Software College Northeastern University

Software Quality Assurance and Testing

Chapter 7 Integration Testing

wuchenni@qq.com

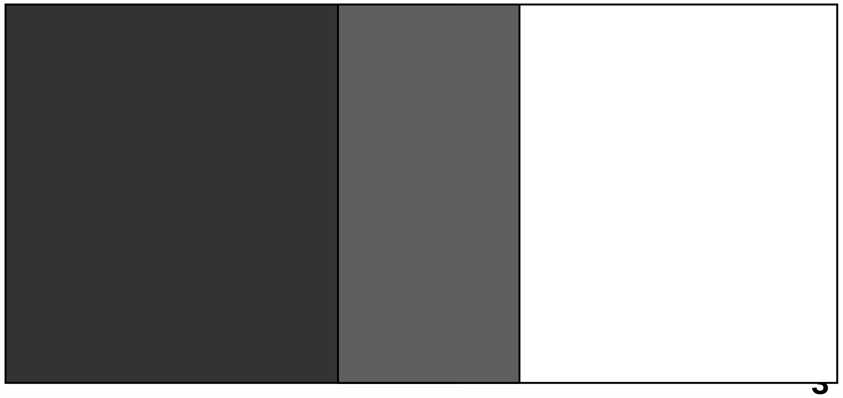
Contents

Chapter 7

Integration Testing

7.1 Integration Testing Introduction

- Test
  - Unit testing → Integration testing → System testing
- Software development
  - Requirement → High-level design → low-level design → Code
- Gray Box Testing



7.1 Integration Testing Introduction

- **Integration testing** *（集成测试、组装测试、联合测试、子系统测试、部件测试）*
  - Integration testing is the phase of software testing in which individual software modules are combined and tested as a group.
  - Integration testing follows unit testing and precedes system testing.

7.1 Integration Testing Introduction

- Integration testing **VS** Unit testing
  - Before Integration testing, Unit testing had been completed.
  - Unit testing and Integration testing focus on the different range.
- Integration testing **VS** System testing
  - System testing and integration testing stand at different angles.

7.1 Integration Testing Introduction

- The purpose of integration testing is to verify functional, performance and reliability requirements placed on high-level design items.

7.1 Integration Testing Introduction

- **The major advantages of integration testing:**
  - Defects are detected early.
  - It is easier to fix defects detected earlier.
  - We get earlier feedback on the health and acceptability of the individual modules and on the overall system.
  - Scheduling of defect fixes is flexible, and it can overlap with development.

7.1 Integration Testing Introduction

- **Major testing focuses:**
  - Interfaces between modules (or components)
  - Integrated functional features
  - Interacting protocols and messages
  - System architectures

7.1 Integration Testing Introduction

- **Integration Testing Steps:**
  - Step 1: Create a test plan
  - Step 2: Design test cases and prepare test data
  - Step 3: If applicable create scripts to run test cases
  - Step 4: Once the components have been integrated execute the test cases
  - Step 5: Fix the bugs if any and retest the codes
  - Step 6: Repeat the test cycle until the components have been successfully integrated

7.1 Integration Testing Introduction

- The number of integration test cycles and the total integration time are determined by the following parameters:
  - Number of modules in the system
  - Relative complexity of the module (cyclomatic complexity)
  - Relative complexity of the interfaces between the modules
  - Number of modules needed to be clustered together in each test cycle
  - Whether the modules to be integrated have been adequately tested before
  - Turnaround time for each test-debug-fix cycle

7.1 Integration Testing Introduction

- Integration testing is conducted in an incremental manner as a series of test cycles.
- In each test cycle, a few more modules are integrated with an existing and tested build to generated larger builds.
- The complete system is built, cycle by cycle until the whole system is operational for system-level testing.

7.1 Integration Testing Introduction

- When to complete the integration testing?
  - The system is fully integrated together.
  - All the test cases have been executed.
  - All the severe and moderated defects found have been fixed.

7.1 Integration Testing Introduction

- Integration testing features
  - Unit testing is not completely, it will powerless for checking whether the content correctness, mutually called relationship of modules.
  - Compared with system testing, integration test cases start from the program structure, it’s purpose and target stronger, testing can discover and locate the problems more efficient.
  - It can simulate the special difficult abnormal flow more easily, and then system can test test cases.
  - Rapid position problem.

7.1 Integration Testing Introduction

- The relationship between integration testing and development
  - Integration testing and outline design corresponds, outline design in the system structure is the basis of integration testing.
  - Booch Grady think integration is an object-oriented developing key activities.
  - In structural design developing, integration test are important equally.
  - The dependence between integration test and architecture design.

7.1 Integration Testing Introduction

- When to carry out the integration testing?
  - – Units or modules to form a component
  - – Components to form a system
  - – Systems to form a product
  - – Integration testing is sometimes defined as the level of testing between unit and system.

7.1 Integration Testing Introduction

- **Integration testing levels**
  - Inter-module integration
  - Intra-subsystem integration
  - Subsystems integration
    - Pairwise testing

7.2 Integration Testing Strategy

- Describe module integration (assembly) approach.
- How to combine the modules of the system? All at the same time assembly or gradually assembled?
- Integration test basic strategy is more, classification is more complex, but can be put into the following two categories:
  - The non-incremental integration strategy -- in one go
  - Incremental integration strategy -- gradually realize

7.2 Integration Testing Strategy

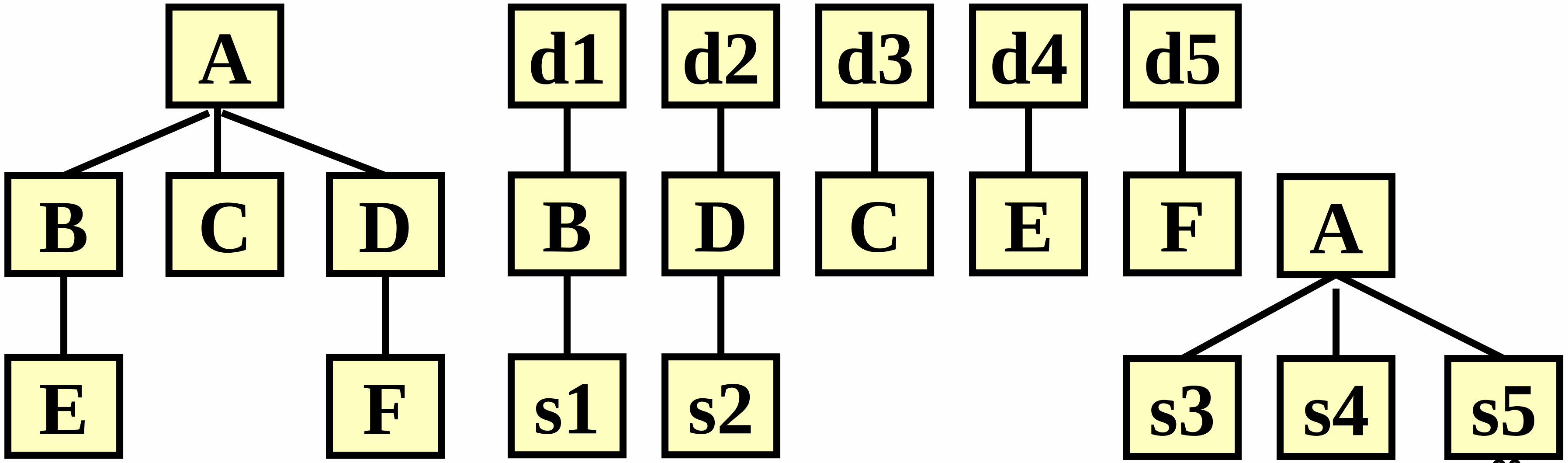
- Integration testing can be divided into two categories:
  - Incremental integration testing: incremental testing expands the set of integrated modules progressively.
  - Non-incremental integration testing: by no incremental testing, software modules are combined and tested randomly.

7.2 Integration Testing Strategy

- Big bang（大爆炸） integration
- Top-down （自顶向下）integration
- Bottom-up （自底向上）integration
- Sandwich（三明治） integration
- Layers（分层） integration
- High-frequency （高频）integration
- Event-based（基于事件） integration

7.2 Integration Testing Strategy

- Big Bang integration
  - It is a non-incremental integration method, it integrates all system components together at one time, not considering the dependence of components and the possible risk.
- Strategy:



7.2 Integration Testing Strategy

- Advantage
  - Integration testing can be completed rapidly and only few stubs and drivers are needed.
  - Several testers can work in the parallel way, and human and material resources utilization is higher.
- Disadvantage
  - Position and change more difficultly when error found.
  - Many interface errors can not be found until system testing.

7.2 Integration Testing Strategy

- Scope
  - Existing system with only minor modifications
  - Small systems with adequate unit testing
  - System made from certified high quality reusable components

7.2 Integration Testing Strategy

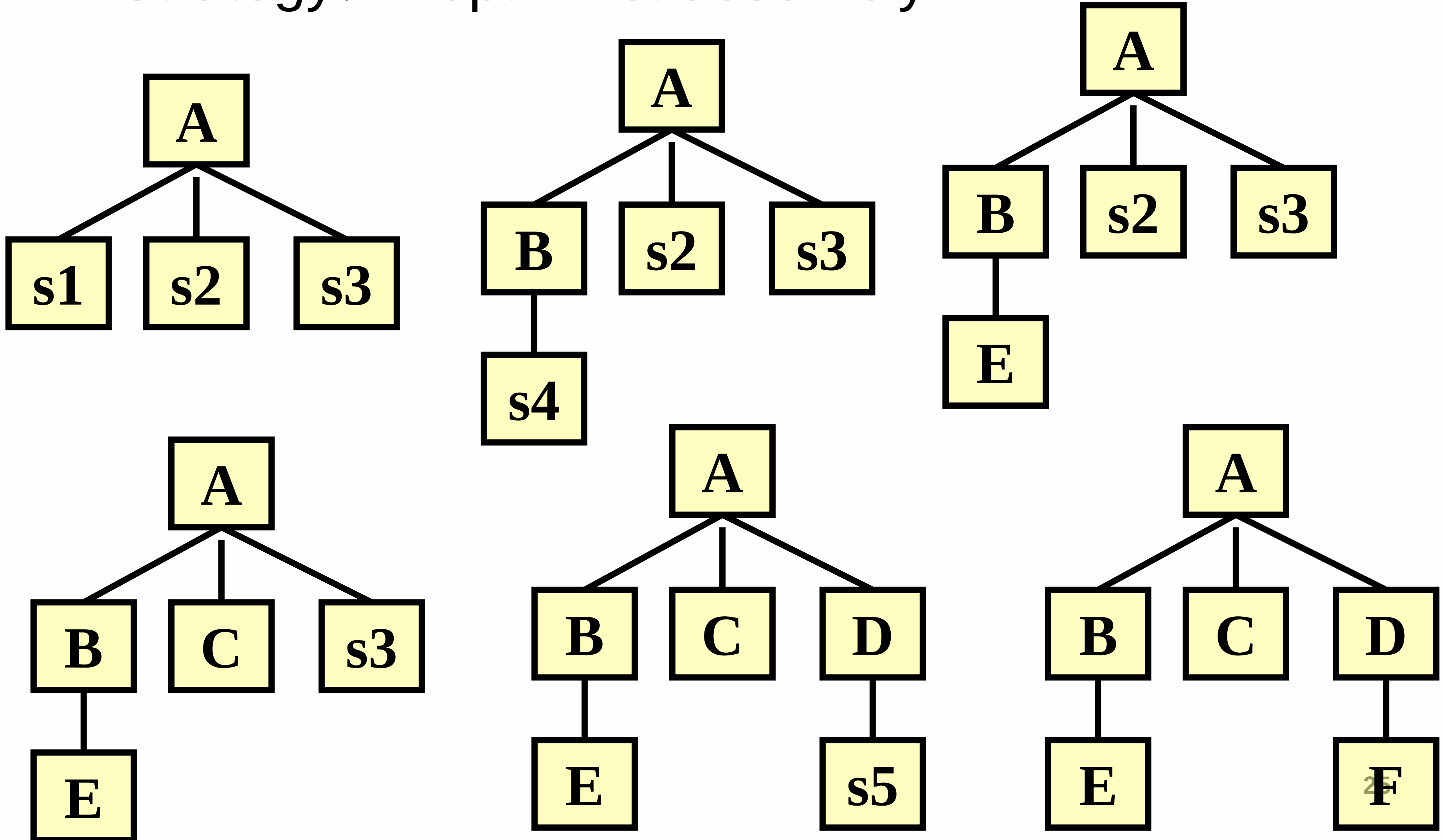
- Instantaneous vs. incremental integration testing
  - Instantaneous integration testing is sometimes referred to as the big bang approach. Locating subtle errors can be very difficult after the bang.
  - Incremental integration testing results in some additional overhead, but can significantly reduce error localization and correction time. The optimum incremental approach is inherently dependent on the individual project and the pros and cons of the various alternatives.

7.2 Integration Testing Strategy

- Top-down integration
  - （1）Focus on the top level components firstly, then gradually test the bottom of components.
  - （2）Depth-first and breath-first strategy can be used.
  - （3）Conduct regression testing, exclude possible errors caused by integrated.
  - （4） All modules are integrated into the system then finish testing, otherwise turn to （2）.

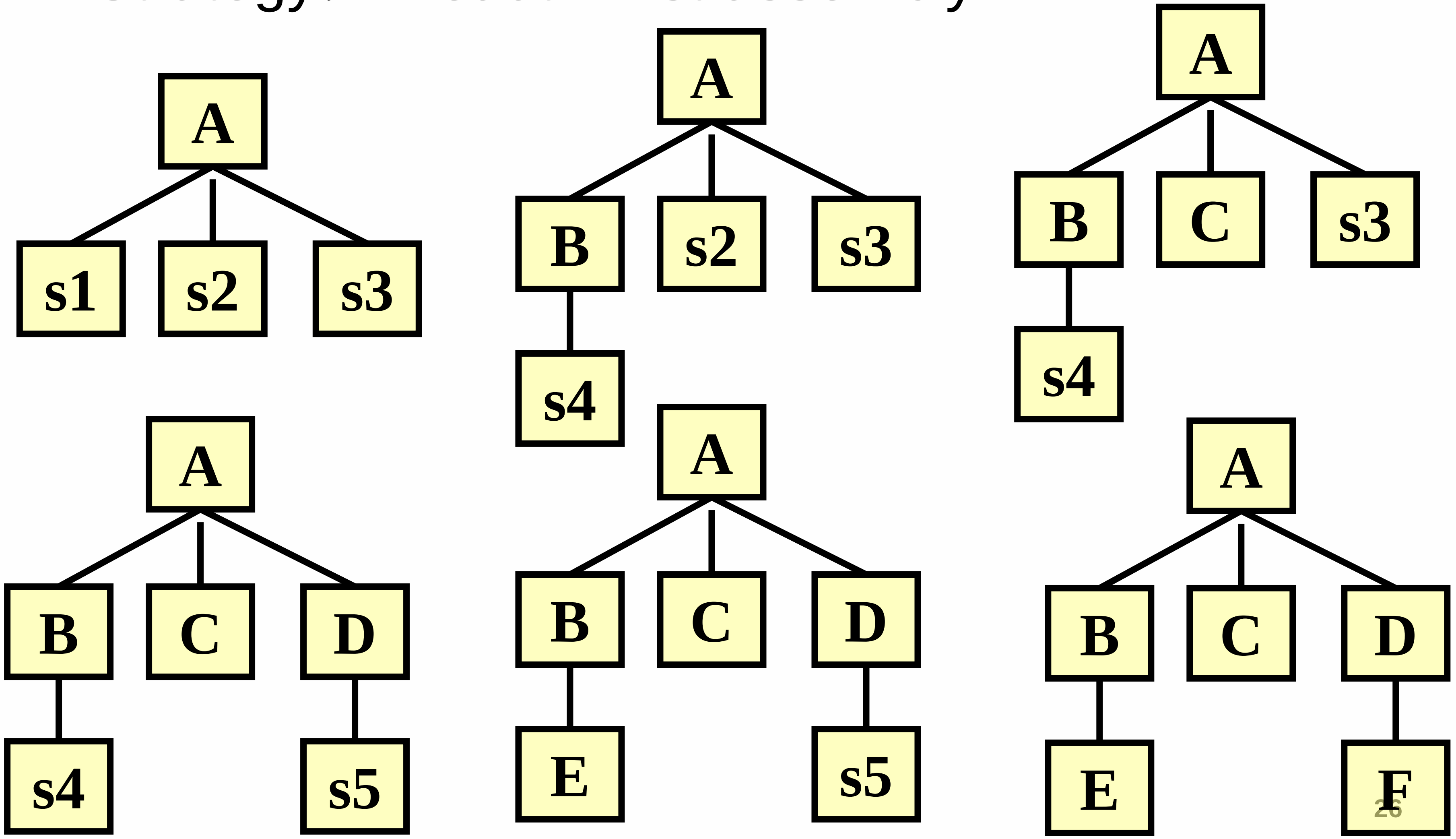
7.2 Integration Testing Strategy

- Strategy：Depth-first assembly

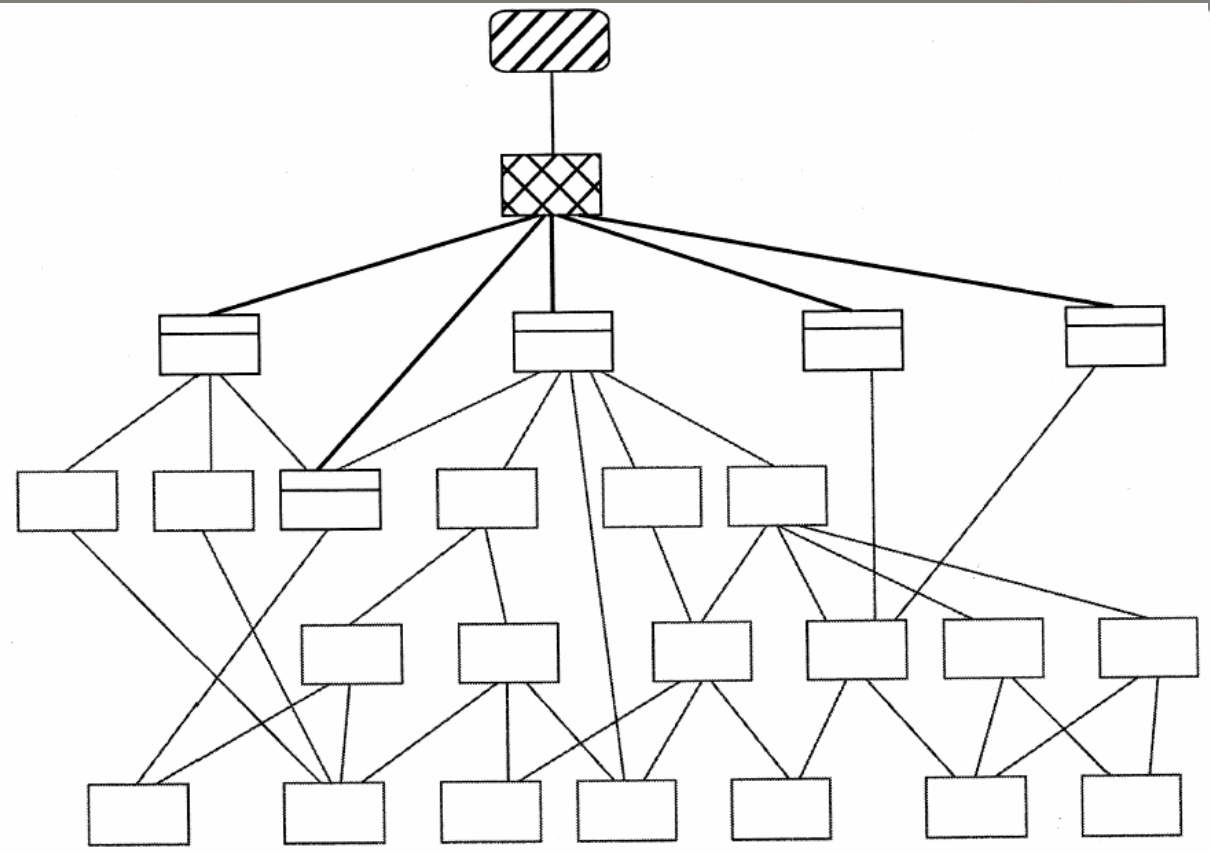


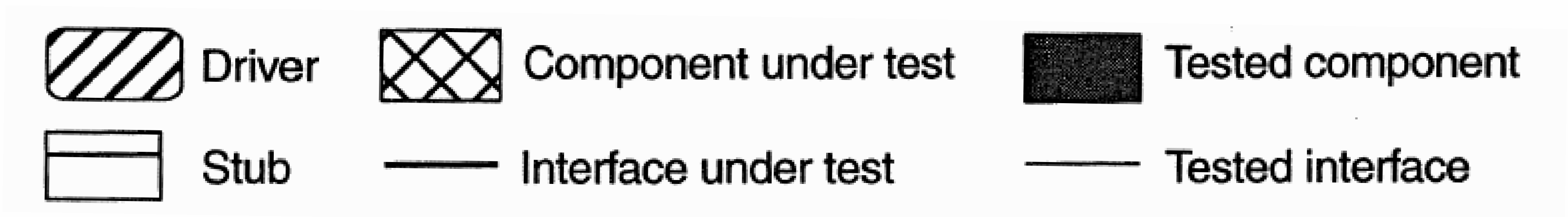
7.2 Integration Testing Strategy

- Strategy：Breadth-first assembly

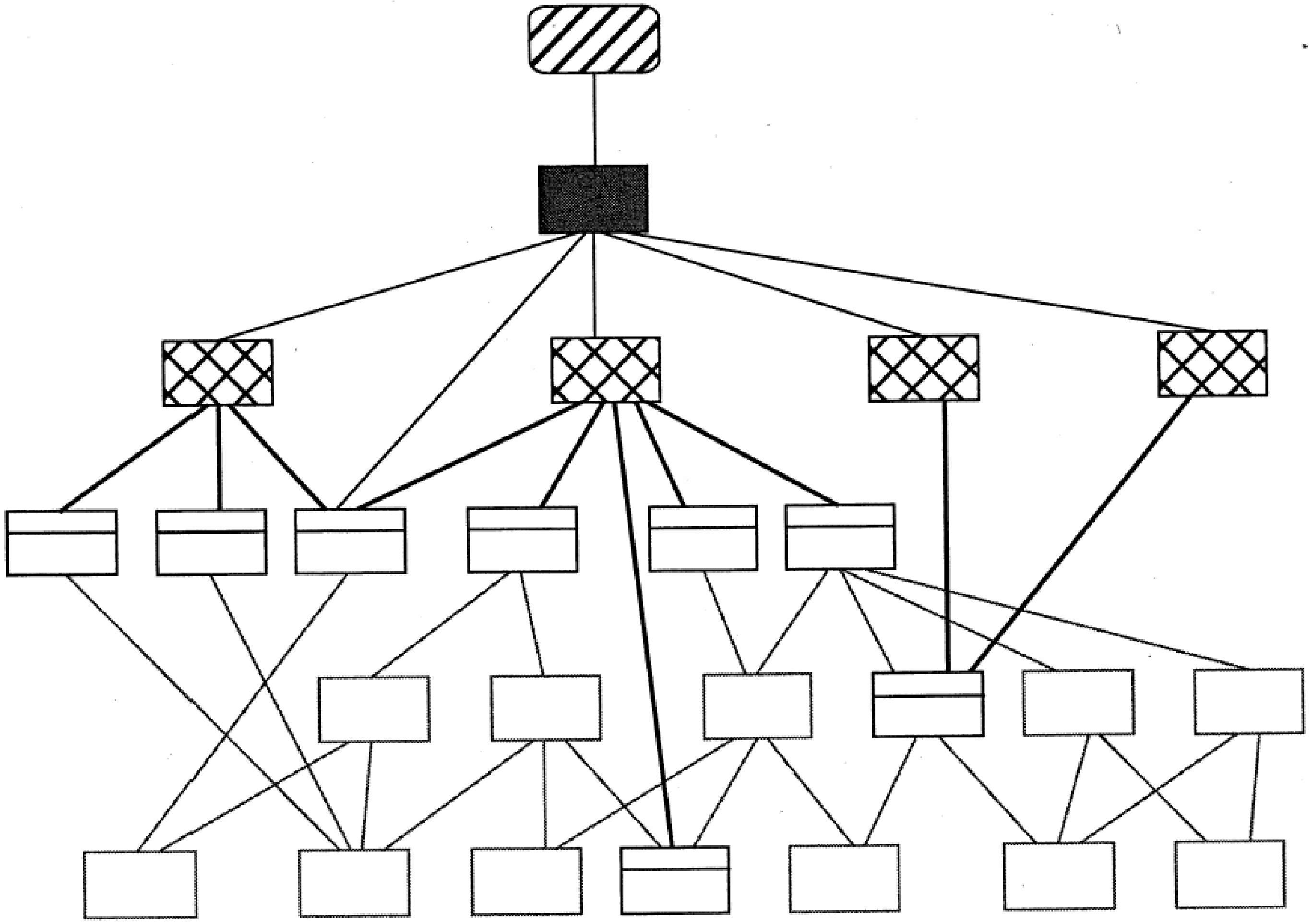


7.2 Integration Testing Strategy

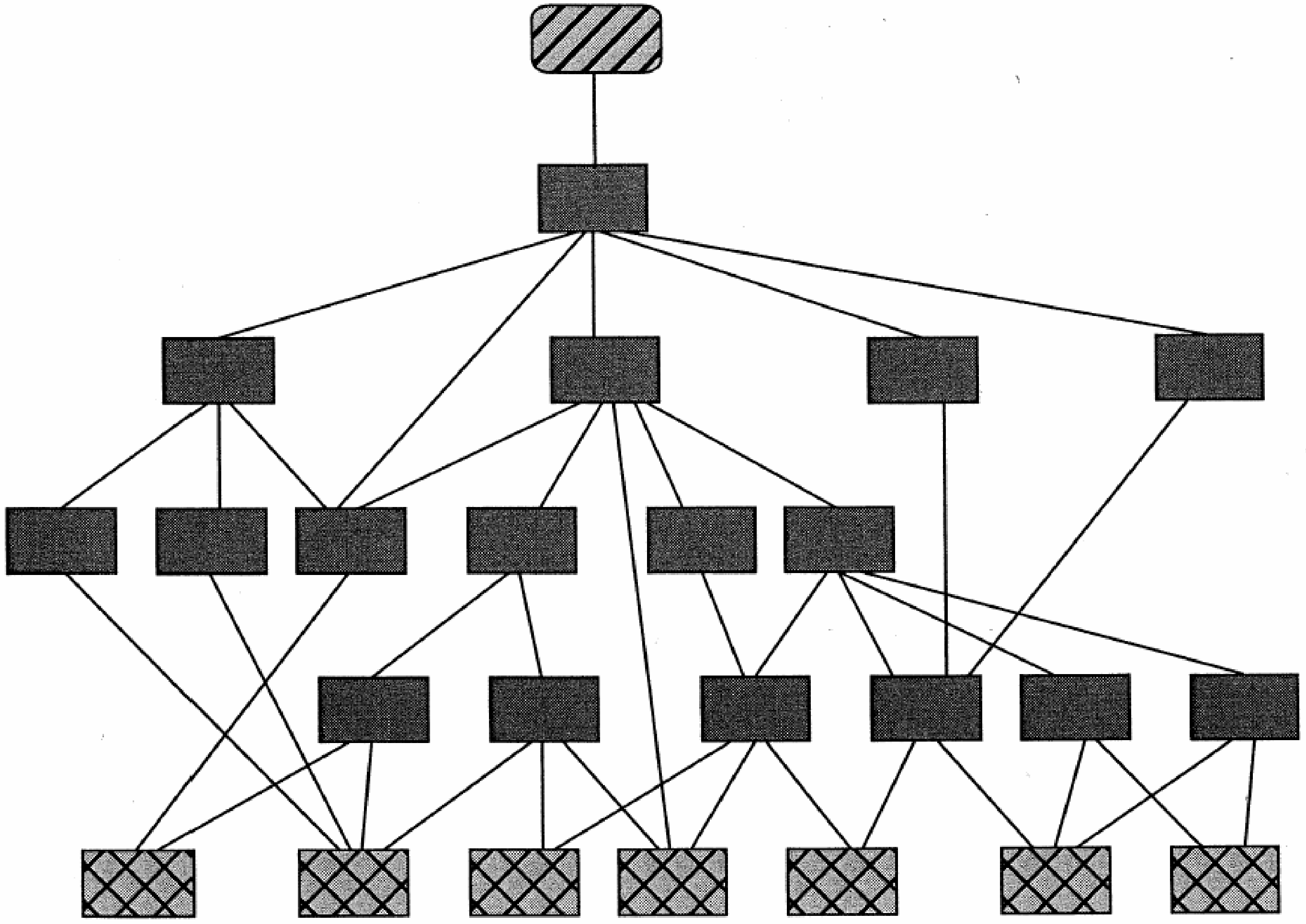




7.2 Integration Testing Strategy



7.2 Integration Testing Strategy



7.2 Integration Testing Strategy

- When stubs can't convey correct and useful information, some other methods can be used
  - Many tests can be postponed to stubs replaced with real modules.
  - Further developing module can simulate the actual function of stubs.
  - Bottom-up integration

7.2 Integration Testing Strategy

- Advantage
  - Early show main points of control and judgments.
  - Use depth-first assembly, and a complete software function can be firstly implemented and validated.
  - Only one driver module is needed at most.
  - Support fault isolation.
- Disadvantage
  - The costs of the development and the maintenance of stubs are both higher.
  - Bottom level components demand can not be predicted may lead to many amendments to the top level components.

7.2 Integration Testing Strategy

- Scope
  - Product control structure comparison clarity and stability.
  - Senior product interface changes relatively small.
  - Products bottom interface undefined or could be changed frequently.
  - Product control module has great technical risk and need to be validated as soon as possible.

7.2 Integration Testing Strategy

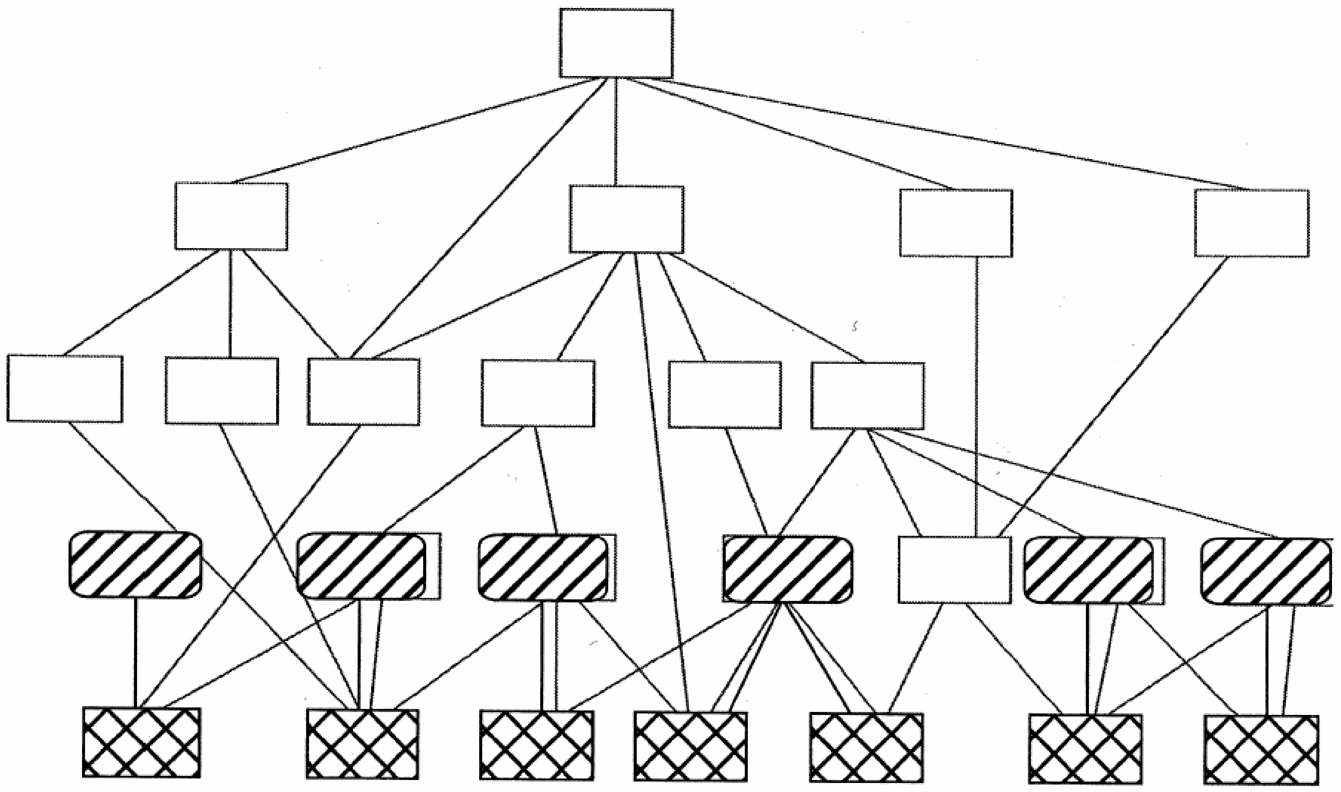
- Bottom-up integration
  - Start from the bottom with minimal dependent components, according to dependency structure, layers upward integration to detect the entire system stability

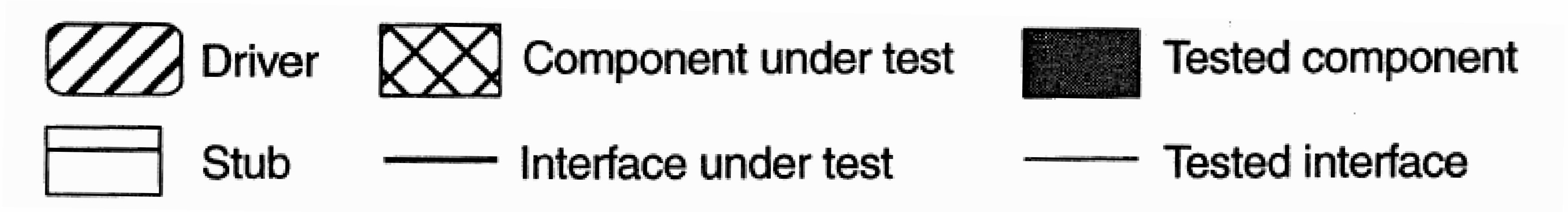
7.2 Integration Testing Strategy

- (1) For a given level of module, its child module (including child module subordinates module) has the assembly and test, so there is no need stub.
- (2) From the bottom modules beginning, also can add two or more leaves module merge to test together.
- (3) Development driver modules to test the module in step2, then instead of the driver module with actual module, and merge tested module as a big modules to test.
- (4) Repeat the process until top-level module test.

7.2 Integration Testing Strategy

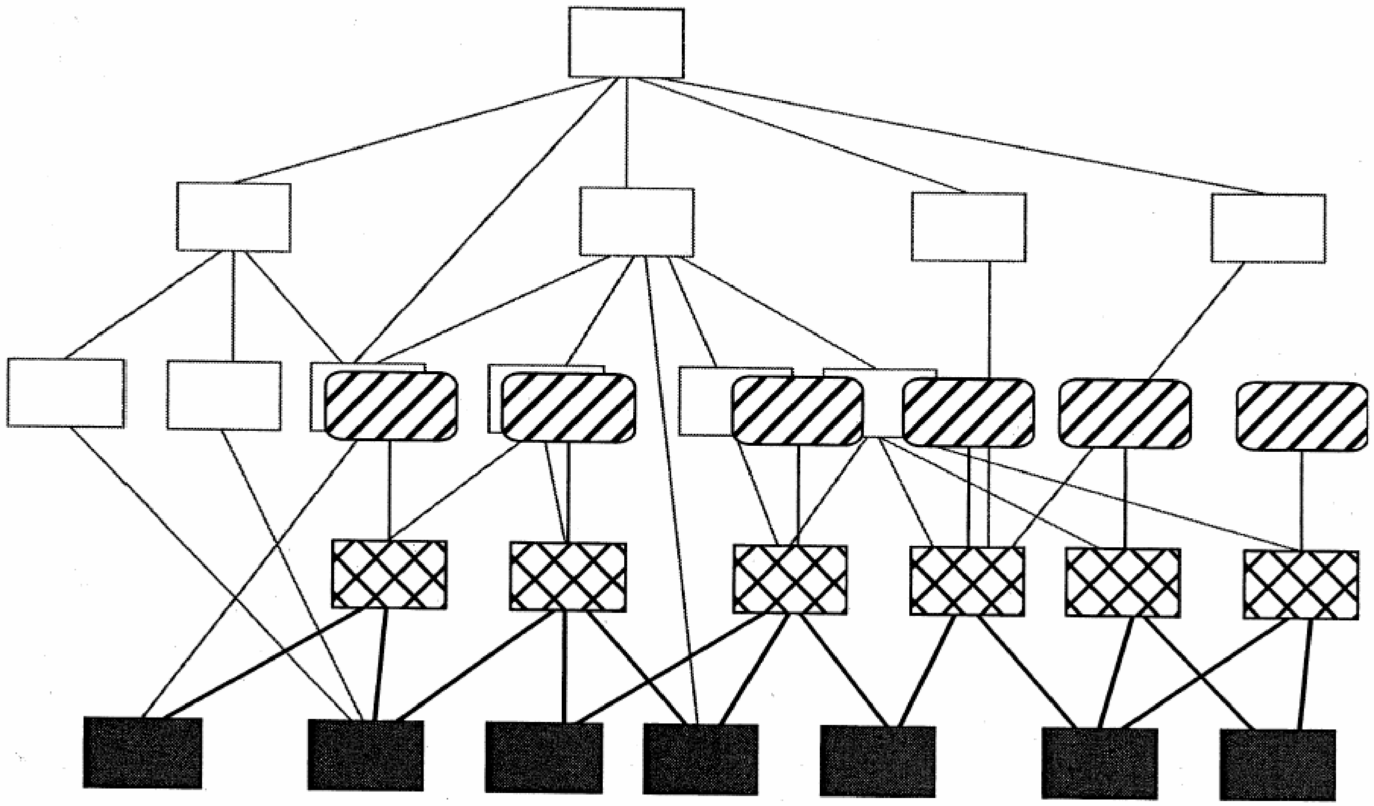
- Strategy
  - stage1





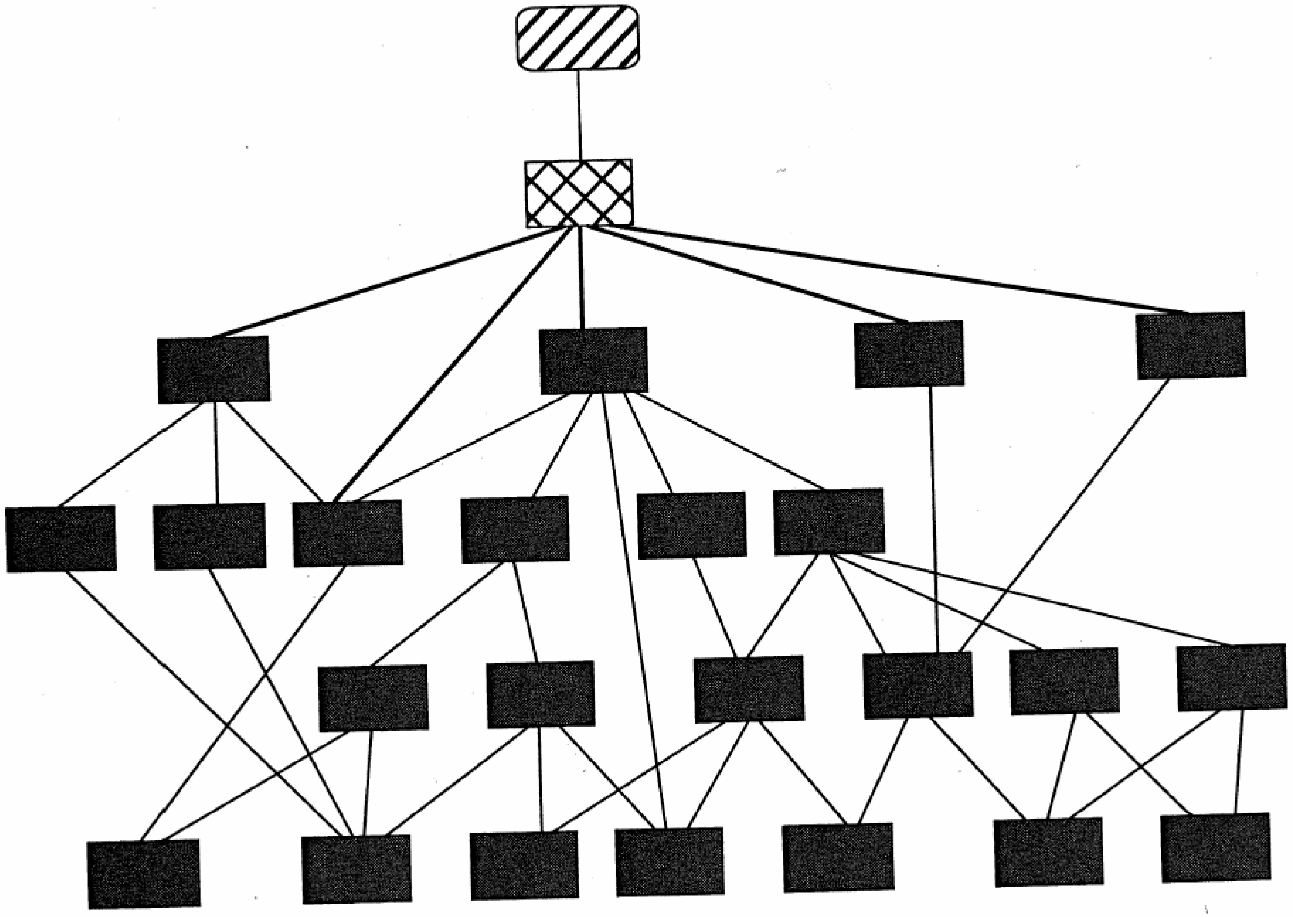
7.2 Integration Testing Strategy

- Strategy
  - stage2



7.2 Integration Testing Strategy

- Strategy
  - stage3



7.2 Integration Testing Strategy

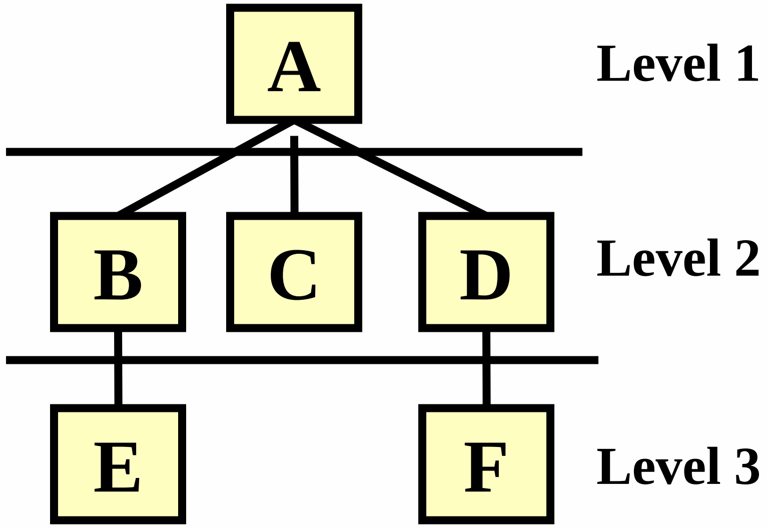
- Advantage
  - Allow conducting early certification for bottom modules, any leaf node is ready for integration testing can be tested.
  - Reduce workload of stub development
  - Support fault isolation
- Disadvantage
  - Driver module development huge workload comparison.
  - Senior validation was deferred until the end, design error can not be found in time.
  - Bottom abnormal hard cover.

7.2 Integration Testing Strategy

- Scope
  - Products which bottom interface is relatively stable and high-level interface changes more frequently.
  - Products which bottom module has been earlier completed.

7.2 Integration Testing Strategy

- Sandwich integration
  - – Combine the advantage of top-down strategy and bottom-up strategy.
- Strategy



- **Test A first, develop stubs substitute B, C, D.**
- **Test E, F, use drivers substitute B, D.**
- **Test (B, E) (D, F), use driver substitute A.**
- **Integrate together at last.**

7.2 Integration Testing Strategy

- Advantage
  - Combine the advantage of top-down strategy and bottom-up strategy.
- Disadvantage
  - The testing of middle level is insufficient before integration testing.
- Scope
  - It is used by the majority of software development projects.

7.2 Integration Testing Strategy

- Layers integration
  - Validate stability and interoperability of a specific level architecture application system by incremental integration method.
- Strategy
  - Layers partitioning for system.
  - Determine integration strategy internal levels.
  - Determine integration strategy between levels.
- Scope
  - Communication software.
  - A clear layers product system.

7.2 Integration Testing Strategy

- High-frequency integration
  - Frequent new code has been added to a stable baseline in order to detect integration fault and control possible baseline deviation.
- Conditions required by high-frequency integration
  - Get a stable increment, and a completed subsystem has been certified no problem.
  - Most meaningful increasing functions can be fixed within a frequent interval, such as daily build.
  - Test kits and code concurrent development, and always maintain the latest version.
  - Use automation.
  - Use configuration management tools, otherwise increment version will be out of control.

7.2 Integration Testing Strategy

- Strategy
  - Developers provide complete incremental codes and tester complete development of related test packets at the same time.
  - Testers amend or add components together to form a new integrated body, and run integration testing kits on it.
  - Evaluation results

7.2 Integration Testing Strategy

- Advantage
  - Errors prevent effectively.
  - Serious errors, omissions and incorrect assumptions can be revealed earlier.
  - Error positioning relatively easy.
  - Reduce the development of stub codes and driver codes.
  - Development and integration can be simultaneously.
- Disadvantage
  - It may be difficult to integrate smoothly at the first several cycles .
  - High-frequency integration frequency needs a good grasp.
- Scope
  - Products developed by the iterative process model .

7.2 Integration Testing Strategy

- Creating a daily build is very popular among many organization
  - It facilitates to a faster delivery of the system
  - It puts emphasis on small incremental testing
  - It steadily increases number of test cases
  - The system is tested using automated, re-usable test cases
  - An effort is made to fix the defects that were found within 24 hours
  - Prior version of the build are retained for references and rollback
  - A typical practice is to retain the past 7-10 builds

7.2 Integration Testing Strategy

- Event-based Integration
  - From the certification path correctness of message, integrate the system incrementally to verify system stability.
- Strategy
  - From the external system, analysis the possible system input sets.
  - Select a message analyzes the modules which were passed through.
  - Integrate these modules and test the message interface.
  - Repeat above steps until all message being tested.
- Advantage
  - Verify a message may need several modules, and the progress will be faster.
  - Reduce driver module development .

7.2 Integration Testing Strategy

- Disadvantage
  - For complex system, the interrelationship among message may be complex and difficult to analyzed.
  - For interface testing, it is inadequate.
- Scope
  - Object-oriented systems.
  - Based on finite state machine embedded systems.

7.2 Integration Testing Strategy

- A test strategy is created for each new build（模块集）and the following issues are addressed while planning a test strategy
  - What test cases need to be selected from the SIT test plan?
  - What previously failed test cases should now be re-executed in order to test the fixes in the new build?
  - How to determine the scope of a partial regression tests?
  - What are the estimated time, resource demand, and cost to test this build?

7.3 Integration Testing Analysis

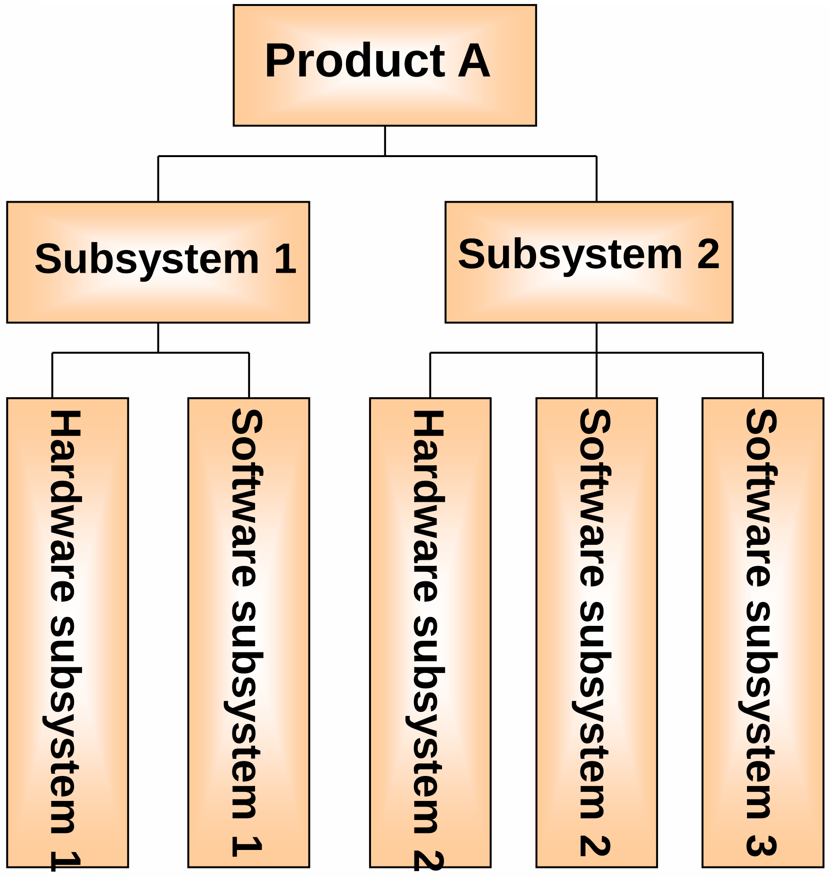
- The concern contents of integration testing
  - 1.Architecture analysis
  - 2.Module analysis
  - 3.Interface analysis
  - 4.Testability analysis
  - 5.Integration testing strategy analysis
  - 6.Common integration testing fault

7.3 Integration Testing Analysis

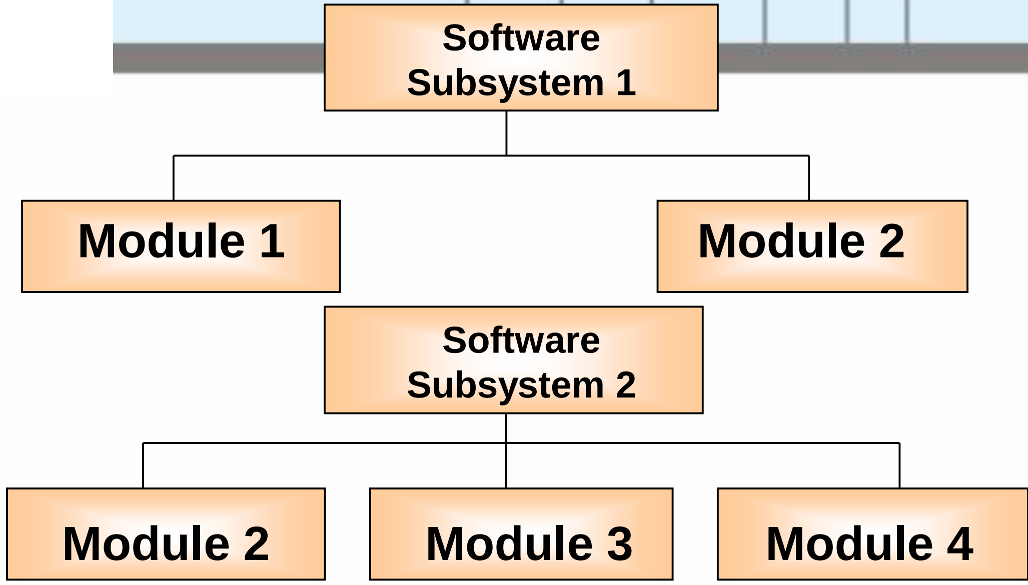
- 1.Architecture Analysis
  - Start from the tracking of requirement, divide the structure levels of system implementation.
    - Purpose
      - The integrated level exam slightly is helpful.
  - Find out the dependency relationship among components.
    - Purpose
      - Identify particle size of integration testing, that is foundation module size.

7.3 Integration Testing Analysis

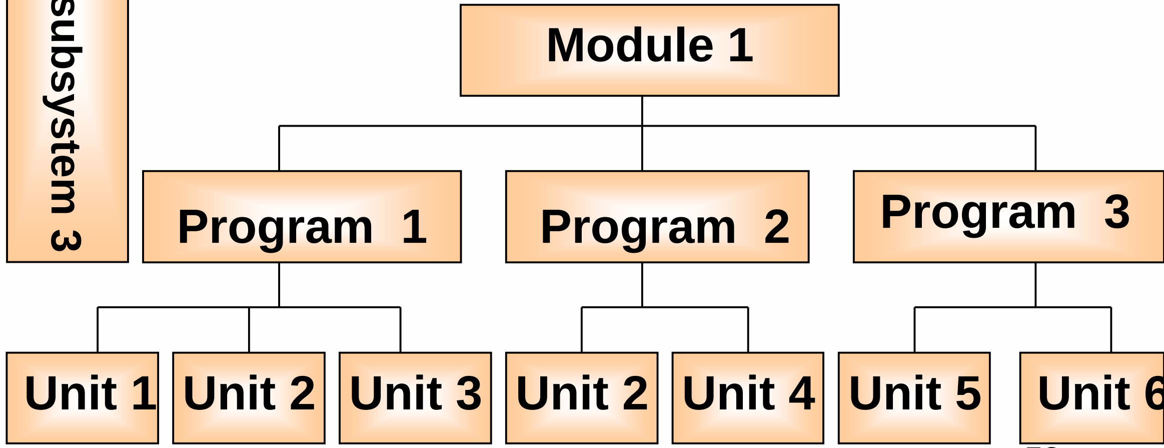
- System structure



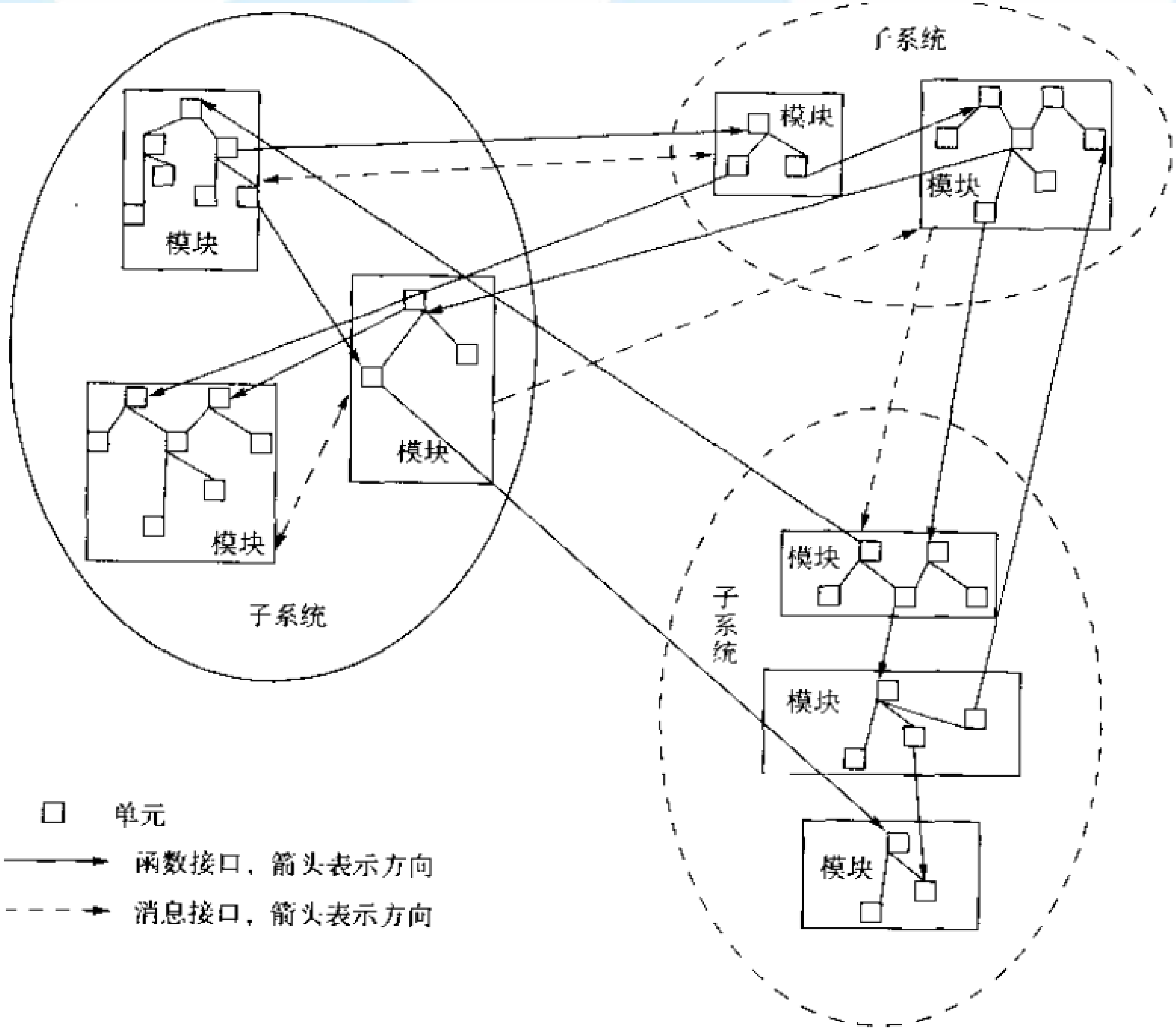
System structure graph



Software structure graph



Module structure graph



系统依赖关系示意图

7.3 Integration Testing Analysis

- 2.Module analysis
  - Clear the test modules.
  - Identify the relationships among the modules and then integrate the most related modules firstly.
  - Integrate low coupling modules by turns.

7.3 Integration Testing Analysis

- 3.Interface analysis
  - Interface division
    - Determine the boundary of system, subsystem and module.
    - Determine module internal interface, subsystem internal interface and system internal interface.
    - Determine interface between modules or subsystems.
    - Determine interface between system and operating system, hardware and software interface and third party.

7.3 Integration Testing Analysis

- **3. Interface analysis**
  - Interface classification
    - Function interface
    - Message interface
    - Class interface
  - Interface data analysis
    - Analysis of data across interfaces.
    - Function interface focuses on parameter number, order, property, etc.
    - Message interface focuses on message type, message domain, etc.
    - Class interface focuses on property and behavior.

7.3 Integration Testing Analysis

- **4.Testability analysis**
  - Mainly focus the testability declination on the increasing integration. So put fully attention to the interfaces can not be tested and find a solution as soon as possible.
- **5.Integration testing strategy analysis**
  - Good integration test strategy main concern
    - Tested objects can be test sufficiently, especially key modules.
    - Modules and interfaces clear.
    - Input resources are fully utilized.

7.3 Integration Testing Analysis

- 6.Common integration testing fault
  - Configuration / version control error.
  - Omission, overlap or conflict of function.
  - Document or database use incorrectly or inconsistently.
  - Wrong object and information binding. Wrong parameters or incorrect parameter values.
  - Conflict between components
  - Resource competition.

7.4 Integration Test Case Design

- Design ideas and methods used
  - 1.For the operation of the system design test cases.
  - 2.Design test cases for positive testing.
  - 3.Design test cases for reverse testing.
  - 4.Design test cases for special requirement.
  - 5.Design test case for meet coverage.
  - 6.Test cases complement.

7.4 Integration Test Case Design

- 1.For the operation of the system design test cases.
  - In Integration testing one need concern is the interface can be used, and it will not block the follow-up of integration testing execution. So testers can according to this principle to design some basic function test cases for minimum function limit validation.
  - Technology and method
    - Equivalence partitioning method
    - Standardize export method（规范导出法）

7.4 Integration Test Case Design

- Standardize export method（规范导出法）
  - According to the corresponding standard descriptions to design test cases.
  - Each test case is used to test one or more standard statements.
  - Basically it is used according to statements from the order of regulating statement design relevant test cases.
  - This method achieved very good correspondence between test cases and standardize statements. Strengthen the standardized readability and maintainability.
  - Is a positive test technology, need to reverse test technology to complement of test cases.
  - When the standards of certain cases require different processing, each statements may need more test cases.

7.4 Integration Test Case Design

- 例子: 计算平方根的函数的规范
  输入：实数；输出：实数
  规范：当输入一个0或比0大的数的时候，返回其正平方根；当输入一个小于0的数时，显示错误信息“平方根非法-输入值小于0”并返回0；库函数Print-Line可以用来输出错误信息。
  - 由规范的3个陈述，可以得到两个用例
  - 用例1：输入4，输出应该为2。
    对应于第一个陈述(当输入一个0或比0大的数时…)
    用例2：输入-1，输出0，并显示错误提示信息。
    对应于第二，第三个陈述（当输入一个小于0的数时，显示错误信息“平方根非法-输入值小于0”并返回0；库函数Print-Line可以用来输出错误信息。）

7.4 Integration Test Case Design

- 2.Design test cases for positive testing
  - The requirement of interface design and module function design is clear, and correct.
  - Testing is focused on the validation of interface requirement and the integrated module function be correctly satisfied.
  - According to outline design document export related test cases.

7.4 Integration Test Case Design

- Technology and method
  - Standard export method
  - Input test
  - Output coverage test
  - Equivalence partitioning
  - State transition testing

7.4 Integration Test Case Design

- 3.Design test cases for reverse testing
  - Reverse testing includes analysis of whether interface achieve the function that the specification do not require, specifications of possible interface omission or interface definition errors, analysis possible abnormalities of interface, including interface in the wrong data, interface data sequence errors.

7.4 Integration Test Case Design

- Technology and method
  - Error guessing method
  - Based on the risk of testing
  - Based on fault testing
  - Boundary value analysis
  - Special value test
  - State transition method

7.4 Integration Test Case Design

- 4.Design test cases for special requirement
  - Safety testing, performance testing, reliability testing should be in system testing phases, but there are some products in the module design document was made clear these indicators, that should test the interface for special requirement as early as possible.
  - Technology and method
    - Standardize export method

7.4 Integration Test Case Design

- 5.Design test case for meet coverage
  - Function cover
  - Interface cover
  - Technology and method
- 6.Test cases complement
  - Added functions, characteristics modify, defect modification
- Note
  - Cost, schedule and quality
  - Highlight the test key, key interface must be covered
  - Fully considering regression and executive automation

7.5 Integration Testing Process

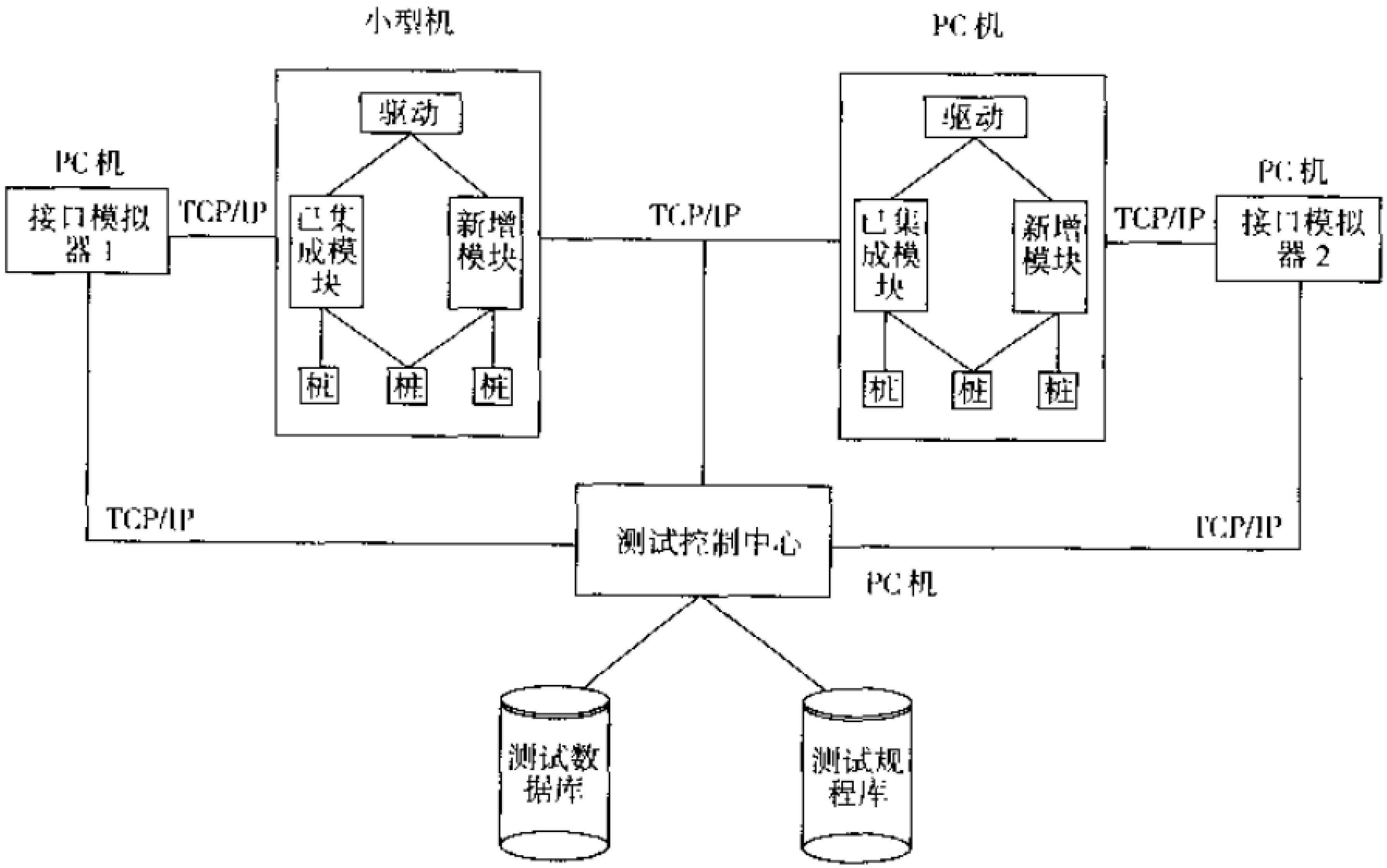
- IEEE Std 1023-1998.Software Verification and Validation Plan Guideline.
- The process of integration testing can be divided into four stages:
  - Planning stage
  - Designing stage
  - Implementation stage
  - Performance stage

7.6 Integration Testing Environment

- For a simple system (single process software), integration test environment and unit test environment is similar.
- For complex systems, often distribution in different hardware and software platforms. Integration test environment is very complex, need to rely on some commercial test tools or tester, sometimes also need to develop some interface simulation tools.

7.6 Integration Testing Environment

- In considering integration testing environment, follow several aspects:
  - Hardware environment
  - Operating environment
  - Database environment
  - Network environment


小型机
PC 机
驱动
已集成模块
新增模块
桩
桩
桩
驱动
已集成模块
新增模块
桩
桩
桩
PC 机
接口模拟器 1
TCP/IP
TCP/IP
TCP/IP
PC 机
接口模拟器 2
TCP/IP
测试控制中心
TCP/IP
PC 机
测试数据库
测试规程库

集成测试环境示意图

7.7 Principle of Integration Testing

- Key modules must be tested sufficiently.
- All interfaces must be tested.
- Integration testing should be according to certain levels.
- When the interface is changed, all related interfaces should be tested by using regression testing.
- Integration testing is conducted according to plan and prevents random testing.
- Integration testing strategy should integrate the relationship among quality, cost and progress.

Thank you !