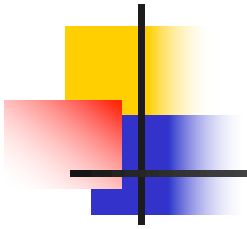
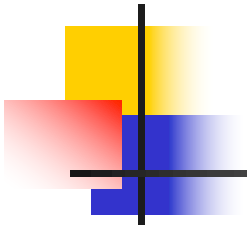
第6章 集成测试

张圣林
zhangsl@nankai.edu.cn
http://nkcs.iops.ai/courses/softwaretesting/



6 集成测试

- 6.1 集成测试概述
- 6.2 集成测试策略
- 6.3 集成测试用例设计
- 6.4 集成测试过程



6.1 集成测试概述

- 视频：集成测试概述

6.1 集成测试概述

- 1999年9月，火星气象人造卫星在经过41周4.16亿英里的成功飞行之后，在就要进入火星轨道时失败了
  - 美国投资5万美元调查事故原因，发现
    - 太空科学家使用的是英制(磅)加速度数据
    - 喷气推进实验室采用公制(牛顿)加速度数据进行计算
  - 在每个模块完成单元测试之后，需要着重考虑一个问题：通过什么方式将模块组合起来进行集成测试，这将影响模块测试用例的设计、所用测试工具的类型、测试的次序以及设计测试用例的费用和纠错的费用等

6.1 集成测试概述

- 集成测试的概念
  - 集成（Integration）
    - 把多个单元组合起来形成更大的单元
  - 集成测试（Integration Testing）
    - 也称为组装测试、联合测试、子系统测试、部件测试
    - 在假定各个软件单元已经通过了单元测试的前提下，检查各个软件单元接口之间的协同工作（参数、全局数据结构、文件、数据库）是否正确
    - 通常采用黑盒测试用例设计方法、灰盒测试方法

6.1 集成测试概述

- 集成测试主要关注下列问题
  - 模块间的数据传递是否正确？（参数个数、类型、次序）
  - 一个模块的功能是否会对另一个模块的功能产生错误的影响？（资源内容及操作）
  - 全局数据结构是否有问题，会不会被异常修改？（使用一致性）
  - 集成后各个模块的累积误差是否会扩大，是否达到不可接受的程度？（精度缺失、逻辑错误）
  - 模块组合起来的功能能否满足要求？（需求覆盖）

6.1 集成测试概述

- 模块分析
  - 集成测试的第一步，也是最重要的工作之一
  - 软件工程有一条事实上的原则，即2/8原则，即测试中发现的80%的错误可能源于20%的模块。例如，IBM OS/370操作系统中，用户发现的47%的错误源于4%的模块
  - 一般将模块划分为3个等级
    - 高危模块、一般模块和低危模块
  - 高危模块应该优先测试

6.1 集成测试概述

- 模块的划分常常遵循下列几个原则
  - 本次测试希望测试哪个模块
  - 把与该模块最紧密的模块聚集在一起
  - 在考虑划分后的外围模块，并分析外围模块和被集成模块之间的信息流是否容易模拟和控制

6.1 集成测试概述

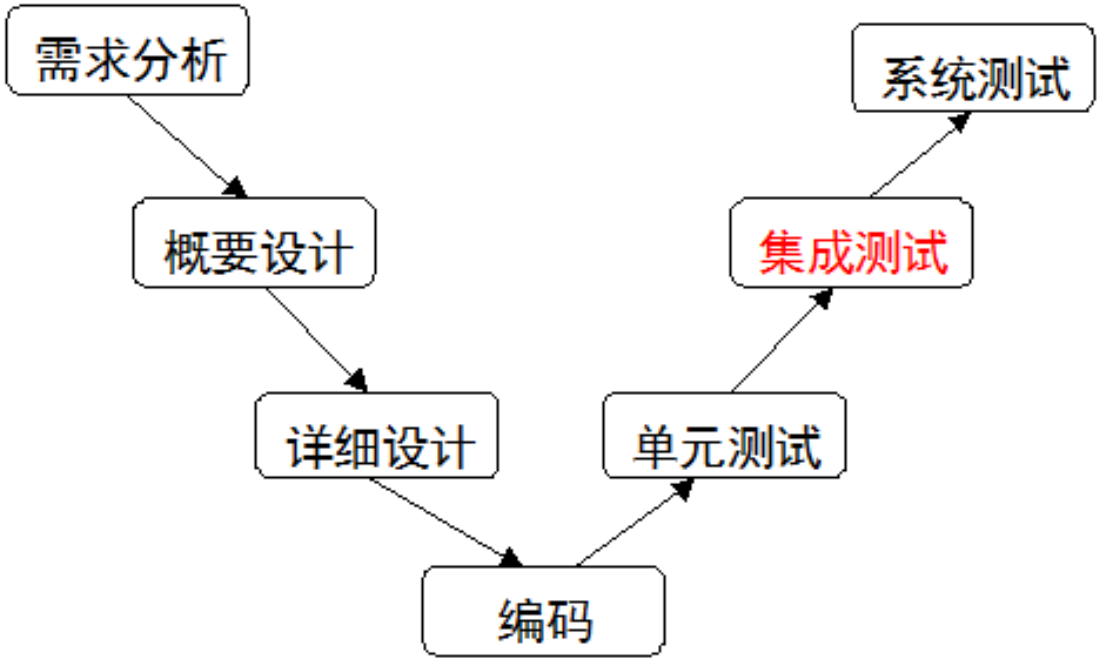


图6.1软件开发的V模型

6.1 集成测试概述

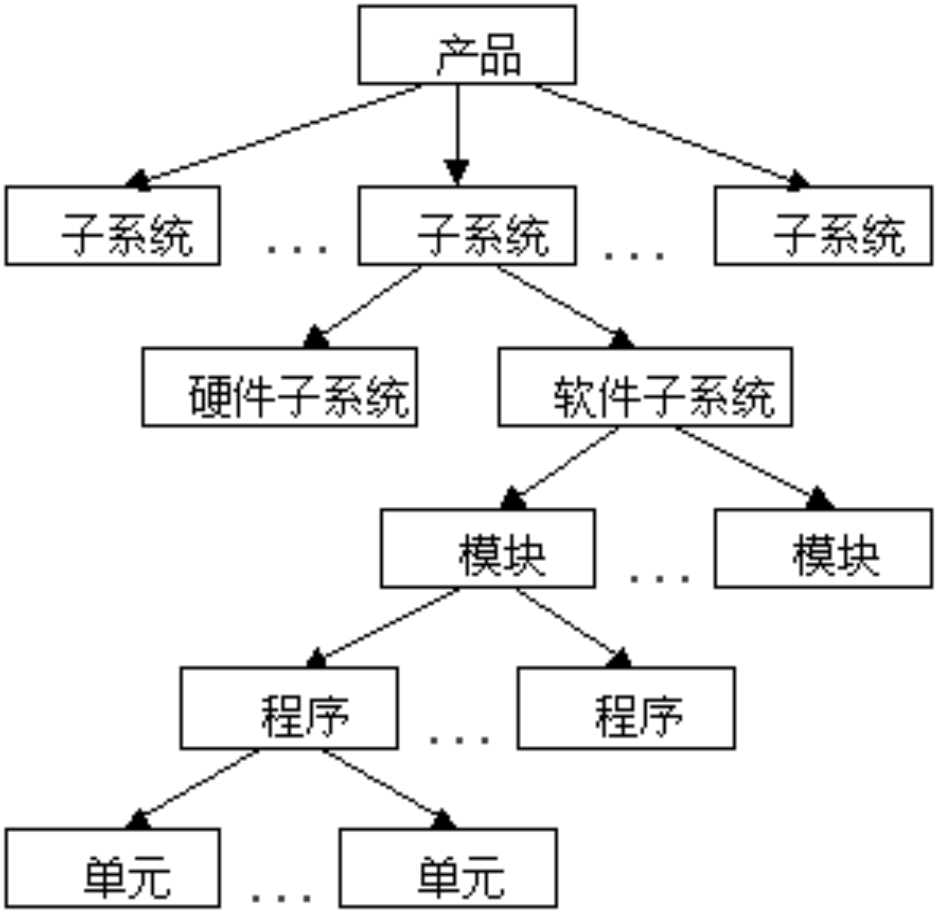


图6.2 软件模块结构图

6.1 集成测试概述

- 6.1.2 集成测试与系统测试的区别

测试对象	**单元**	**系统**
测试时间	**开发过程**	**开发完成**
测试方法	**黑白结合**	**黑盒**
测试内容	**接口**	**需求**
测试目的	**接口错误**	**需求不一致**
测试角度	**开发者**	**用户**

6.1 集成测试概述

- 6.1.3 集成测试与开发的关系
  - 集成测试与软件开发过程中的概要设计阶段相对应，软件概要设计中关于整个系统的体系结构是集成测试用例输入的基础
    - 概要设计作为软件设计的骨架，从一个成熟的体系结构中，可以清晰地看出大型系统中的组件或子系统的层次构造
    - 为集成测试策略的选取提供了重要的参考依据，从而可以减少集成测试过程中桩模块和驱动模块开发的工作量，促使集成测试快速、高质量的完成

6.1 集成测试概述

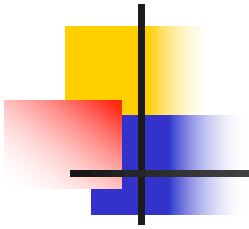
- 6.1.4 集成测试的层次与原则
  - 集成测试的层次

    传统软件按集成粒度不同，可以分为4个层次
    - 模块内集成测试
    - 模块间集成测试
    - 子系统内集成测试
    - 子系统间集成测试

    面向对象的应用系统，按集成粒度不同可分为2个层次
    - 类内集成测试
    - 类间集成测试

6.1 集成测试概述

- 6.1.4 集成测试的层次与原则
  - 集成测试的原则
    - 所有公共接口必须被测试到
    - 关键模块必须进行充分测试
    - 集成测试应当按一定层次进行
    - 集成测试策略选择应当综合考虑质量、成本和进度三者的关系
    - 集成测试应当尽早开始，并以文档为基础
    - 在模块和接口的划分上，测试人员应该和开发人员进行充分沟通
    - 当测试计划中的结束标准满足时，集成测试才能结束
    - 当接口发生修改时，涉及到的相关接口都必须进行回归测试
    - 集成测试应根据集成测试计划和方案进行，不能随意测试
    - 项目管理者应保证测试用例经过审核
    - 测试执行结果应当如实的记录

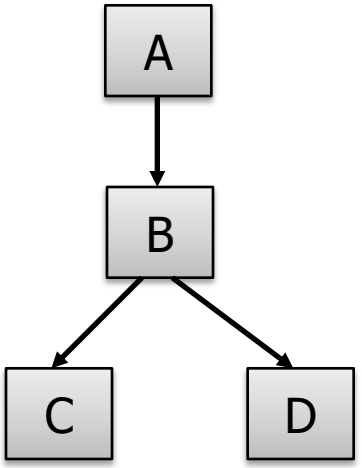


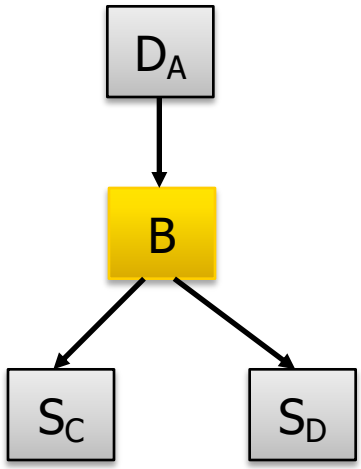
6.2 集成测试策略

- 模块组装方法
  - 非渐增式集成
    - 先分别测试每个模块，再把所有模块按设计要求放在一起结合成所要的程序
  - 渐增式集成
    - 把下一个要测试的模块同已经测试好的模块结合起来进行测试，然后再把下一个待测试的模块结合起来进行测试
    - 同时完成单元测试和集成测试

6.2 集成测试策略

- 辅助模块
  - 驱动模块（Driver）
    - 用以模拟待测模块的上级模块；接受测试数据，并传送给待测模块，启动待测模块，并打印出相应的结果
  - 桩模块（Stub）
    - 也称存根程序。用以模拟待测模块工作过程中所调用的模块。桩模块由待测模块调用，它们一般只进行很少的数据处理，例如打印入口和返回，以便于检验待测模块与其下级模块的接口





6.2 集成测试策略

- 6.2.1 非渐增式集成
  - 首先对每个子模块进行测试（即单元测试），然后将所有模块全部集成起来一次性进行集成测试
  - 举例：

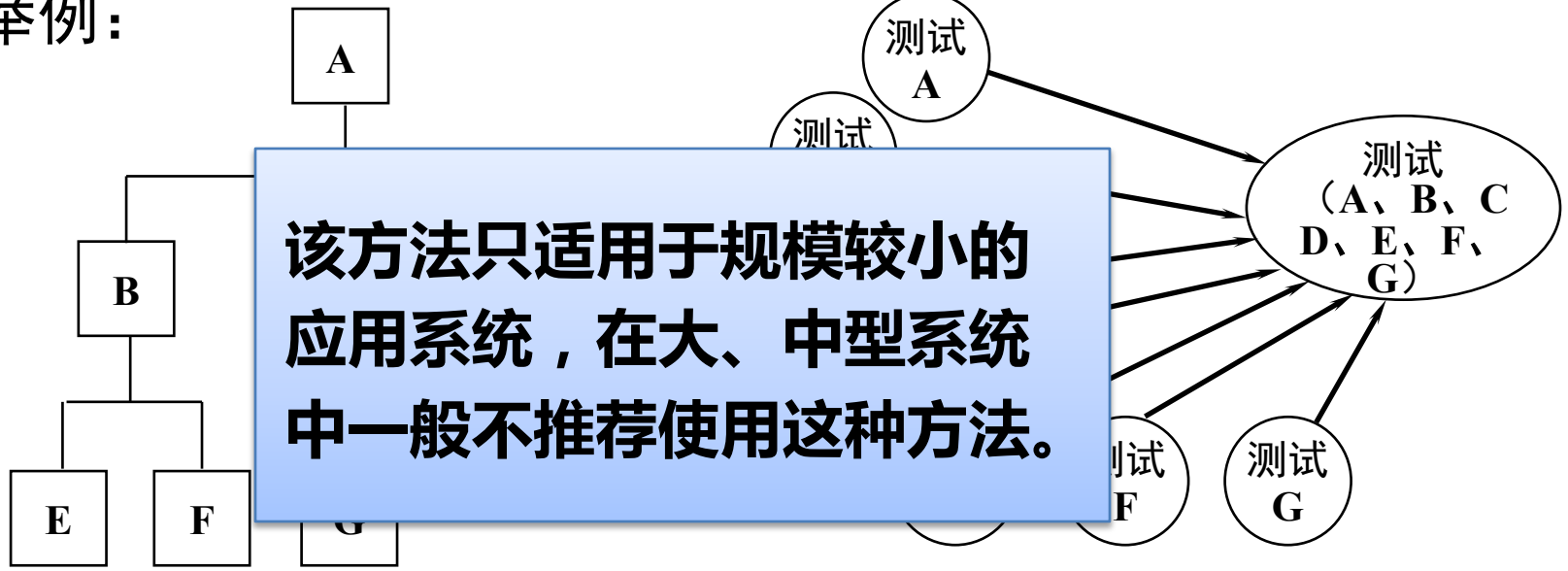
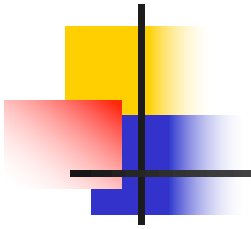


图6. 3 程序结构图

图6.4 非渐增式集成

6.2 集成测试策略

- 6.2.2 渐增式集成
  - 与“一步到位”的非渐增式集成相反，它把程序划分成小段来构造和测试
  - 容易定位和改正错误；对接口可以进行更彻底的测试；可以使用系统化的测试方法。目前在进行集成测试时普遍采用渐增式集成方法
  - 渐增方式有自顶向下、自底向上和三明治集成三种集成策略



6.2 集成测试策略

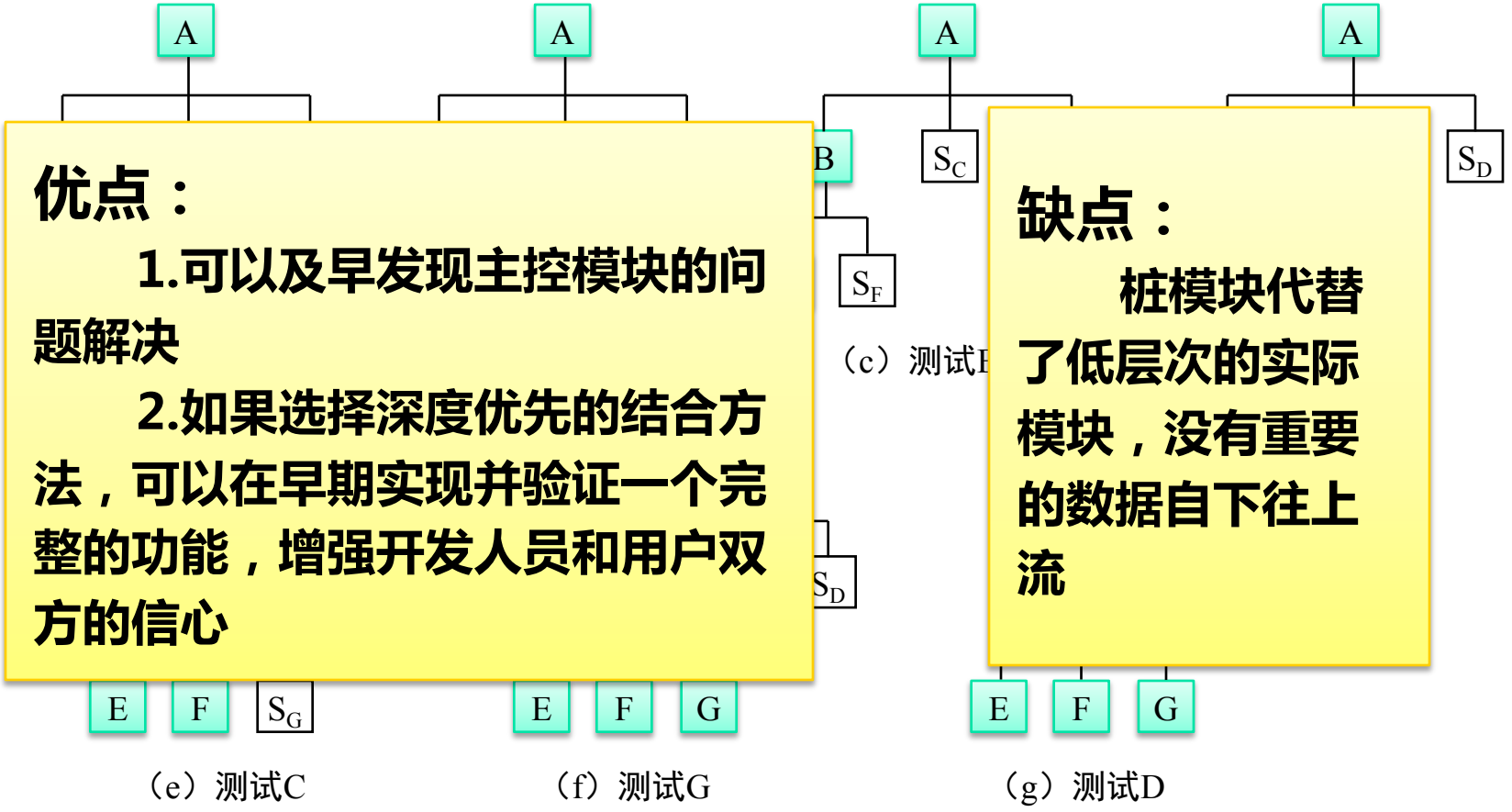
- 1．自顶向下集成
  - 从主控制模块开始，沿着程序的控制层次向下移动，逐渐把各个模块结合起来。在把附属于（及最终附属于）主控制模块的那些模块组装到程序结构中去
  - 使用深度优先的策略，或者使用宽度优先的策略

6.2 集成测试策略

- 自顶向下集成测试的步骤
  - 对主控制模块进行测试，测试时用桩模块代替所有直接附属于主控制模块的模块
  - 根据选定的结合策略（深度优先或宽度优先），每次用一个实际模块代换一个桩模块（新结合进来的模块往往又需要新的桩模块）
  - 为了保证加入模块没有引进新的错误，可能需要进行回归测试（即，全部或部分地重复以前做过的测试）

6.2 集成测试策略

- 1. 自顶向下集成



（c）测试E

（e）测试C　　（f）测试G　　（g）测试D

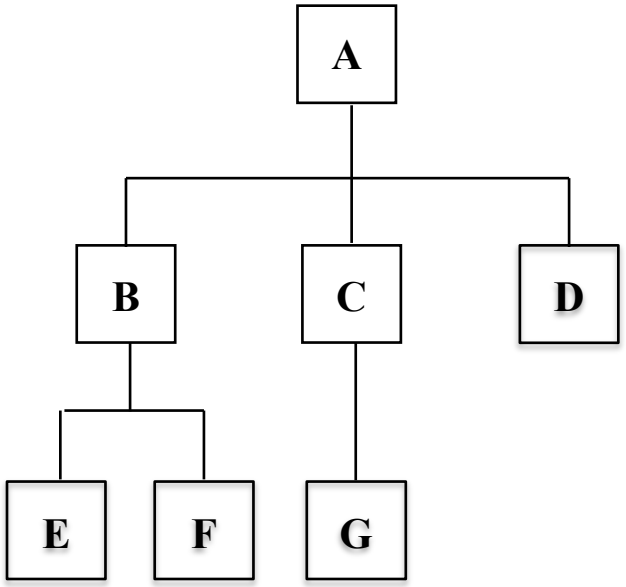
图6.5 自顶向下集成

6.2 集成测试策略

- 2．自底向上集成
  - 从“原子”模块（即最底层的模块）开始组装和测试
  - 不需要桩模块
  - 步骤
    - 把低层模块组合成实现某个特定的软件子功能的族
    - 写一个驱动程序（用于测试的控制程序），协调测试数据的输入和输出
    - 对由模块组成的子功能族进行测试
    - 去掉驱动程序，沿软件结构自下向上移动，把子功能族组合起来形成更大的子功能组

6.2 集成测试策略

- 2．自底向上集成



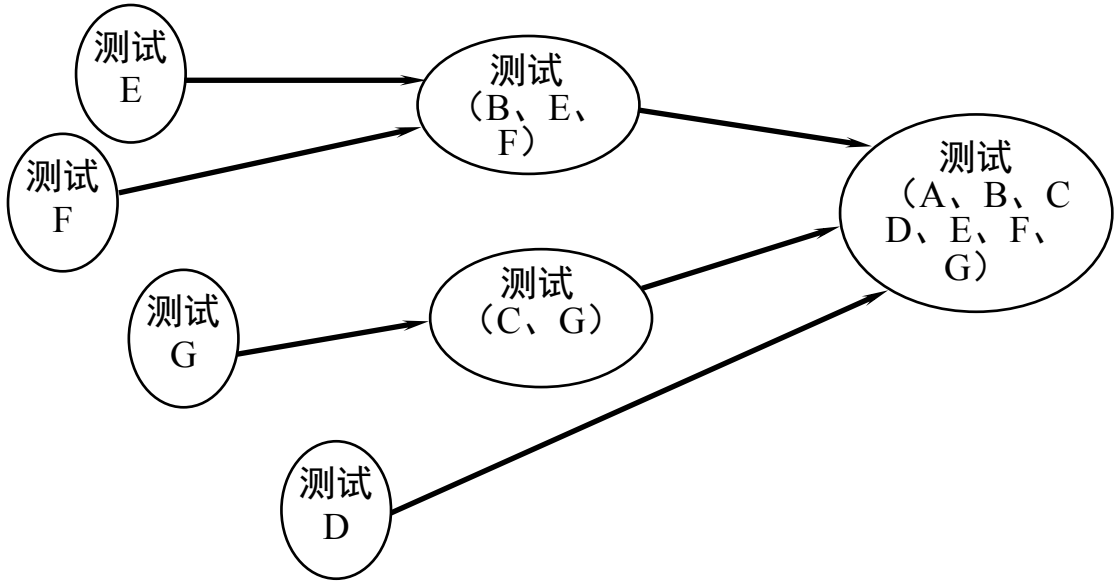


图6.6 自底向上集成

6.2 集成测试策略

- 6.2.3 三明治集成
  - 混合增量式测试策略，综合了自顶向下和自底向上两种集成方法的优点
  - 桩模块和驱动模块的开发工作都比较小
  - 代价：一定程度上增加了定位缺陷的难度

6.2 集成测试策略

- 6.2.3 三明治集成测试的过程
  - 确定以哪一层为界来进行集成，如图6-3中的B模块
  - 对模块B及其所在层下面的各层使用自底向上的集成策略
  - 对模块B所在层上面的层次使用自顶向下的集成策略
  - 对模块B所在层各模块同相应的下层集成
  - 对系统进行整体测试

6.2 集成测试策略

- 6.2.3 三明治集成测试的过程

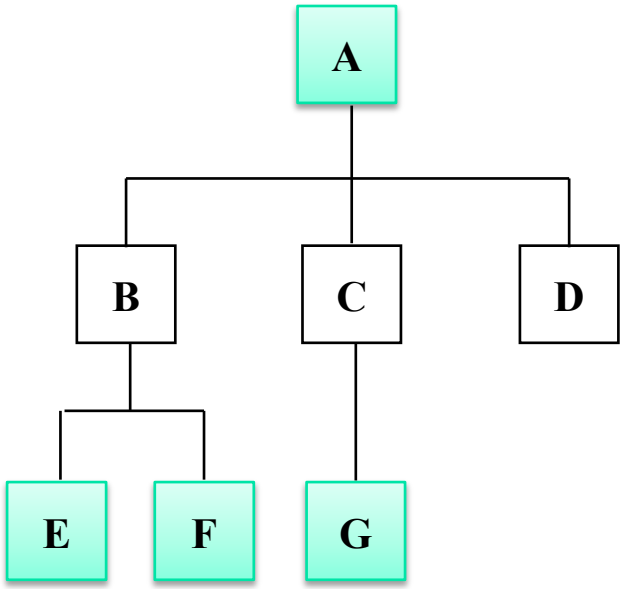


图6. 3 程序结构图

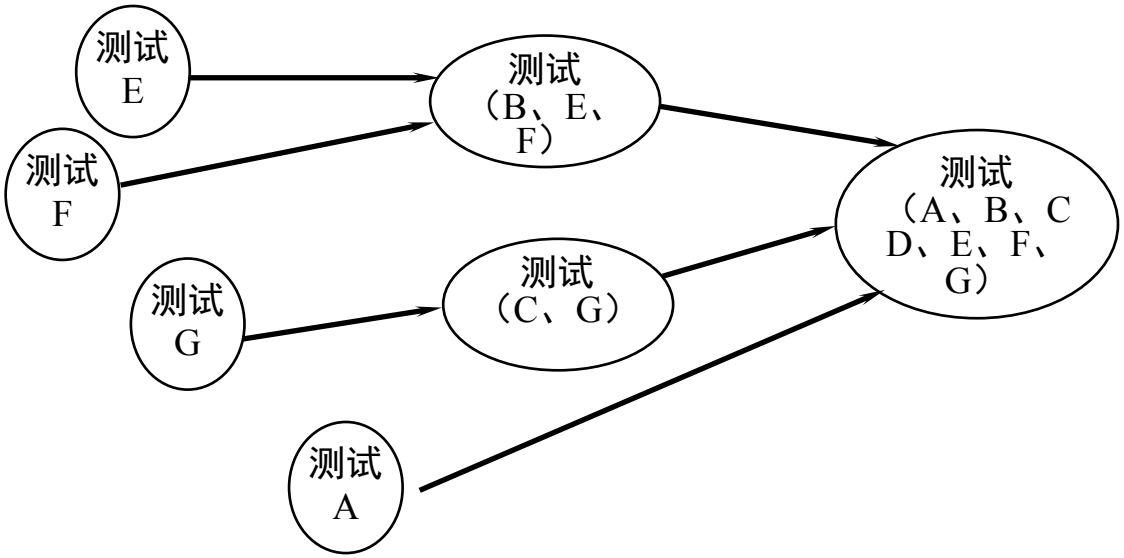


图6.7 三明治集成

6.3 集成测试用例设计

- 1．为系统运行设计用例
  - 目的：达到合适的功能覆盖率和接口覆盖率
  - 可使用的主要测试分析技术
    - 等价类划分
    - 边界值分析
    - 基于决策表的测试
    - 正交实验法
    - 状态图法

6.3 集成测试用例设计

- 2．为正向集成测试设计用例
  - 测试目标：验证集成后的模块是否按照设计实现了预期的功能
  - 可直接根据概要设计文档导出相关测试用例，使用的主要测试分析技术有
    - 输入域测试
    - 输出域测试
    - 等价类划分
    - 状态转换测试
    - 规范导出法

6.3 集成测试用例设计

- 3．为逆向集成测试设计用例
  - 测试目标：分析被测接口是否实现了需求规格没有描述的功能，检查规格说明中可能出现的接口遗漏等
  - 可使用的主要测试分析技术
    - 错误猜测法
    - 基于风险的测试
    - 基于故障的测试
    - 边界值分析
    - 特殊值测试
    - 状态转换测试

6.3 集成测试用例设计

- 4．为满足特殊需求设计用例
  - 测试目标：接口的安全性指标、性能指标等
  - 可使用的主要测试分析技术为规范导出法
- 5．为覆盖设计用例
  - 可使用的主要测试分析技术有
    - 功能覆盖分析
    - 接口覆盖分析

6.4 集成测试过程

- 集成测试可划分为5个阶段：计划阶段、设计阶段、实施阶段、执行阶段、评估阶段

6.4 集成测试过程

集成测试计划

- 1．计划阶段（制定行动指南）
  - 确定被测试对象和测试范围
  - 评估集成测试被测试对象的数量及难度，即工作量
  - 确定角色分工和划分工作任务
  - 标识出测试各个阶段的时间、任务、约束条件
  - 考虑一定的风险分析及应急计划
  - 考虑和准备集成测试需要的测试工具、测试仪器、环境等资源
  - 考虑外部技术支援的力度和深度，以及相关培训安排；定义测试完成标准

6.4 集成测试过程

- 2．设计阶段
  - 被测对象结构分析
  - 集成测试模块分析
  - 集成测试接口分析
  - 集成测试策略分析
  - 集成测试工具分析
  - 集成测试环境分析
  - 集成测试工作量估计和安排

集成测试方案

6.4 集成测试过程

- 3．实施阶段
  - 前提条件：设计阶段的评审已经通过
  - 环节
    - 集成测试用例设计
    - 集成测试规程设计
    - 集成测试代码设计
    - 集成测试脚本开发
    - 集成测试工具开发或选择

6.4 集成测试过程

- 4．执行阶段
  - 按照相应的测试规程，借助集成测试工具，并把需求规格说明书、概要设计、集成测试计划/设计/用例/代码/脚本作为测试执行的依据来执行集成测试用例
  - 测试执行的前提条件就是单元测试已经通过评审
  - 测试执行结束后，测试人员要记录下每个测试用例执行后的结果，填写集成测试报告，最后提交给相关人员评审

6.4 集成测试过程

- 5．评估阶段
  - 集成测试执行结束后，要召集相关人员（测试设计人员、编码人员、系统设计人员等）对测试结果进行评估，确定是否通过集成测试

谢谢！