软件测试过程

苏州大学计算机科学与技术学院

3 集成测试

1. 什么是集成测试
2. 集成测试的主要任务
3. 集成测试中的角色及工作职责
4. 集成测试策略
5. 集成测试技术
6. 集成测试原则
7. 集成测试分析
8. 集成测试用例设计
9. 集成测试过程
10. 集成测试环境

3.1 什么是集成测试

- 将经过单元测试的模块按照设计要求连接起来，组成所规定的软件系统的过程称为“集成”
- 集成测试又称组装测试、联合测试、部件测试

3.2 集成测试的主要任务

- 将各模块连接起来，检查模块相互调用时，数据经过接口是否丢失
- 将各个子功能组合起来，检查能否达到预期要求的各项功能
- 一个模块的功能是否会对另一个模块的功能产生不利的影响
- 全局数据结构是否有问题，会不会被异常修改
- 单个模块的误差积累起来，是否被放大，从而达到不可接受的程度

3.3 集成测试中的角色及工作职责

软件评测部：

角　色	职　责
测试设计员	负责制定集成测试计划、设计集成测试、实施集成测试、评估集成测试。
测试员	执行集成测试，记录测试结果。

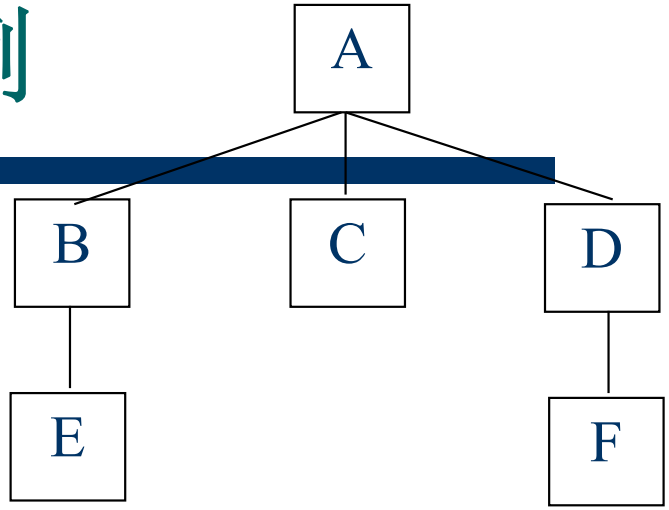
软件项目组：

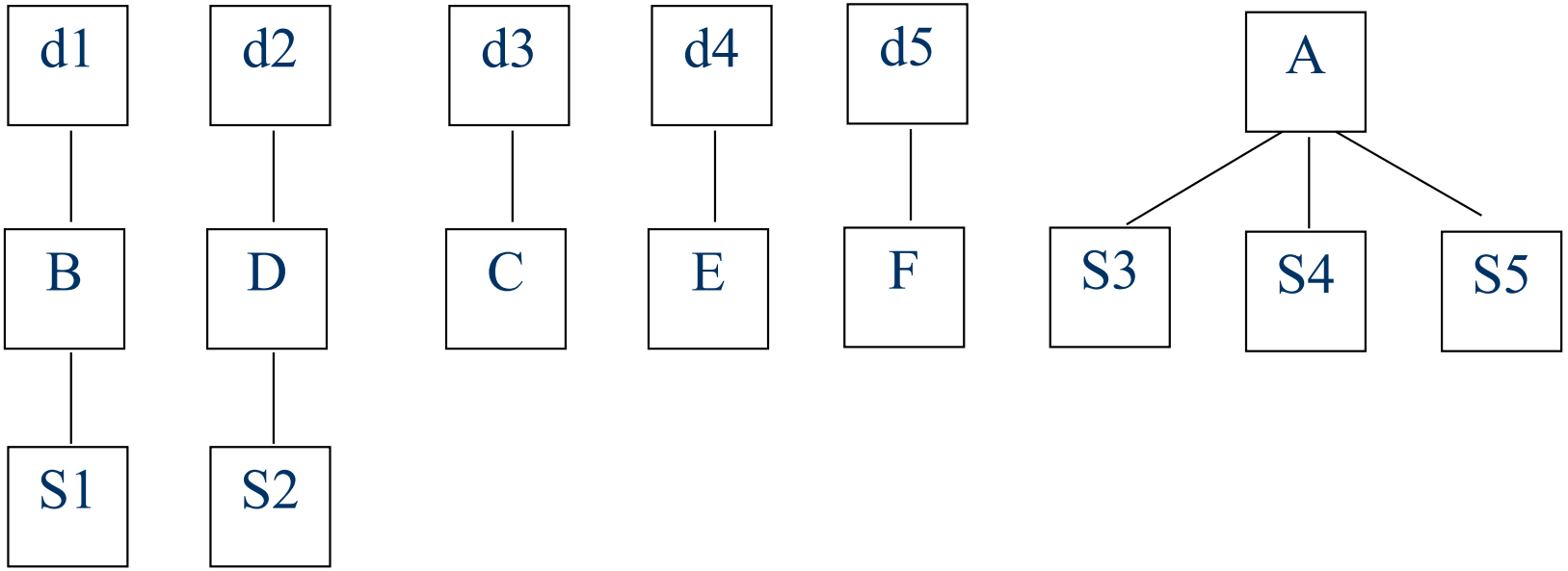
角　色	职　责
实施员	负责实施类（包括驱动程序和桩），并对其进行单元测试。根据集成测试发现的缺陷提出变更申请。
配置管理员	负责对测试工件进行配置管理。
集成员	负责制定集成构建计划，按照集成计划将通过了单元测试的类集成。
设计员	负责设计测试驱动程序和桩。根据集成测试发现的缺陷提出变更申请。

3.4 集成测试策略(1/3)

- 集成测试的方法
  - 基于分解的集成
  - 基于调用图的集成
  - 高频集成
  - …
- 基于功能分解的集成测试：
  - 大爆炸集成 （非增量式集成）
  - 自顶向下集成
  - 自底向上集成
  - 三明治集成

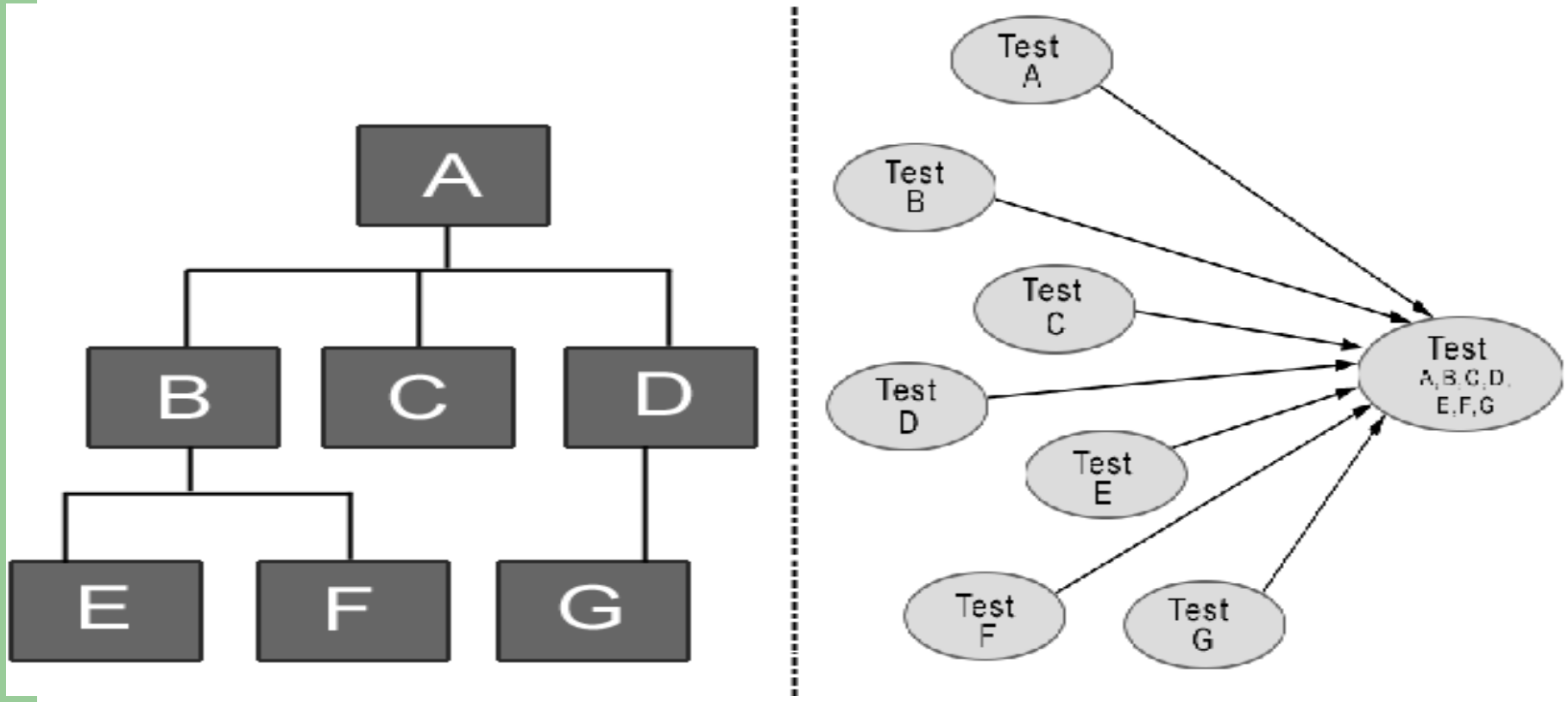
非增量式集成测试示例





大棒集成方法(Big-bang Integration)

- 大棒集成方法：先对每一个子模块进行测试（单元测试阶段），然后将所有模块一次性的全部集成起来进行集成测试 。



非渐增式集成测试的特点

- 优点
  - 迅速完成集成测试
  - 测试用例较少
- 缺点
  - 错误难以定位
  - 即使通过测试，许多接口错误也会隐藏
- 应用
  - 小的、良构的系统，其模块已接受了充分的测试
  - 一个已经存在的系统，只是做了少量的修改
  - 通过复用可信赖的模块构造系统

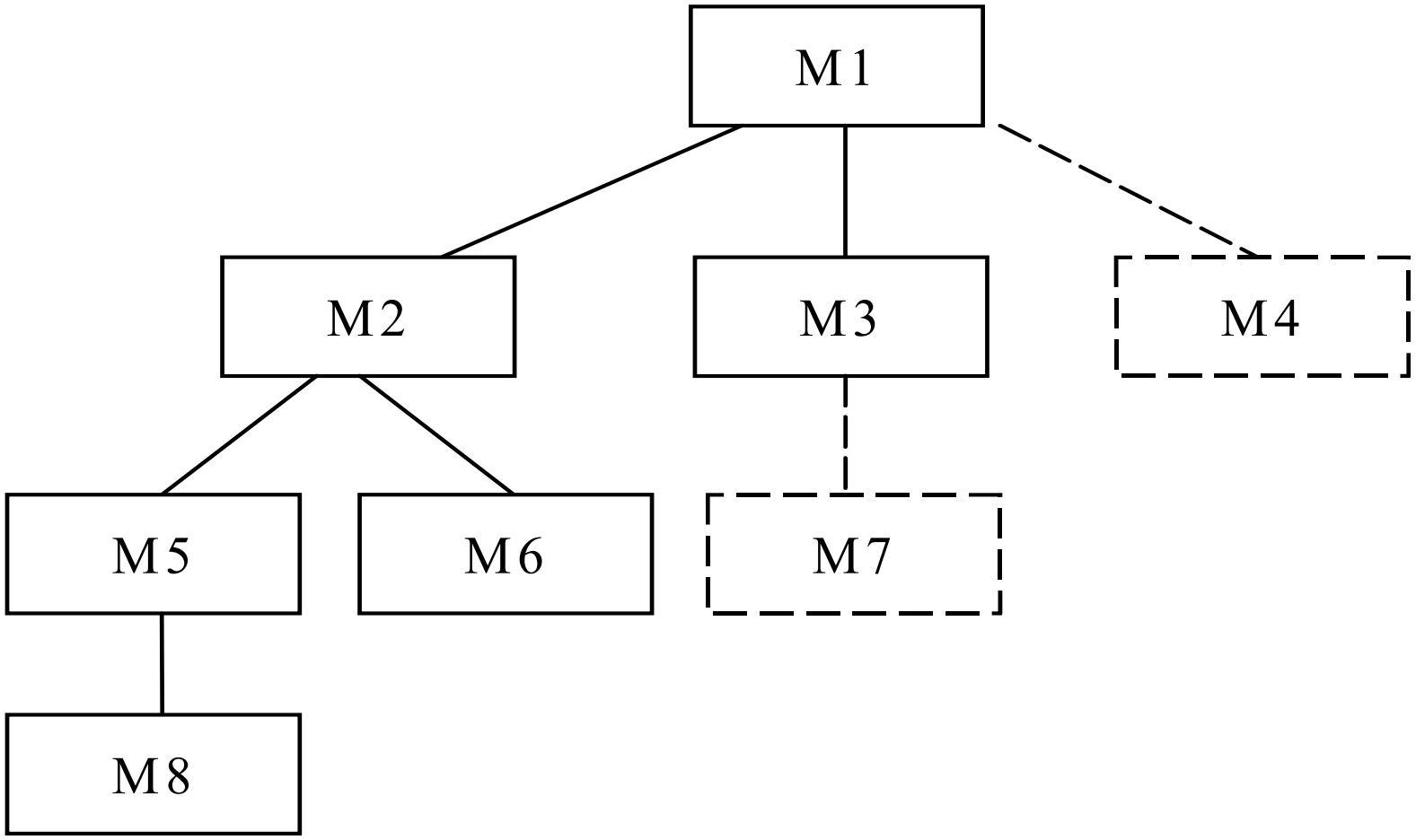
3.4 集成测试策略(2/3)

- 增量式集成
  - 自顶向下增量式测试
    - 按结构图自上而下逐步集成和逐步测试
    - 首先集成主控模块（主程序），然后按照软件控制层次结构向下进行集成
  - 自底向上增量式测试
    - 从最底层的模块开始，按结构图自下而上逐步进行集成和测试
  - 混合集成（三明治集成）
    - 以中间层为目标层，以上用自顶向下，以下用自底向上

自顶向下增量式集成测试的步骤

- 以主控模块作为测试驱动器，把对主控模块进行单元测试时引入的被调用模拟子模块用实际模块替代
- 依照所选用的模块集成策略，下层的被调用模拟子模块一次一个地被替换为真正的模块
- 在每个模块被集成时，都必须立即进行一遍测试

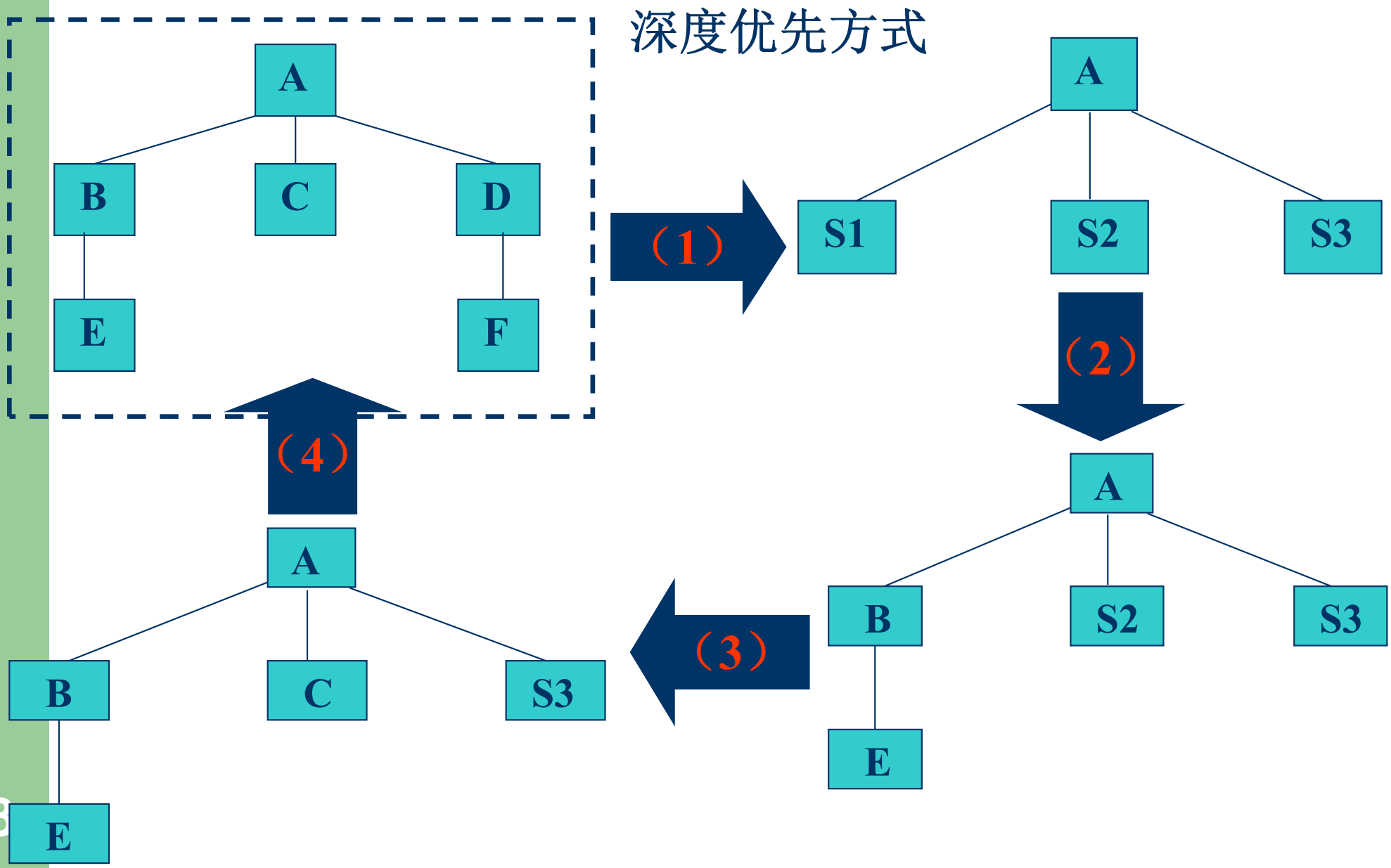
自顶向下集成测试示例



深度优先：M1, M2, M5, M8, M6, M3, M7, M4

广度优先：M1, M2, M3, M4, M5, M6, M7, M8

自顶向下测试（续）



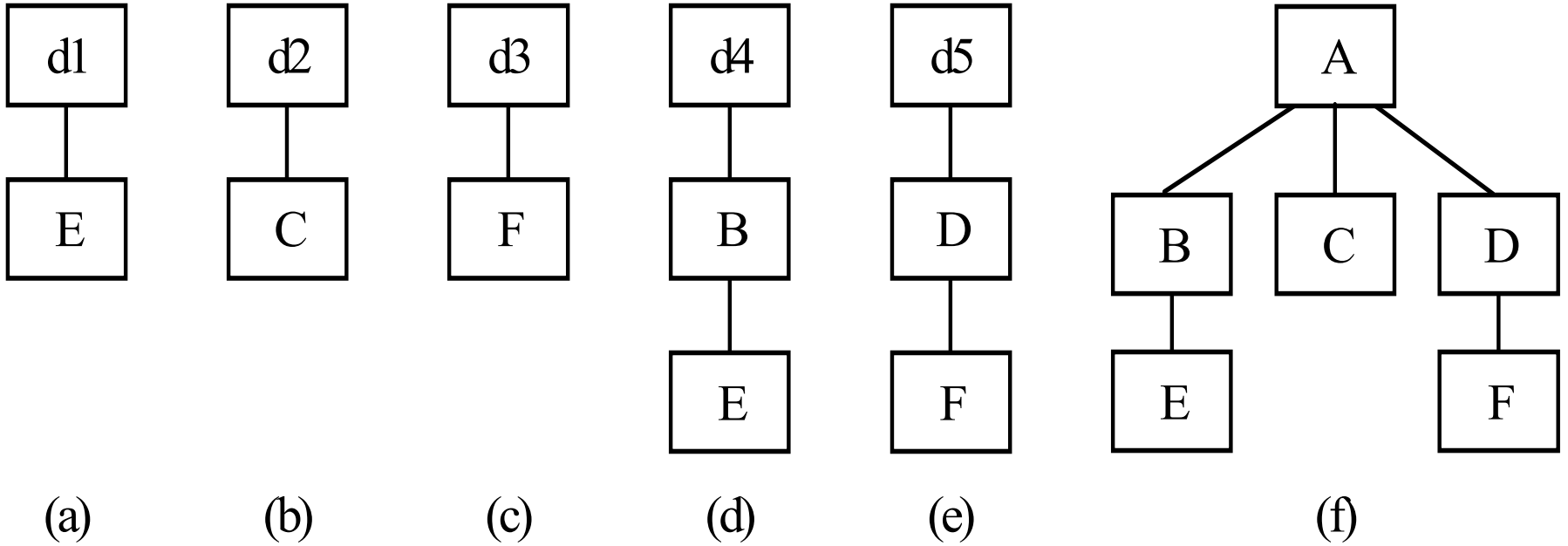
自顶向下集成测试的特点

- 优点
  - 测试和集成可以较早的开始
  - 减少了驱动器的开发
  - 如果底层接口未定义或可能修改，则可以避免提交不稳定的接口
- 缺点
  - 桩的开发代价较大
  - 在底层模块中一个无法预料的需求可能迫使顶层模块的修改
  - 要充分测试底层模块可能比较困难
- 适用范围
  - 增量式开发
  - 并行软件开发

自底向上增量式集成测试的步骤

- 从最底层的模块开始，按结构图自下而上逐步进行集成和测试。
- 依照所选用的模块集成策略，将驱动模块 **(driver)** 用实际模块代替。
- 在每个模块被集成时，都必须立即进行一遍测试

自底向上集成测试示例



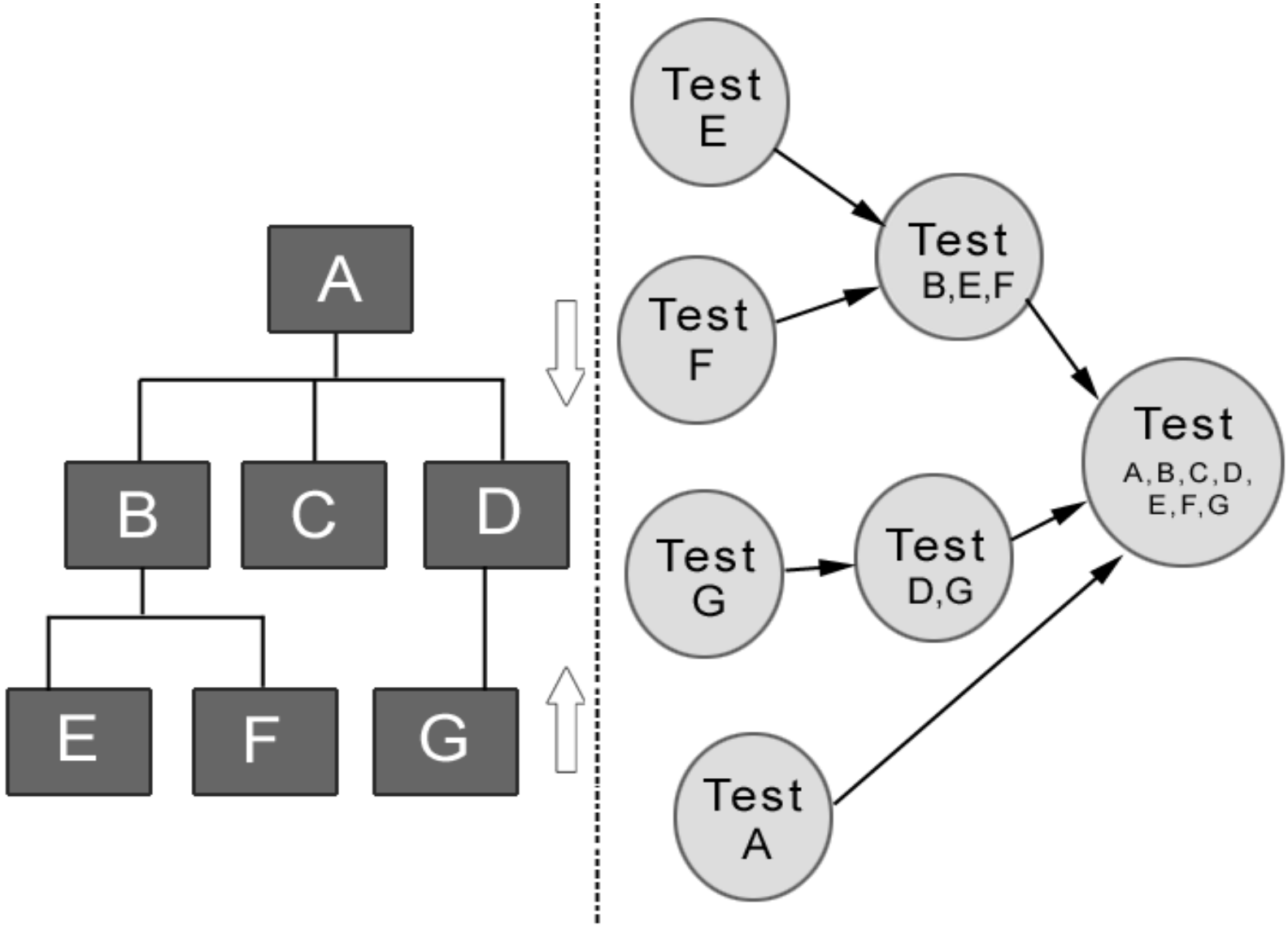
自底向上集成测试的优点

- 无需构造桩模块，桩模块往往千差万别。
- 驱动模块具有某种统一性，且随着测试层次的提供，驱动模块数量减少。
- 涉及复杂算法和真正输入、输出的模块一般在底层，且是较易出错的模块。可以尽早发现错误。
- 各子树的集成和测试可以并行。

自底向上集成测试的缺点

- 缺点
  - 驱动器的开发耗费量大
  - 高层模块的可操作性和互操作性测试得不充分
- 适用范围
  - 重要需求的模块在底层

混合集成（三明治集成）


A
B
C
D
E
F
G
Test E
Test F
Test B,E,F
Test G
Test D,G
Test A,B,C,D,E,F,G
Test A

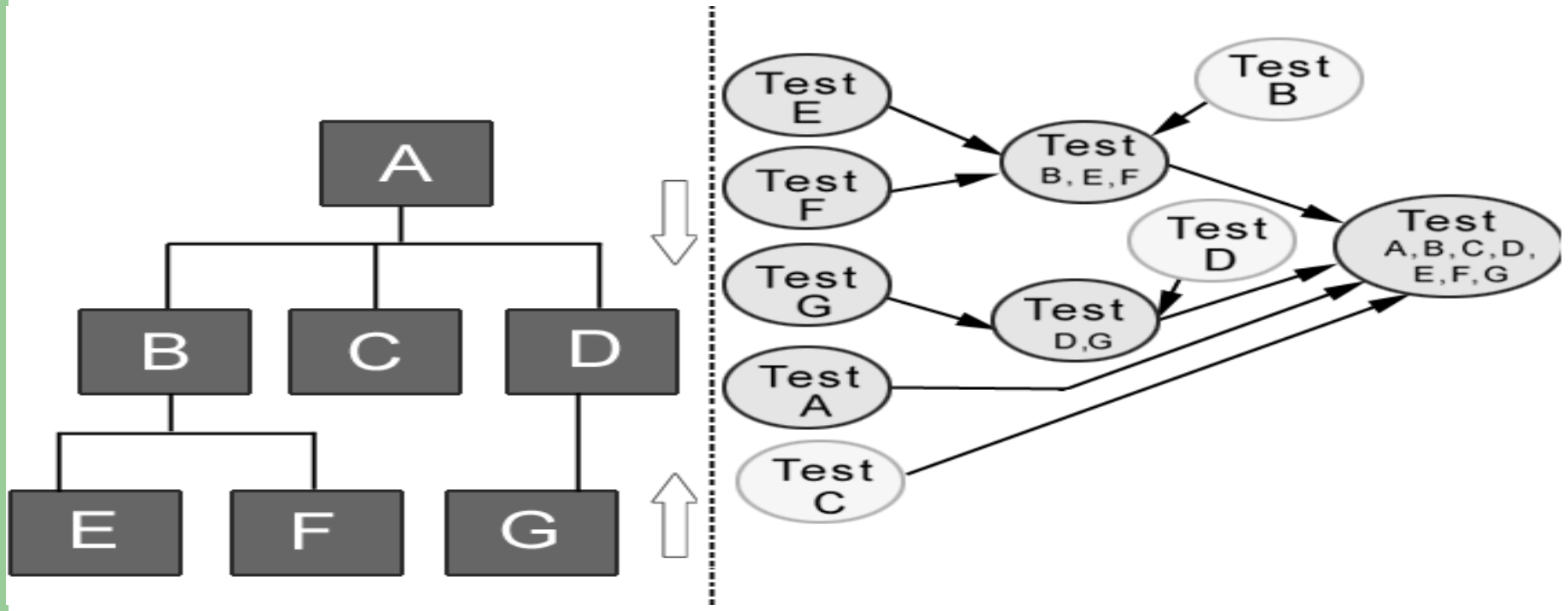
混合集成（三明治集成）

- 采用三明治方法的优点是：它将自顶向下和自底向上的集成方法有机地结合起来，不需要写桩程序因为在测试初自底向上集成已经验证了底层模块的正确性。

- 采用这种方法的主要缺点是：在真正集成之前每一个独立的模块没有完全测试过。
  - *改进的三明治集成方法*

改进的三明治集成方法

- 改进的三明治集成方法，不仅自两头向中间集成，而且保证每个模块得到单独的测试，使测试进行得比较彻底



基于调用图的集成

- 基于分解的集成—以功能分解树为基础
- 基于调用图的集成—向结构性测试方向发展
- 基于调用图的集成测试方法
  - 成对集成
  - 相邻集成

- 成对集成
  - 免除桩/驱动器的开发工作 -- 使用实际的代码
  - 为避免大爆炸集成 – 限制在调用图的一对单元上
  - 对调用图中的每条边有一个集成测试过程。

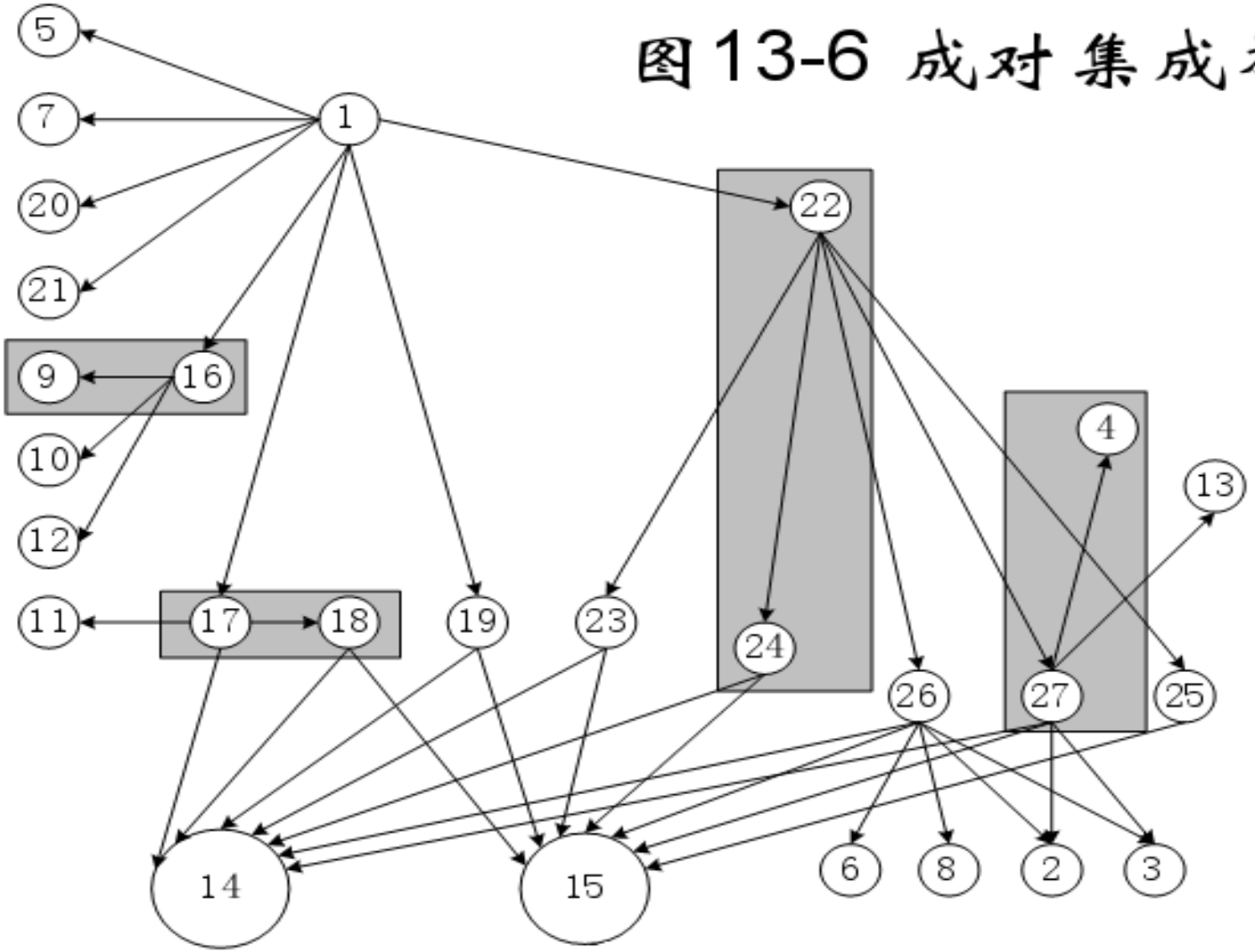


图13-6 成对集成举例

40次集成测试会话。

相邻集成

- 借用拓扑学中的邻接概念
- 在有向图中，结点邻居包括所有直接前驱结点和所有直接后继结点
- 这对应结点的桩和驱动器集合

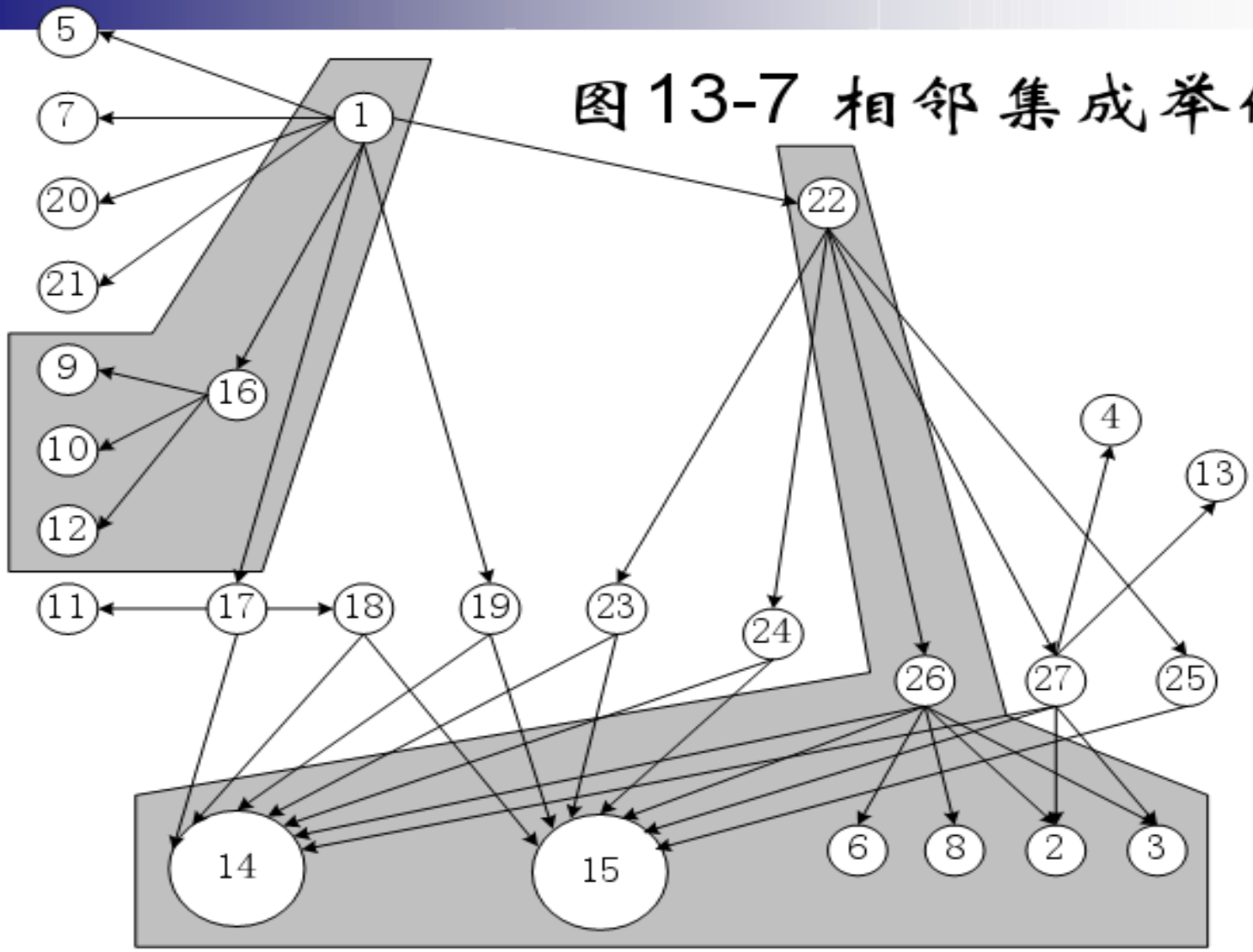


图13-7 相邻集成举例

节点	前驱	后继
16	1	9, 10, 12
17	1	11, 14, 18
18	17	14, 15
19	1	14, 15
23	22	14, 15
24	22	14, 15
26	22	14, 15, 6, 8, 2, 3
27	22	14, 15, 2, 3, 4, 13
25	22	15
22	1	23, 24, 26, 27, 25
1	-	5, 7, 20, 21, 16, 17, 19, 22

11次集成测试会话

基于调用图的集成测试的优点

- 免除了驱动器/桩的开发工作
- 接口关系测试充分
- 测试集中于衔接的功能性
- 测试和集成可以并行开始

基于调用图的集成测试的缺点

- 缺点
  - 调用或协作的关系可能是错综复杂的
  - 参与者没有被单独测试，要充分测试底层模块较困难
  - 特定的调用或协作可能是不完全的
  - 缺陷隔离
- 适用范围
  - 尽快论证一个可运行的调用或协作
  - 被测系统已清楚定义了模块的调用和协作关系

高频集成

- 高频集成测试是指同步于软件开发过程，每隔一段时间对开发团队的现有代码进行一次集成测试。

- 该集成测试方法频繁地将新代码加入到一个已经稳定的基线中，以免集成故障难以发现，同时控制可能出现的基线偏差。

高频集成

- 使用高频集成测试需要具备一定的条件:
  - 可以持续获得一个稳定的增量，并且该增量内部已被验证没有问题;
  - 大部分有意义的功能增加可以在一个相对稳定的时间间隔（如每个工作日）内获得;
  - 测试包和代码的开发工作必须是并行进行的，并且需要版本控制工具来保证始终维护的是测试脚本和代码的最新版本;
  - 必须借助于使用自动化工具来完成。

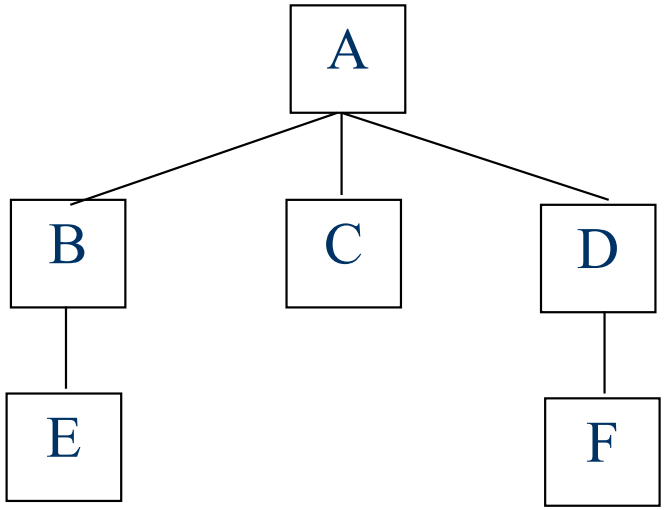
高频集成

- 优点：能在开发过程中及时发现代码错误，能直观地看到开发团队的有效工程进度。

- 缺点：需要开发和维护源代码和测试包。有时候可能不能暴露深层次的编码错误和图形界面错误。

3.4 集成测试策略(3/3)

- 基于功能的集成
  - 依据功能的优先级，逐一将某个功能上的各个模块集成起来
  - 核心系统先行集成测试
- 基于进度的集成
  - 将最早获得的模块进行集成，以最大程度保持与开发的并行性



非增量式测试VS增量式测试

1）非增量式测试是先分散测试，然后集中起来再一次完成集成测试。如果在模块的接口处存在错误，只会在最后的集成测试时一下暴露出来；很难确定出错的真正位置、所在的模块、错误的原因。这种方法适合在规模较小的应用系统中使用。

2）增量式测试的逐步集成和逐步测试的方法，把可能出现的差错分散暴露出来，便于找出问题和修改。模块在逐步继承的测试中得到了较为频繁的考验，因而可能取得较好的测试效果；

3）总的来说，增量式测试比非增量式测试具有一定的优越性。

3.5 集成测试技术

- 集成测试主要测试软件的结构问题，因为测试建立在模块的接口上，所以多为黑盒测试，适当辅以白盒测试
- 有时又把集成测试称作灰盒测试
- 集成测试一般也不使用真实数据，测试人员可以使用手工制作一部分代表性的测试数据。在创建测试数据时，应保证数据充分测试软件系统的边界条件。
- 在集成测试时可适当地重用单元测试时生成的数据。

3.6 集成测试原则

- 所有公共接口都要被测试到
- 关键模块 必须进行充分的测试
  - ① 对应几条需求。
  - ② 具有高层控制功能。
  - ③ 复杂，易出错。
  - ④ 有特殊的性能要求。
- 集成测试应当按一定的层次进行
- 集成测试的策略选择应当综合考虑质量、成本和进度之间的关系
- 集成测试应当尽早开始，并以总体设计为基础
- 在模块与接口的划分上，测试人员应当和开发人员进行充分的沟通
- 当接口发生修改时，涉及的相关接口必须进行再测试
- 测试执行结果应当如实记录

3.7 集成测试分析

- 体系结构分析
  - 确定基本模块的大小，合理划分测试的模块，为测试策略的确定提供依据
- 模块分析
  - 划分关键模块（基本模块、共享模块等）
  - 分清高危模块、一般模块和低危模块
- 接口分析
  - 集成测试的重点是接口的功能性、可靠性、安全性、完整性、稳定性等
- 可测试性分析
  - 一般可测试性会随集成范围的增加而下降
  - 尽早评估可测试性、有利于集成测试的可操作性分析
- 集成策略分析
  - 分析测试策略的优劣
  - 好的测试策略应该是测试充分、总成本低

3.8 集成测试用例设计

- 为正向测试设计用例
  - 正向：接口、功能的正确性
  - 输入输出域分析、等价分类法、状态转换
- 为逆向测试设计用例
  - 逆向：接口、功能的错误、异常、多余、遗漏
  - 不执行其不应该完成的工作
  - 错误猜测、边界值、状态转换、基于风险

3.9 集成测试过程

- 计划
- 设计
- 实施
- 执行
- 评估

3.10 集成测试环境

- 硬件
- 操作系统
- 数据库
- 网络
- 工具软件及其它支撑软件