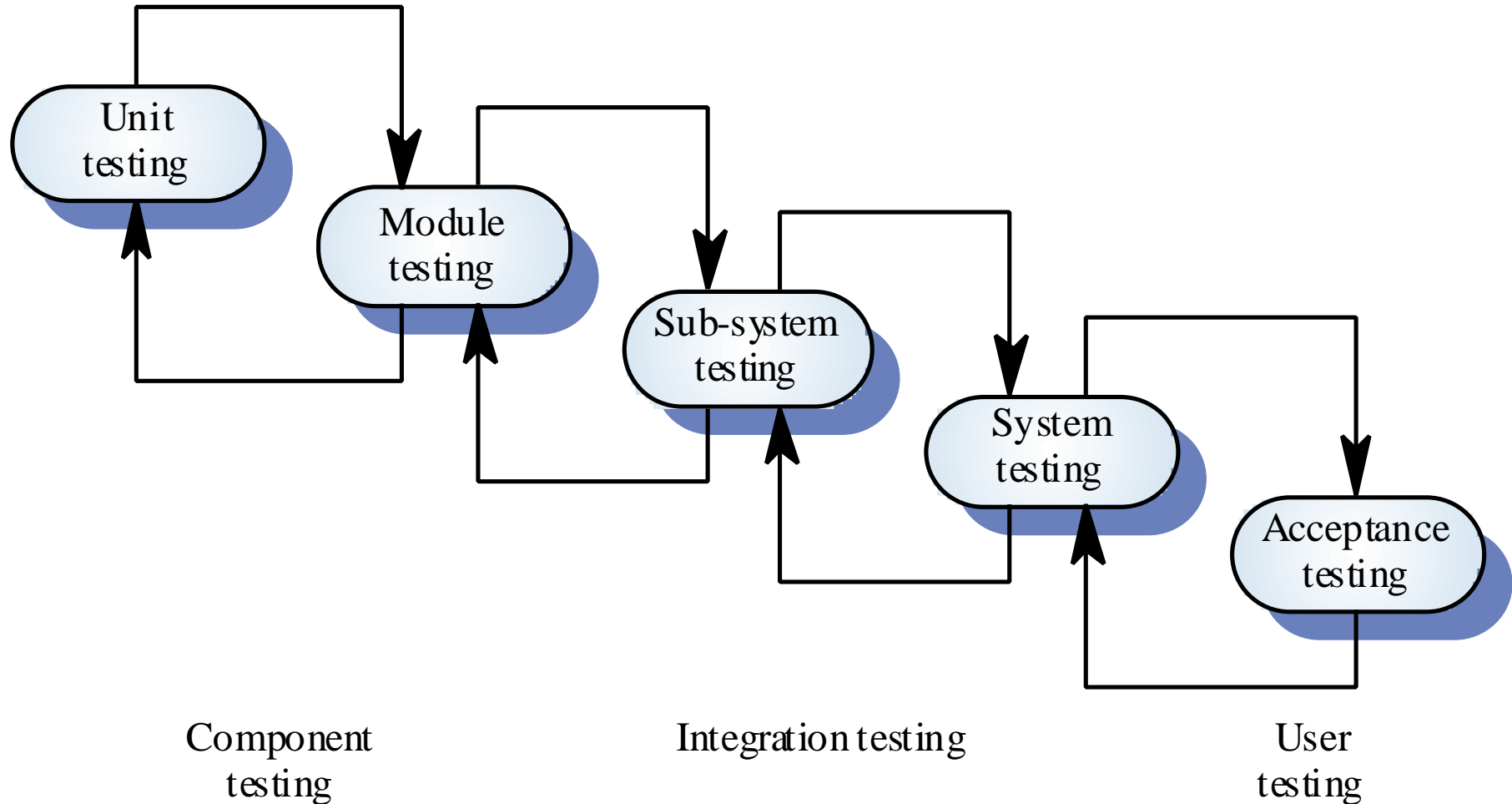
# Testing stages

---

- ◆ Unit testing
  - testing of individual components
- ◆ Module testing
  - testing of collections of dependent components
- ◆ Sub-system testing
  - testing collections of modules integrated into sub-systems
- ◆ System testing
  - testing the complete system prior to delivery
- ◆ Acceptance testing
  - testing by users to check that the system satisfies requirements. Sometimes called alpha testing

# The testing process

---



# Object-oriented system testing

---

- ◆ Less closely coupled systems. Objects are not necessarily integrated into sub-systems
- ◆ Cluster testing. Test a group of cooperating objects
- ◆ Thread testing. Test a processing thread as it weaves from object to object. Discussed later in real-time system testing

# Test planning and scheduling

---

- ◆ Describe major phases of the testing process
- ◆ Describe tracability of tests to requirements
- ◆ Estimate overall schedule and resource allocation
- ◆ Describe relationship with other project plans
- ◆ Describe recording method for test results

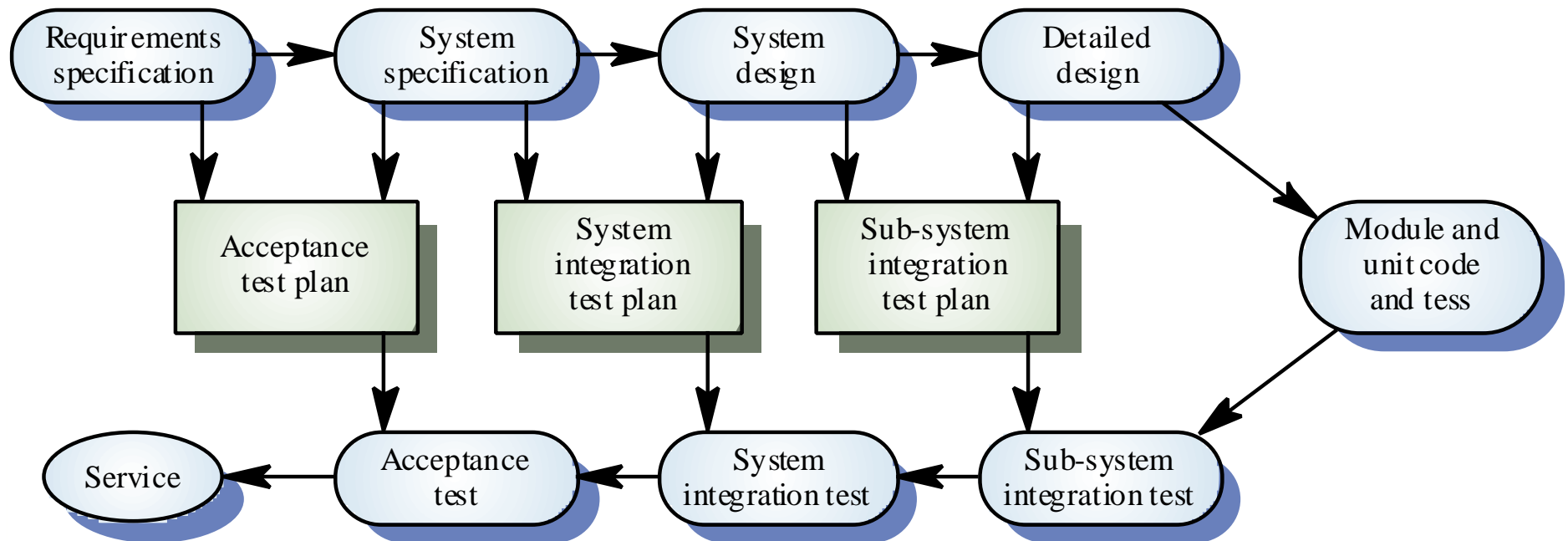
# The test plan

---

- ◆ The testing process
- ◆ Requirements traceability
- ◆ Tested items
- ◆ Testing schedule
- ◆ Test recording procedures
- ◆ Hardware and software requirements
- ◆ Constraints

# The V-model of development

---



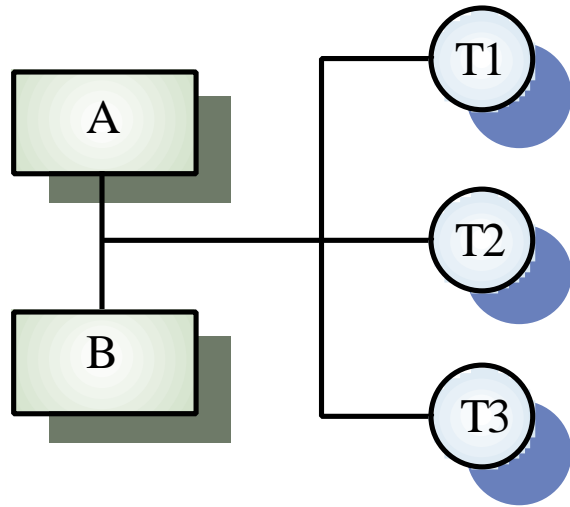
# Testing strategies

---

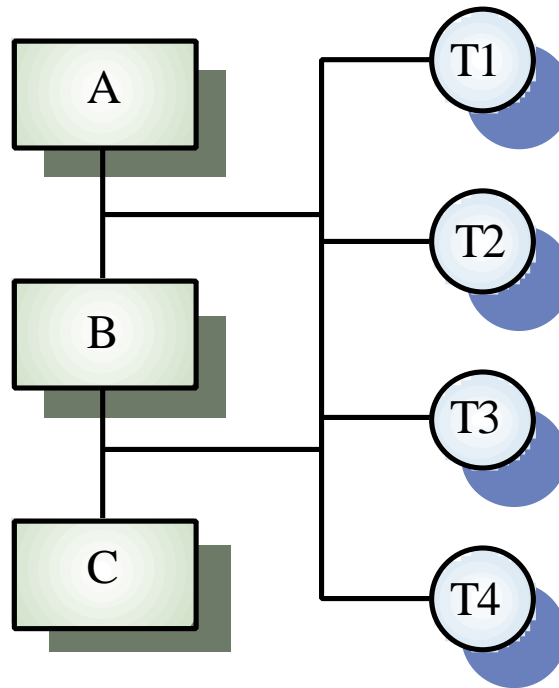
- ◆ Testing strategies are ways of approaching the testing process
- ◆ Different strategies may be applied at different stages of the testing process
- ◆ Strategies covered
  - Top-down testing
  - Bottom-up testing
  - Thread testing
  - Stress testing
  - Back-to-back testing

# Incremental testing

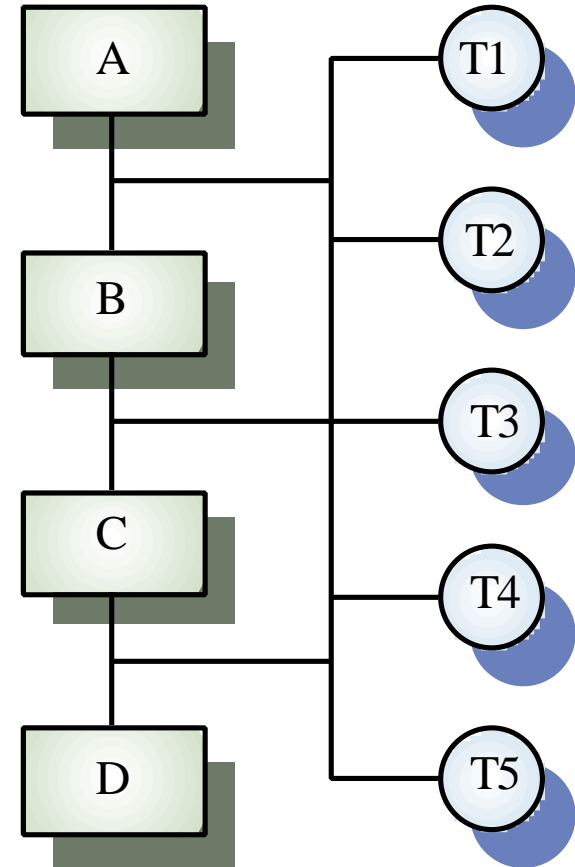
---



Test sequence  
1



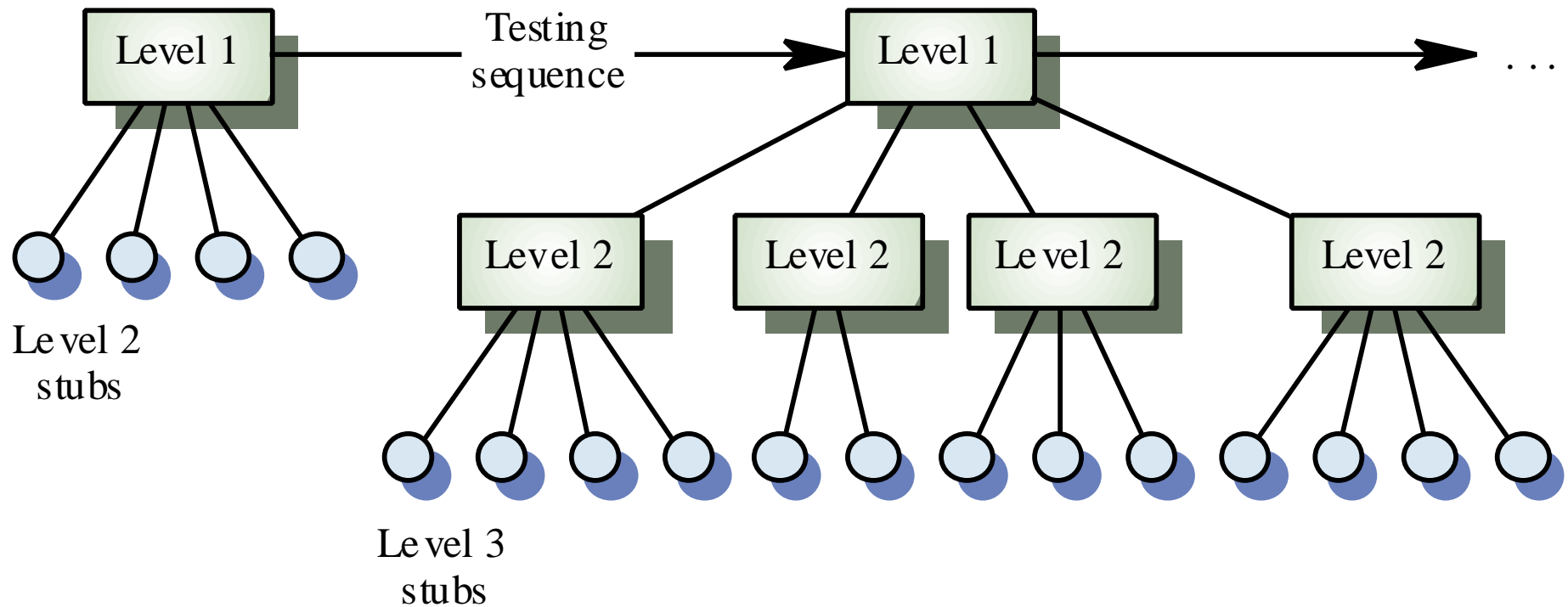
Test sequence  
2



Test sequence  
3

# Top-down testing

---



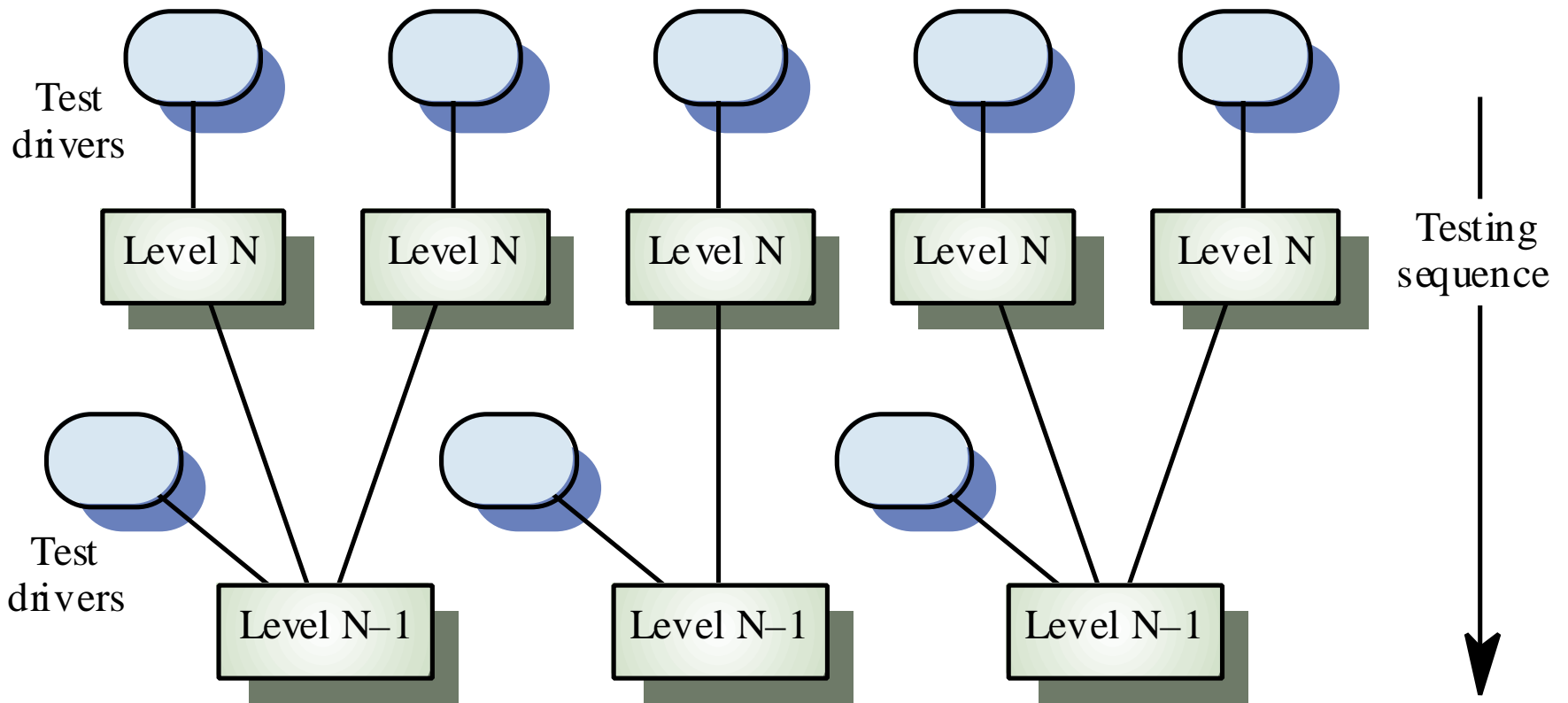
# Top-down testing

---

- ◆ Start with the high-levels of a system and work your way downwards
- ◆ Testing strategy which is used in conjunction with top-down development
- ◆ Finds architectural errors
- ◆ May need system infrastructure before any testing is possible
- ◆ May be difficult to develop program stubs

# Bottom-up testing

---



# Bottom-up testing

---

- ◆ Necessary for critical infrastructure components
- ◆ Start with the lower levels of the system and work upward
- ◆ Needs test drivers to be implemented
- ◆ Does not find major design problems until late in the process
- ◆ Appropriate for object-oriented systems

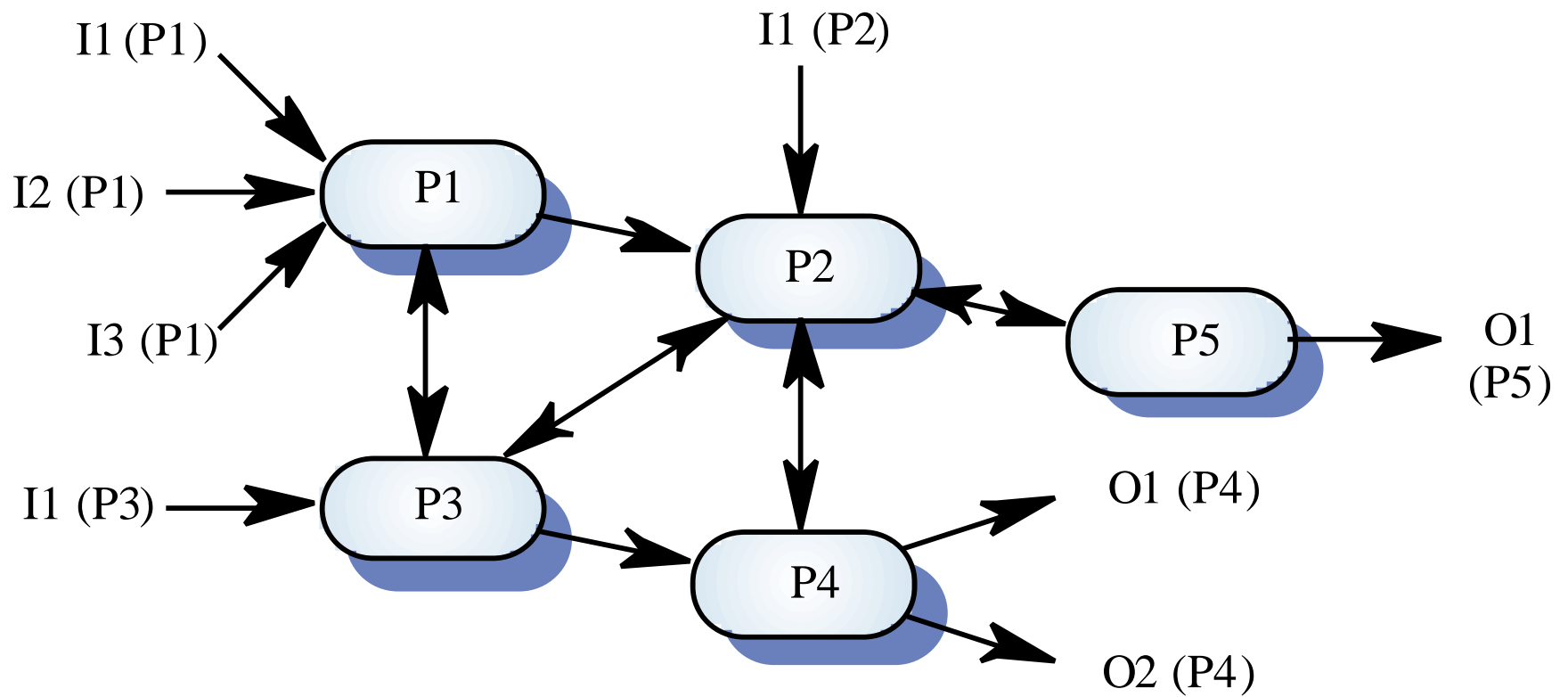
# Thread testing

---

- ◆ Suitable for real-time and object-oriented systems
- ◆ Based on testing an operation which involves a sequence of processing steps which thread their way through the system
- ◆ Start with single event threads then go on to multiple event threads
- ◆ Complete thread testing is impossible because of the large number of event combinations

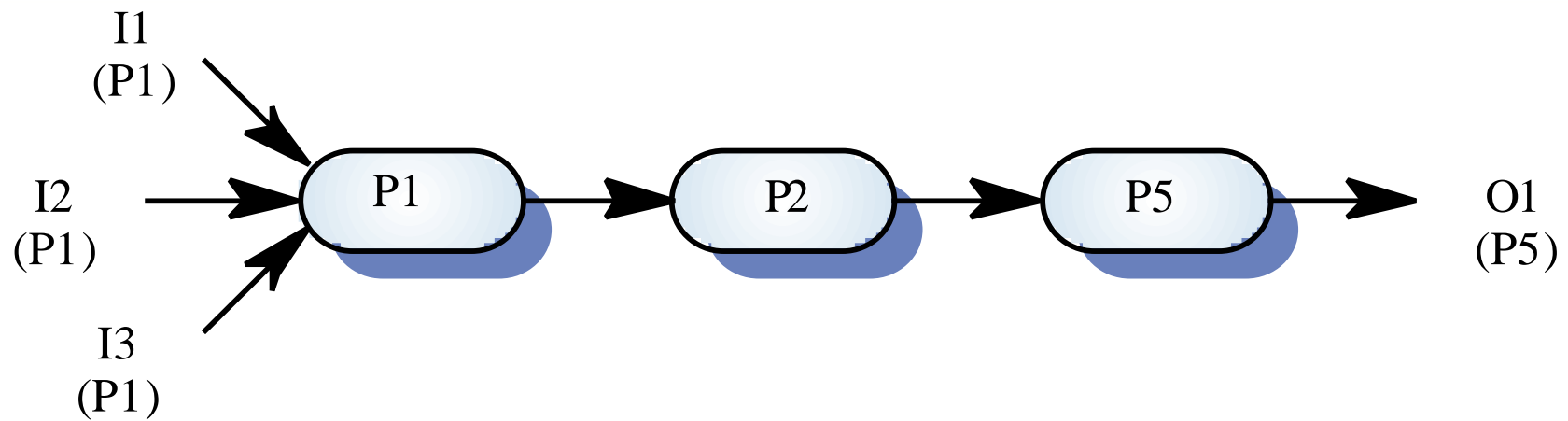
# Process interactions

---



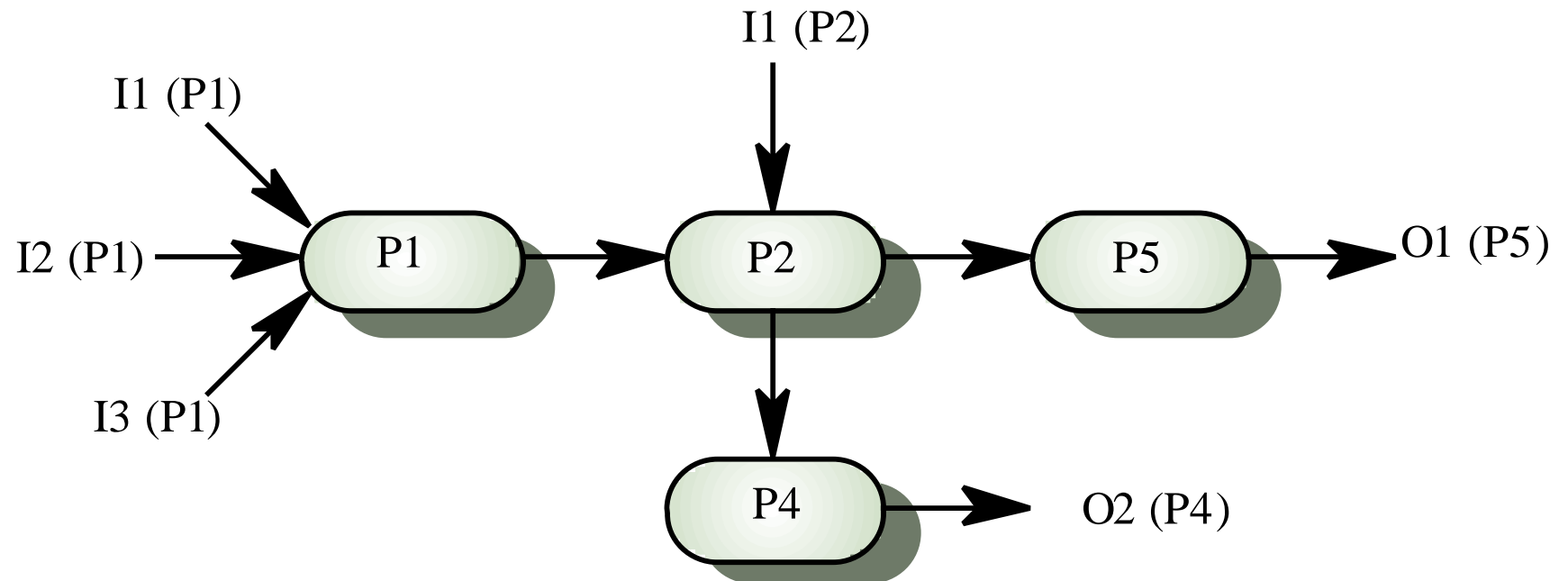
# Thread testing

---



# Multiple-thread testing

---



# Stress testing

---

- ◆ Exercises the system beyond its maximum design load. Stressing the system often causes defects to come to light
- ◆ Stressing the system test failure behaviour.. Systems should not fail catastrophically. Stress testing checks for unacceptable loss of service or data
- ◆ Particularly relevant to distributed systems which can exhibit severe degradation as a network becomes overloaded

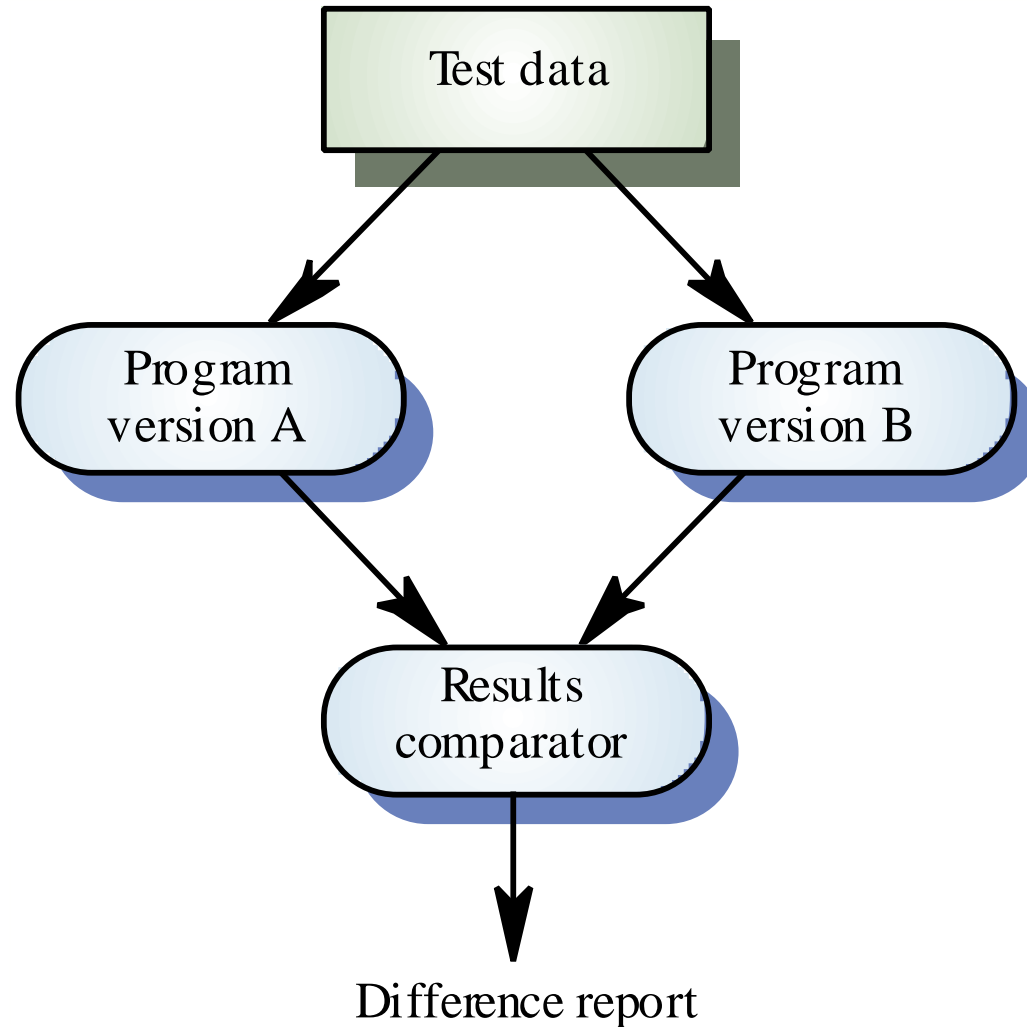
# Back-to-back testing

---

- ◆ Present the same tests to different versions of the system and compare outputs. Differing outputs imply potential problems
- ◆ Reduces the costs of examining test results. Automatic comparison of outputs.
- ◆ Possible when a prototype is available or with regression testing of a new system version

# Back-to-back testing

---



# Key points

---

- ◆ Verification and validation are not the same thing
- ◆ Testing is used to establish the presence of defects and to show fitness for purpose
- ◆ Testing activities include unit testing, module testing, sub-system testing, integration testing and acceptance testing
- ◆ Object classes should be tested in O-O systems

# Key points

---

- ◆ Testing should be scheduled as part of the planning process. Adequate resources must be made available
- ◆ Test plans should be drawn up to guide the testing process
- ◆ Testing strategies include top-down testing, bottom-up testing, stress testing, thread testing and back-to-back testing