集成测试

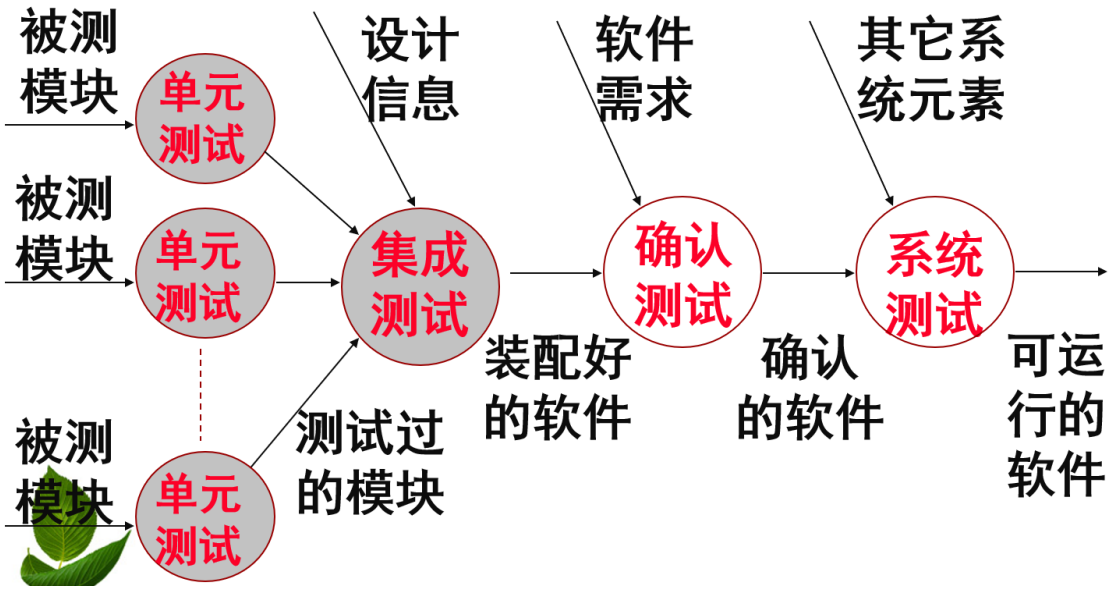
1、单元测试后问题出现了——软件在系统集成时会经常有这样的情况发生：

（1）每个模块都能单独工作，但这些模块集成在一起之后却不能正常工作

（2）系统集成后虽可以正常运行，但系统的容错性、安全性以及整体性却得不到保障，系统不能长时间运行等等。

这就需要进行集成测试和系统测试，以找出其中的软件缺陷，来提高整个软件的质量和可靠性。

2、集成测试又称组装测试，是在单元测试的基础上，将所有模块按照设计要求组装成子系统或系统进行的测试活动。又称子系统测试、联合测试



3、单元测试、集成测试与系统测试的差别

	对象	目的	测试依据	测试方法
单元测试	模块内部程序错误	消除局部模块逻辑和功能上的错误和缺陷	模块逻辑设计 模块外部说明	大量采用白盒测试方法
集成测试	模块间的集成和调用关系	找出与软件设计相关的程序结构，模块调用关系，模块间接口方面的问题	程序结构设计	灰盒测试，采用较多黑盒方法构造测试用例
系统测试	整个系统，包括系统软硬件等	对整个系统进行一系列的整体、有效性测试	系统结构设计 目标说明书 需求说明书等	黑盒测试

9

4、集成测试的目的：确保各单元组合在一起后能够按既定意图协作运行，并确保增量的行为正确，所测试的内容包括单元间的接口以及集成后的功能

具体来说，集成测试考虑以下问题：

（1）接口问题：在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失

（2）功能问题：各个子功能组合起来，能否达到预期要求的父功能

（3）模块间问题：一个模块的功能是否会对另一模块的功能产生不利影响

（4）数据结构问题：全局数据结构是否有问题

（5）误差积累问题：单个模块的误差积累起来，是否会放大，从而达到不可接受的程度

5、集成测试的特点

同单元测试相比：单元测试具有不彻底性，对于模块间接口信息内容

的正确性、相互调用关系是否符合设计无能为力。只能靠集成测试来进行保障。

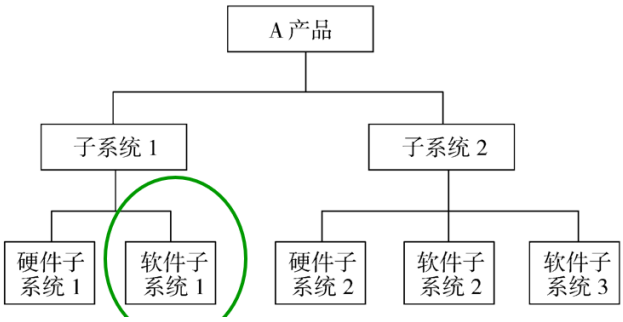
同系统测试相比：由于集成测试用例是从程序结构出发的，目的性、针对性更强，测试项发现问题的效率更高，定位问题的效率也更高

集成测试能够较容易地测试到系统测试用例难以模拟的特殊异常流程，从纯理论的角度来讲，集成测试能够模拟所有实际情况

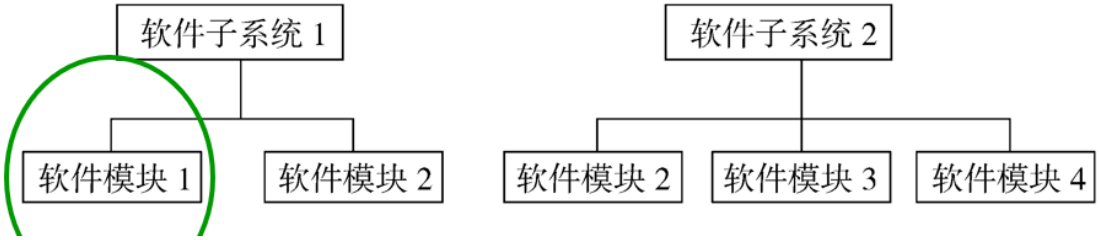
定位问题较快，由于集成测试具有可重复性强，对测试人员透明的特点，发现问题后容易定位，所以能够有效地较快进度，减少隐患

6、集成测试的层次

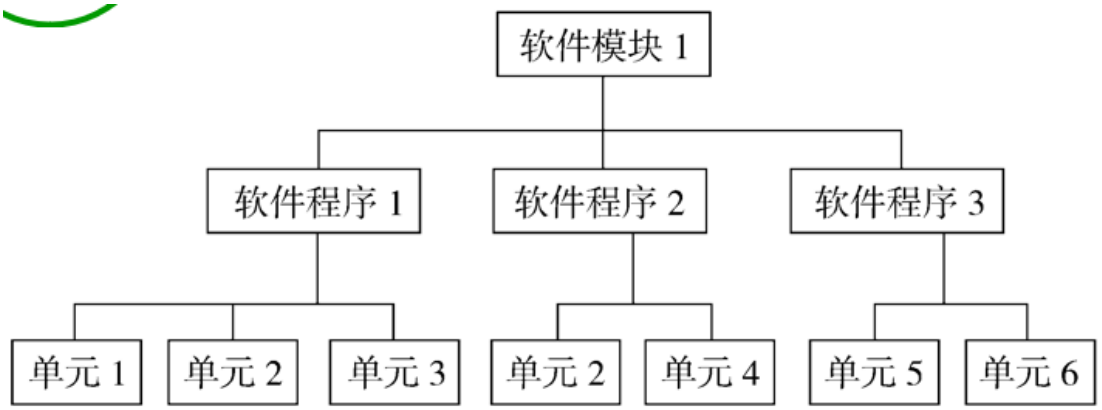
系统结构图



软件结构图



软件模块结构图



7、集成测试的级别

由于集成的力度(粒度？)不同，一般可以把集成测试划分为三个级别：

（1）模块内集成测试

（2）子系统内集成测试：先测试子系统内的功能模块，然后将各个功能模块组合起来确认子系统的功能是否达到预期要求

（3）子系统间集成测试：测试的单元是子系统之间的接口，子系统是可单独运行的程序或进程

8、集成测试的方法

（1）静态测试技术——针对概要设计的测试，概要设计是设计师根据用户交互过程和用户需求来形成交互框架和视觉框架的过程. 概要设计的主要任务是把需求分析得到的系统扩展用例图转化为软件结构和数据结构

（2）动态测试技术——灰盒测试，黑盒测试与白盒测试相结合

灰盒测试的优点：

能够进行基于需求的测试和基于路径的覆盖测试

可深入被测对象的内部，便于错误的是别分析和解决

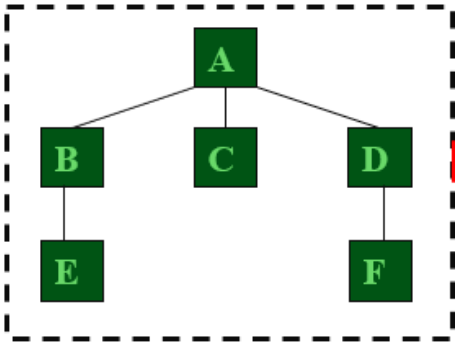
能够保证设计的黑盒测试用例的完整性，防止功能或功能组合的遗漏

能够减小需求或设计不详细或不完整性对测试有效性造成影响

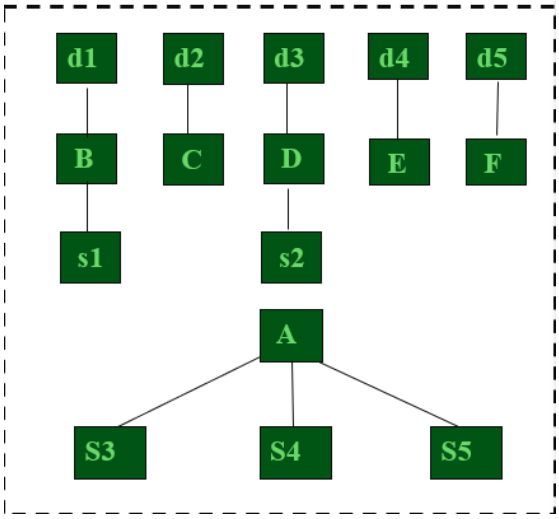
9、集成测试的策略

集成策略指在测试对象分析基础上，描述软件模块集成的方式方法。我们已经知道，集成测试是把功能模块或程序单元组合起来进行测试，发现模块在组合过程中的缺陷。那么，系统中的各个模块如何组合？是全部同时组装还是逐渐组装模块？这是集成策略将要解答的问题。

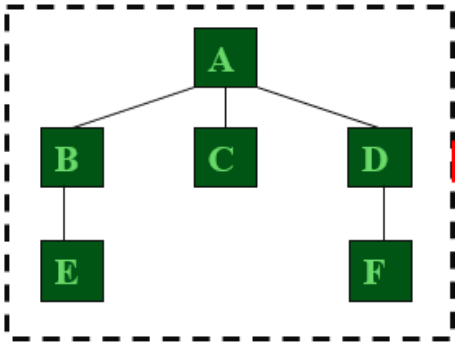
（1）非增量式集成策略(Non-Incremental Testing)——采用一步到位的方法来构造测试，对所有模块进行个别的单元测试后，按照程序结构图将各模块连接起来，把连接后的程序当作一个整体进行测试，又叫大爆炸式集成(Big Bang)



i. 程序结构图



ii. 单元测试示意图



iii. 集成测试示意图

模块d_1, d_2, d_3, d_4, d_5是对各个模块做单元测试时建立的**驱动模块**

s_1, s_2, s_3, s_4, s_5是为单元测试而建立的**桩模块**

评述：这种一次性集成方式 将所测模块连接起来进行测试，但是一次是运行成功地可能性并不大。其结果发现有错误，但茫然找不到原因，查错和改错都会遇到困难。适用于一个维护性或被测试系统较小的项目

非增量式集成策略的优点：① 方法简单 ② 允许多测试人员同时并行工作，人力物力资源利用率较高

非增量式集成策略的缺点：① 必须为每个模块准备相应的驱动模块和桩模块，测试成本较高 ② 一旦集成后包含多种错误，难以纠正

在非增量式集成测试时，应当确定关键模块，对这些关键模块及早进行测试。关键模块的特征：

① 完成需求规格说明中的关键功能

② 在程序的模块结构中位于较高的层次(高层控制模块)

③ 较复杂、较易发生错误

④ 有明确定义的性能要求

(2) 增量式集成策略(Incremental Testing)——逐步实现，逐次将未曾集成测试的模块和已经集成测试的模块(或子系统)结合成程序包，再将这些模块集成为较大系统，在继承过程中边连接边测试，以发现连接过程中产生的问题。

按照不同的实施次序，增量式集成测试又可以分为三种不同的方法：

(1) **自顶向下增量式测试**

自顶向下增量式测试表示逐步集成和逐步测试是按照结构图自上而下进行的，即模块集成的顺序是首先集成主控模块(主程序)，然后依照控制层次结构向下进行集成。从属于主控模块的按深度优先方式(纵向)或者广度优先方式(横向)集成到结构中去。

深度优先方式：首先集成在结构中的一个主控路径下的所有

模块，主控路径的选择是任意的

广度优先方式：首先沿着水平方向，把每一层中所有直接隶属于上一层的模块集成起来，直到底层。

自顶向下集成测试的整个过程由 3 个步骤完成：

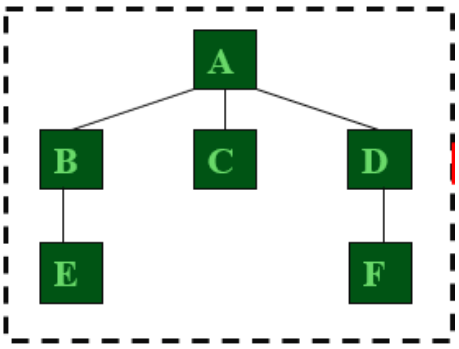
i. 主控模块作为测试驱动器

ii. 根据集成的方式(深度或广度)，下层的桩模块一次一次地被替换为真正的模块

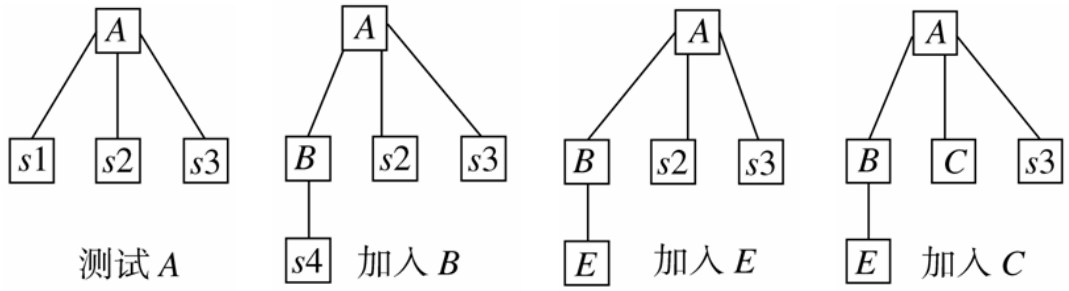
iii. 在每个模块被集成时，都必须进行单元测试

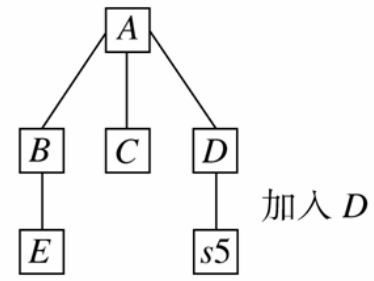
重复第 2 步，直到整个系统被测试完成

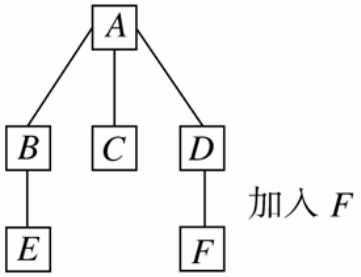
例如：



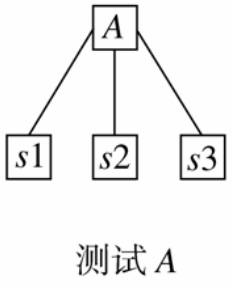
深度优先组装方式如下：

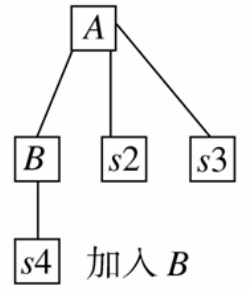


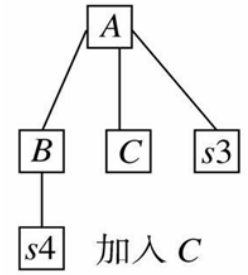


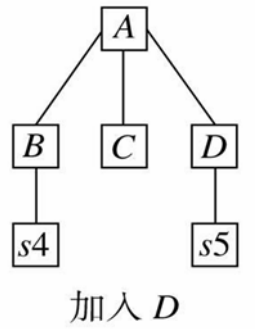


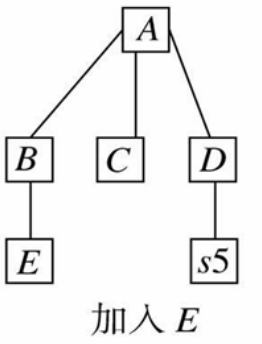
广度优先组装方式如下：

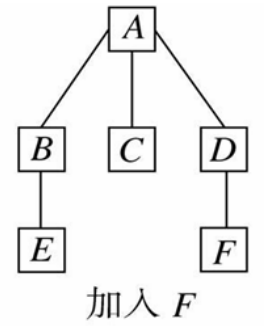












优点：较早地验证了主要控制和判断点

按深度优先可以首先实现和验证一个完整的软件功能

功能较早证实，减少驱动器开发的费用

只需一个驱动，减少驱动器开发的费用

支持故障隔离

缺点：桩的开发量大

底层验证被推迟

底层组件测试不充分

适用范围：产品控制结构比较清晰和稳定

高层接口变化较小

底层接口未定义或经常可能被修改

产口控制组件具有较大的技术风险，需要尽早被验证

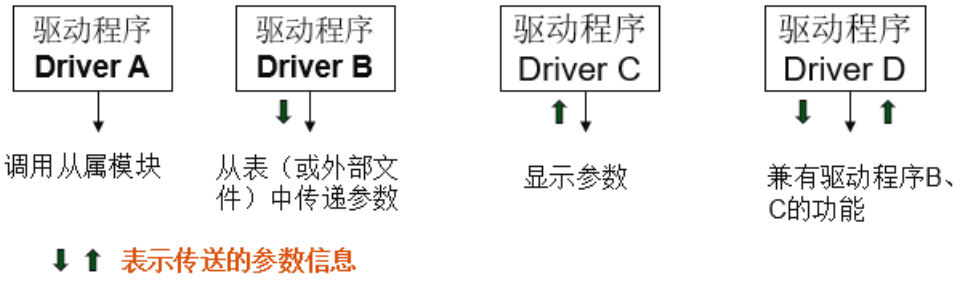
希望尽早能看到产品的系统功能行为

（2）自底向上增量式测试

从具有最小依赖性的底层组件开始，按照依赖关系树的计够，逐层向上集成，以检验系统的稳定性。

最常用的集成策略，其他方法都或多或少应用此种方法。

自底向上进行集成和测试时，需要为所测试模块或子系统编制相应的驱动模块。常用的几种类型的驱动模块如图所示：



自底向上增量式集成测试步骤

（1）起始于模块依赖关系树的底层叶子模块，也可以把两个或多个叶子模块合并到一起进行测试

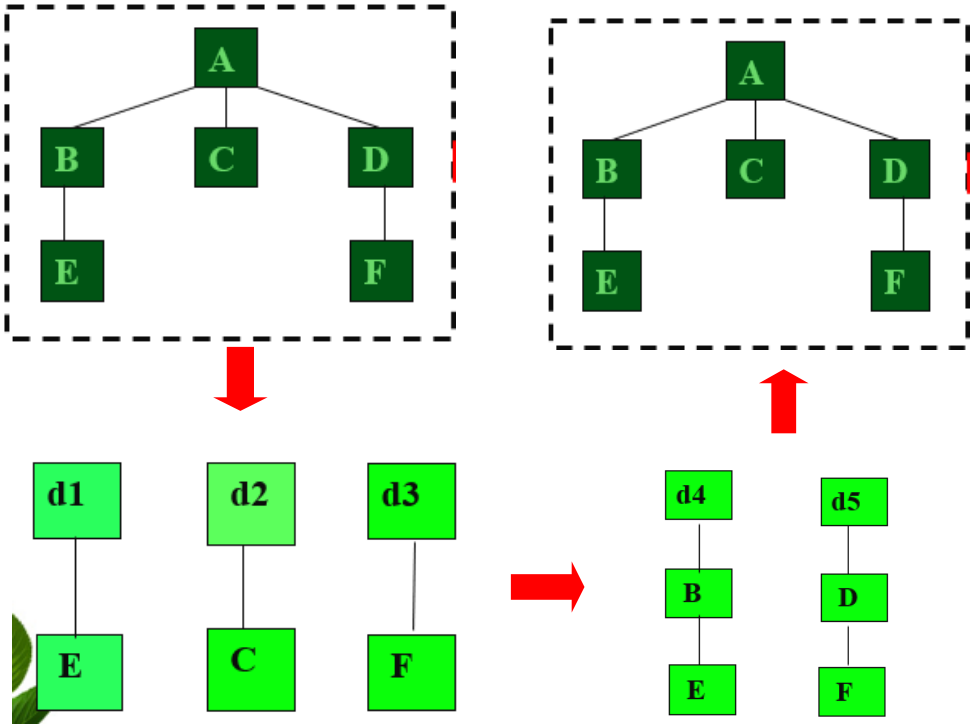
（2）使用驱动模块对步骤 1 选定的模块(或模块组)进行测试

（3）用实际模块代替驱动模块，与它已测试的直属子模块组装成一个更大的模块进行测试

（4）重复上面的行为，直到系统最顶层模块被加入到已测系

统中

例如：



优点：对底层组件行为较早验证

工作最初可以并行继承，比自顶向下效率高

减少了桩的工作量

能较好锁定软件故障所在位置

缺点：驱动的开发工作量大

对高层的验证被推迟，设计上的错误不能被及时发现

适用范围：适应于底层接口比较稳定

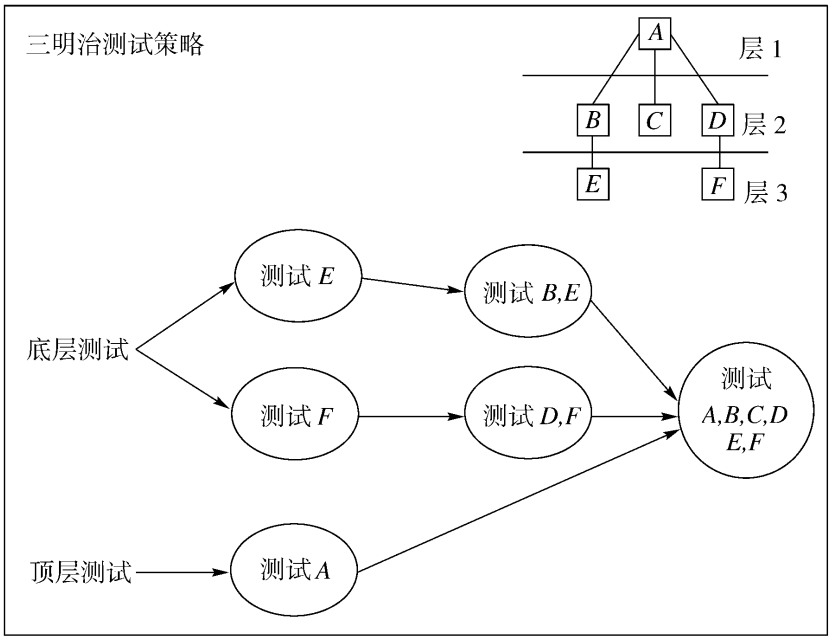
高层接口变化比较频繁

底层组件较早被完成

（3）**三明治增量式测试(混合增量式测试)**

把系统化分成三层，中间一层为目标层，目标层之上采用自

顶向下集成，之下采用自底向上集成



集成步骤：

（1）首先对目标层之上一层使用自顶向下集成，因此测试 A，使用桩代替 B，C，D

（2）其次对目标层之下一层使用自底向上集成，因此测试 E，F，使用驱动代替 B，D

（3）其三，把目标层下面一层与目标层集成，因此测试（B，E），（D，F），使用驱动代替 A

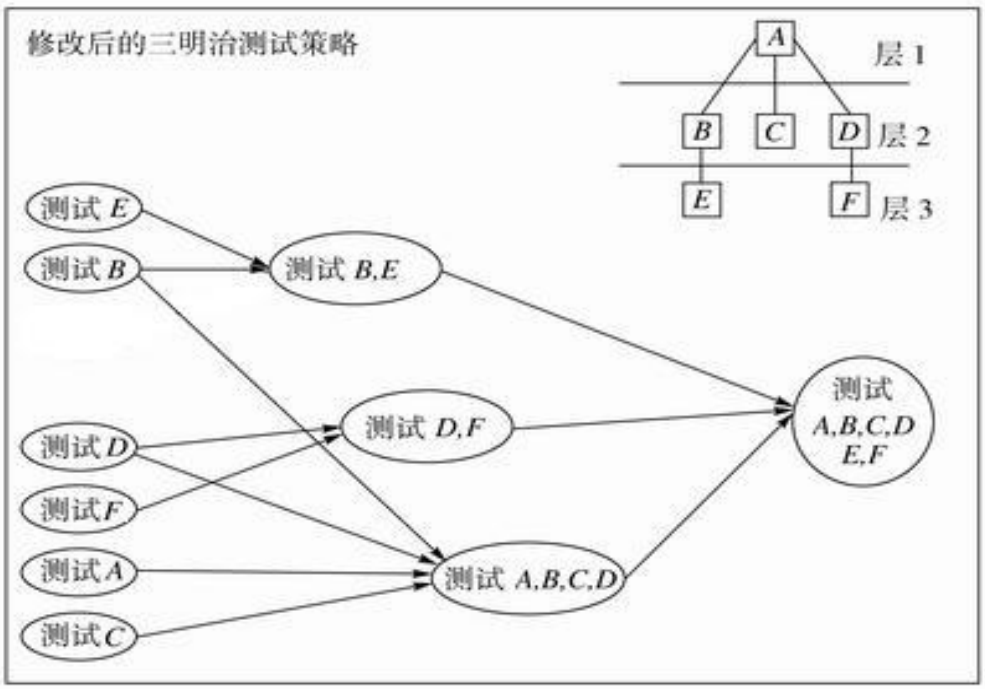
（4）最后，把三层集成到一起，因此测试（A,B,C,D,E,F）

优点：集合了自顶向下和自底向上两种策略的优点

缺点：中间层测试不充分

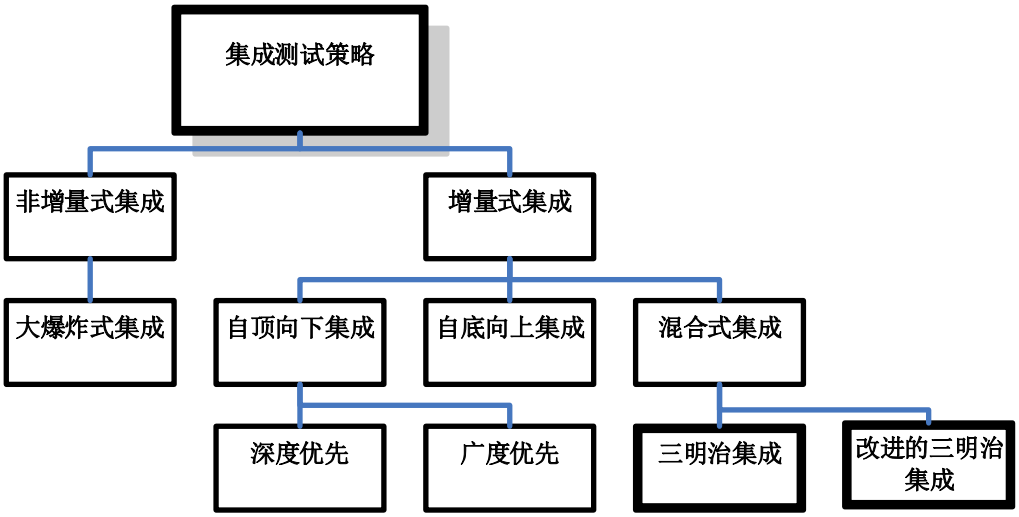
适用范围：适应于大部分软件开发项目

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 。



10、不同集成测试方法的比较

综上所述，可得到集成策略框图如下：



（1）非增量式测试与增量式测试的比较

非增量式测试的方法是先分散测试，然后集中起来再一次完成集成测试。加入在模块的接口处存在错误，只会在最后的集成测试时一下子暴露出来

增量式测试是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。而且一些模块在逐步集成的测试中，得到了较多次的考验，因此，可能会取得较好的测试效果。

结论：增量式测试要比非增量式测试具有一定的优越性。

（2）自顶向下与自底向上增量式测试的比较

自顶向下增量式测试：主要优点在于它可以自然的做到逐步求精，一开始就能让测试者看到系统的框架。主要缺点是需要提供桩模块，并且在输入/输出模块接入系统以前，在桩模块中表示测试数据有一定困难。

自底向上增量式测试：主要优点在于由于驱动模块模拟了所有调用参数，即使数据流并未构成邮箱的非环状图，生成测试数据也无困难。主要缺点在于直到最后一个模块被加进去之后才能看到整个程序(系统)的框架

3种增量测试策略比较

	自顶向下	自底向上	混合式
形成可工作程序所需时间	早	晚	早
是否需要驱动器	否	是	是
是否需要程序桩	是	否	是
集成开始时可否平行工作	低	中等	中等
测试特殊路径的能力	难	易	中等
计划和控制顺序的能力	难	易	难

集成测试方法选择指南

因素	建议使用集成方法
设计和需求清晰	自顶向下
需求、设计和体系结构不断变化	自底向上
体系结构改变、设计稳定	混合
体系结构较小，规模小	瞬时集成
体系结构复杂，软件规模大	混合

11、其他集成测试策略

（1）核心系统测试

核心系统先行集成测试法的思想是先对核心软件部件进行集成测试，在测试通过的基础上再按各外围软件部件的重要程度逐个集成到核心系统中。每次加入一个外围软件部件都产生一个产品基线，直至最后形成稳定的软件产品。核心系统先行集成测试法对应的集成过程是一个逐渐趋于闭合的螺旋形曲线，代表产品逐步定型的过程。

优点：该集成测试方法对于快速软件开发很有效果，适合较复杂系统的集成测试，能保证一些重要的功能和服务的实现。

缺点：采用此法的系统一般应能明确区分核心软件部件和外围软件部件，核心软件部件应具有较高的耦合度，外围软件部件内部也应具有较高的耦合度，但各外围软件部件之间应具有较低的耦合度。

（2）高频集成测试

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。如某些自动化集成测试工具能实现每日深夜对开发团队的现有代码进行一次集成测试，然后将测试结果发到各开发人员的电子邮箱中。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。

优点：该测试方案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：在于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）基于功能的集成测试

基于功能的集成测是从功能角度出发，按照功能的关键程度对模块的集成顺序进行组织，尽可能早点进行测试。其关注于测试验证系统的关键功能。

此方法适应于对较大风险的产品、技术探索的项目或者是对功能实现没有把握的产品，其功能的实现比质量要关键。

（4）基于进度的集成测试

基于进度的集成测试主要是从系统的进度和质量两方面的考虑，尽早地进行集成测试，以提高开发和集成的并行性，有效地缩短项目的开发时间，提高开发项目的质量。

但此方法早期测试缺乏整体性，仅能进行独立的集成，导致许多接口要到后期才能验证。桩模块与驱动模块的开发量大，模块可能不稳定产生变化，导致测试的重复与浪费。

（5）基于风险的集成测试

基于风险的集成测试是基于一种假设的方法，系统的错误往往集中在系统风险最高的模块中，因此对高风险的模块接口先进行重点测试，从而保证系统的稳定性。

尽早验证高风险的接口有助于加速系统的稳定性，有利加强对系统的信心。此方法可以与功能集成测试结合起来使用。主要适应于系统中风险较大的模块测试。

（6）客户/服务器的集成测试

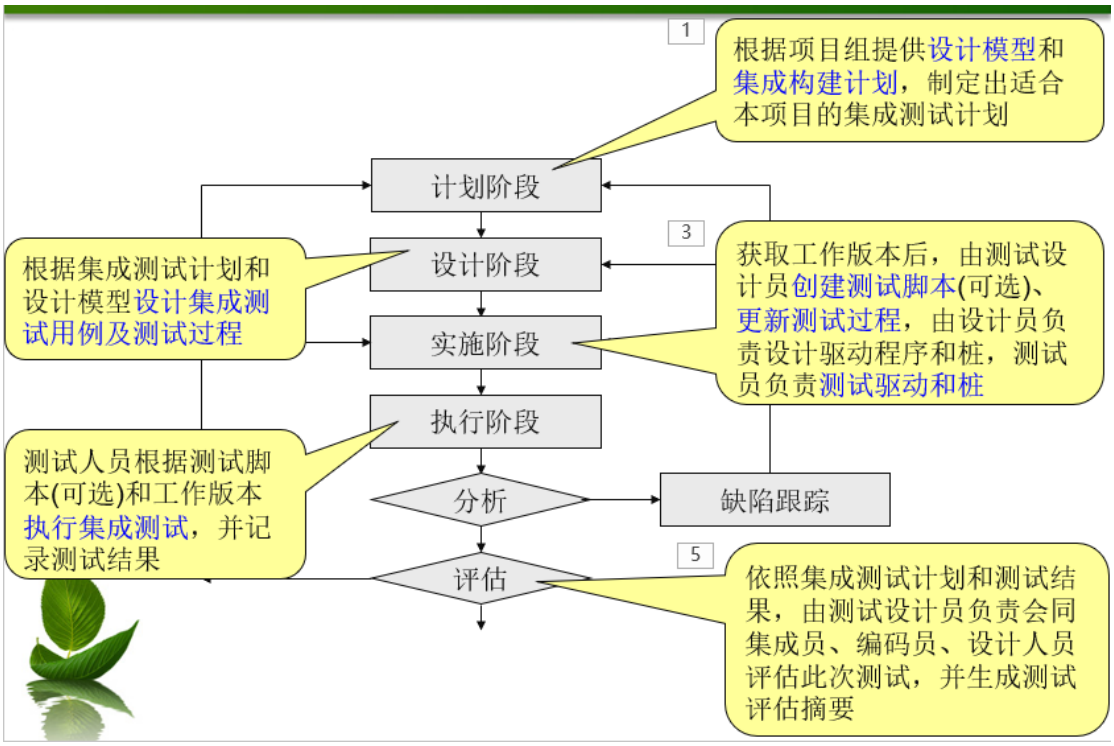
客户/服务器的集成测试主要针对客户/服务器系统，对系统客户端与服务器端交互进行集成测试。先单独测试每个客户端和服务器端，再将第一个客户端与服务端集成测试，加入下一个客户端与服务端集成测试，如此下去，将所有客户端与服务器端集成测完成。

此方法对集成次序没有约束，有利于复用和扩充；支持可控制和可重复的测试。但驱动模块与桩模块的开发成本高。

12、集成测试的流程

集成测试主要由系统部的系统设计人员、软件评测部完成，开发人员也参与集成测试。集成测试相对来说是挺复杂的，而且对于不同的技术、平台和应用差异也比较大，更多是和开发环境融合在一起。集成测试所确定的测试的内容，主要来源于设计模型。

过程	工作内容	工作结果	担当人员和职责
制定集成测试计划	设计模型 集成构建计划	集成测试计划	测试设计员负责制定集成测试计划
设计集成测试	集成测试计划 设计模型	集成测试用例 测试过程	测试设计员负责设计集成测试用例和测试过程。
实施集成测试	集成测试用例 测试过程 工作版本	测试脚本（可选） 测试过程（更新）	测试设计员负责编制测试脚本（可选），更新测试过程。
		驱动程序或稳定桩	设计员负责设计驱动程序和桩，实施员负责实施驱动程序和桩。
执行集成测试	测试脚本（可选） 工作版本	测试结果	测试员负责执行测试并记录测试结果
评估集成测试	集成测试计划 测试结果	测试评估摘要	测试设计员负责会同集成员、编码员、设计员等有关人员（具体化）评估此次测试，并生成测试评估摘要。



集成测试的计划阶段

（1）集成测试准备

在进行集成测试前，先要准备如下内容：

（a）测试的文档准备：需求规格说明书、概要设计文档、产品开发计划。

（b）人员组织：测试人员、开发人员、测试质量控制员、测试经理、开发经理、产品经理。

角色	职责
测试人员	负责测试用例设计、执行、记录、回归测试
开发人员	负责定位和解决问题，修复缺陷
测试质量控制人员	负责对集成测试进行监控、评审
测试经理	负责制定测试计划、安排测试任务
开发经理	负责代码的修复和安排管理
产品经理	负责解决资源要求（人力资源、工具等），并对测试结果监督

（2）集成测试策略和环境

集成测试采用的测试技术和工具，完成测试影响的资源分配及特殊的考虑。

其环境考虑内容如下：

（a）硬件环境：　尽可能考虑实际环境。

（b）操作系统环境：不同机型使用不同的操作系统版本。

（c）数据库环境：从性能、版本、容量等多方面考虑。

（d）网络环境

（3）测试日程计划

根据软件设计文档来评估测试有多少项目，再根据测试的工作量进行安排。开始时间在概要设计完成评审后大约一个星期。

（4）活动步骤

（a）确定被测试对象和测试范围；

（b）评估集成测试被测试对象的数量及难度，即工作量 ；

（c）确定角色分工和任务；

（d）标识出测试各阶段的时间、任务、约束等条件；

（e）考虑一定的风险分析及应急计划；

（f）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；

（g）考虑外部技术支援的力度和深度，以及相关培训安排；

（h）定义测试完成标准。

（5）输出

集成测试计划阶段最后得到集成测试计划，此计划必须通过概要设计阶段基线评审通过。

集成测试的设计阶段

（1）时间安排

集成测试的设计在系统详细设计阶段就可以开始。

（2）依据

集成测试的设计阶段的主要依据是：需求规格说明书、概要设计、集成测试计划。

（3）入口条件

系统的概要设计基线已通过评审。

（4）活动步骤

（a）被测对象结构分析；

（b）集成测试模块分析；

（c）集成测试接口分析；

（d）集成测试策略分析；

（e）集成测试工具分析；

（f）集成测试环境分析。

（5）输出

集成测试设计阶段的输出是集成测试设计方案。

（6）出口条件

集成测试设计通过详细设计基线评审。

集成测试的实施阶段

根据集成测试计划，建立集成测试环境，完成测试设计任务。

（1）时间安排：在系统编码阶段开始后就可以进行了。

（2）依据：系统的需求规格说明书、概要设计、集成测试计划、集成测试设计计划。

（3）入口条件：系统的详细设计基线通过评审。

（4）活动步骤

（a）集成测试用例设计；

（b）集成测试代码设计（如果需要）；

（c）集成测试脚本设计（如果需要）；

（d）集成测试工具准备（如果需要）；

（5）输出

最后输出有：集成测试用例、集成测试规程、集成测试代码、集成测试脚本、集成测试工具。

测试用例模板如表所示：

测试任务编号		测试任务描述		
作　者		日　期		
设计方法		测试用例脚本文件名		
预置条件				
用例编号	输入	执行动作	预期结果	备注

（6）出口条件

集成测试设计阶段的出口条件是：测试用例和测试规程通过编码阶段基线评审。

集成测试的执行阶段

按照集成测试用例设计要求进行构建平台。

（1）时间安排

系统的单元测试已经完成后就可以开始执行系统的集成测试了。

（2）输入

集成测试需要的基本内容：需求规格说明书、概要设计、集成测试计划、集成测试例、集成测试规程、集成测试代码（如果有）、集成测试脚本、集成测试工具、详细设计代码、单元测试报告。

（3）入口条件

系统的单元测试阶段已经通过基线化评审。

（4）活动步骤

（a）执行集成测试用例；

（b）回归集成测试用例；

（c）撰写集成测试报告；

（5）输出

集成测试实施后，最后生成集成测试报告。

（6）出口条件

集成测试报告通过集成测试阶段基线评审。

集成测试的评估阶段

由测试设计员负责，与集成测试人员、编码员、设计员等对集成测试结果进行统计，生成测试执行报告和缺陷记录报告。并对集成测试进行评估，测试结果进行评测，形成结论，最后整理形成报告。