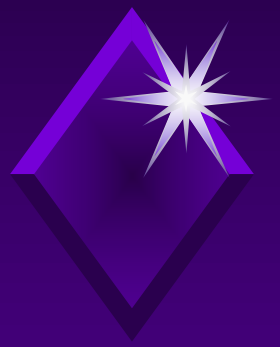
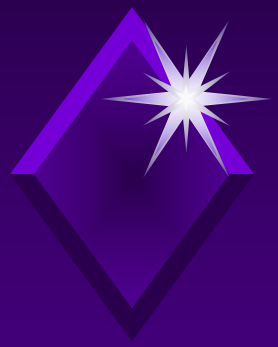
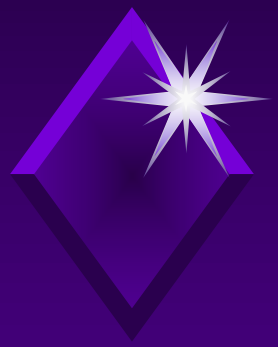


软件测试组织与管理及测试系列方法



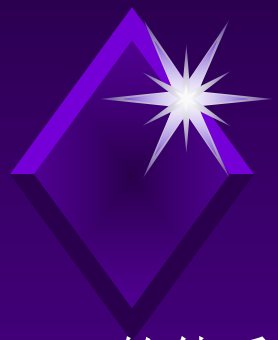
内容目录

- 软件质量概述
- 软件测试组织与管理
- 软件测试策略
- 测试步骤
- 总结



软件质量概述——Topics

- 软件测试质量概念
- 软件测试意义
- 软件测试概念
- 软件测试的重要性
- 软件测试目的



软件质量概述

软件质量是指与软件产品满足规定的和隐含的需求的能力有关的特征和特性的全体。通常来说软件质量应该包含六方面的特性：

1）功能性：软件所实现的功能达到它的设计规范和满足 用户需求的程度

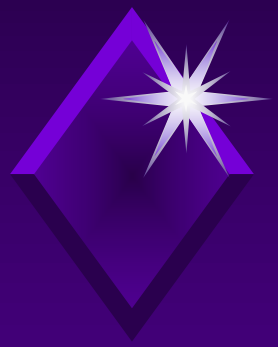
2）可靠性：在规定的时间和条件下，软件所能维持其性能水平的程度

3）易用性：对于一个软件，用户学习、操作、准备输入和理解输出所作努力的程度

4）效率：在指定条件下，用软件实现某种功能所需的计算机资源（包括时间）的有效程度

5）可维修性：在一个运行软件中，当环境改变或软件发生错误时，进行相应修改所做努力的程度

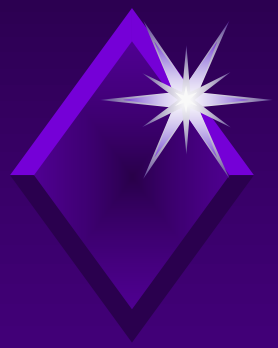
6）可移植性：软件从一个计算机系统或环境移植到另一个系统或环境的容易程度



软件质量概述

软件测试的意义：

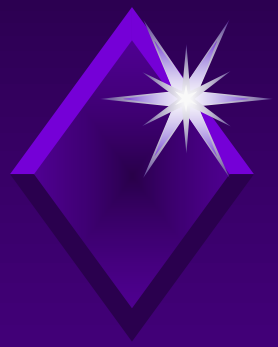
软件危机曾经是软件界甚至整个计算机界最热门的话题。为了解决这场危机，软件从业人员、专家和学者做出了大量的努力。现在人们已经逐步认识到所谓的软件危机实际上仅是一种状况，那就是软件中有错误，正是这些错误导致了软件开发在成本、进度和质量上的失控有错是软件的属性，而且是无法改变的，因为软件是由人来完成的所有由人做的工作都不会是完美无缺的。问题在于我们如何去避免错误的产生和消除已经产生的错误，使程序中的错误密度达到尽可能低的程度 。



软件质量概述

- 软件测试的概念

软件测试的定义有许多种，其中比较权威的是IEEE在1983年提出的使用人工或自动手段来运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别



软件质量概述

- 软件测试的重要性

软件测试在软件生命周期中占据重要的地位，在传统的瀑布模型中软件测试学仅处于运行维护阶段之前，是软件产品交付用户使用之前保证软件质量的重要手段。近来，软件工程界趋向于一种新的观点，即认为软件生命周期每一阶段中都应包含测试，从而检验本阶段的成果是否接近预期的目标，尽可能早的发现错误并加以修正如果不在早期阶段进行测试，错误的延时扩散常常会导致最后成品测试的巨大困难 。

软件质量概述

- 软件测试的重要性：

软件中的错误密度也需要测试来进行估计测试是所有工程学科的基本组成单元，是软件开发的重要部分。自有程序设计的那天起测试就一直伴随着。统计表明，在典型的软件开发项目中，软件测试工作量往往占软件开发总工作量的40％以上。而在软件开发的总成本中，用在测试上的开销要占30％到50％如果把维护阶段也考虑在内，讨论整个软件生存期时，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，其中必定还包含有许多测试工作 。

软件质量概述

- 软件测试的认识误区：

1）软件开发完成后进行软件测试

2）软件发布后如果发现问题，那是软件测试人员的错

3）软件测试要求不高，随便找个人都行

4）软件测试是测试人员的事情，与程序员无关

5）项目进度吃紧是时少做测试，时间富裕时多做测试

6）软件测试是没有前途的工作，只有程序员才是软件高手

这些观点对软件测试工作是极为不利的，必须澄清认识、端正态度才可能提高软件产品的质量。

软件质量概述

- 软件测试目的：

如果测试的目的是为了尽可能多地找出错误，那么测试就应该直接针对软件比较复杂的部分或是以前出错比较多的位置。如果测试目的是为了给最终用户提供具有一定可信度的质量评价，那么测试就应该直接针对在实际应用中会经常用到的商业假设 。

在谈到软件测试时，许多人都引用Grenford J. Myers在《The Art of Software Testing》一书中的观点:

(1)软件测试是为了发现错误而执行程序的过程;

(2)测试是为了证明程序有错，而不是证明程序无错误

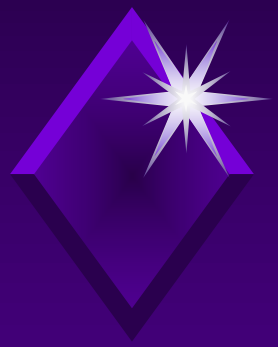
(3)一个好的测试用例是在于它能发现至今未发现的错误

(4)一个成功的测试是发现了至今未发现的错误的测试

软件质量概述

- 软件测试目的：

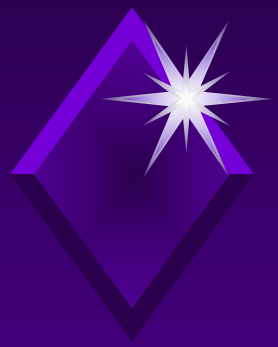
这种观点可以提醒人们测试要以查找错误为中心，而不是为了演示软件的正确功能。但是仅凭字面意思理解这一观点可能会产生误导认为发现错误是软件测试的唯一目，查找不出错误的测试就是没有价值的，事实并非如此首先，测试并不仅仅是为了要找出错误。通过分析错误产生的原因和错误的分布特征，可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。同时种分析也能帮助我们设计出有针对性地检测方法，改善测试的有效性其次，没有发现错误的测试也是有价值的，完整的测试是评定测试质量的一种方法 。



软件测试的组织与管理

Topics

- 测试的过程及组织
- 测试的人员组织
- 软件测试文件



软件测试的组织与管理

- 测试的过程及组织：

随着软件开发规模的增大、复杂程度的增加，以寻找软件中的错误为目的的测试工作就显得更加困难。然而，为了尽可能多地找出程序中的错误，生产出高质量的软件产品，加强对测试工作的组织和管理就显得尤为重要

- 测试的过程及组织

当设计工作完成以后，就应该着手测试的准备工作了一般来讲，由一位对整个系统设计熟悉的设计人员编写测试大纲，明确测试的内容和测试通过的准则，设计完整合理的测试用例，以便系统实现后进行全面测试 。

软件测试的组织与管理

- 测试的过程及组织

在实现组将所开发的程序经验证后，提交测试组，由测试负责人组织测试，测试一般可按下列方式组织：

1）首先，测试人员要仔细阅读有关资料，包括规格说明、设计文档使用说明书及在设计过程中形成的测试大纲、测试内容及测试的过准则，全面熟悉系统，编写测试计划，设计测试用例，作好测试前的准备工作。

2）为了保证测试的质量，将测试过程分成几个阶段，即:代码审查单元测试、集成测试、确认测试和系统测试。

软件测试的组织与管理

- 测试的过程及组织

3）代码会审：代码会审是由一组人通过阅读、讨论和争议对程序进行静态分析的过程。会审小组在充分阅读待审程序文本、控制流程图及有关要求、规范等文件基础上，召开代码会审会，程序员逐句讲解程序的逻辑，并展开热烈的讨论甚至争议，以揭示错误的关键所在。实践表明，程序员在讲解过程中能发现许多自己原来没有发现的错误，而讨论和争议则进一步促使了问题的暴露。

4）单元测试：

单元测试集中在检查软件设计的最小单位—模块上，通过测试发现实现该模块的实际功能与定义该模块的功能说明不符合的情况，以及编码的错误。

软件测试的组织与管理

- 测试的过程及组织

3）集成测试：集成测试是将模块按照设计要求组装起来同时进行测试，主要目标是发现与接口有关的问题。如数据穿过接口时可能丢失；一个模块与另一个模块可能有由于疏忽的问题而造成有害影响把子功能组合起来可能不产生预期的主功能；个别看起来是可以接受的误差可能积累到不能接受的程度；全程数据结构可能有错误等 。

4）确认测试：确认测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是确认测试的任务，即软件的功能和性能如同用户所合理期待的那样 。

软件测试的组织与管理

- 测试的过程及组织

7）系统测试：软件开发完成以后，最终还要与系统中其他部分配套运行，进行系统测试。包括恢复测试、安全测试、强度测试和性能测试等。

经过上述的测试过程对软件进行测试后，软件基本满足开发的要求，测试宣告结束，经验收后，将软件提交用户。

软件测试的组织与管理

- 测试的人员组织

为了保证软件的开发质量，软件测试应贯穿于软件定义与开发的整个过程。因此，对分析、设计和实现等各阶段所得到的结果，包括需求规格说明、设计规格说明及源程序都应进行软件测试。基于此测试人员的组织也是分阶段的。

1）)软件的设计和实现都是基于需求分析规格说明进行的

需求分析规格说明是否完整、正确、清晰是软件开发成败的关键为了保证需求定义的质量，应对其进行严格的审查。

2）设计评审

软件设计是将软件需求转换成软件表示的过程。主要描绘出系统结构、详细的处理过程和数据库模式。按照需求的规格说明对系统结构的合理性、处理过程的正确性进行评价，同时利用关系数据库的规范理论对数据库模式进行审查。

软件测试的组织与管理

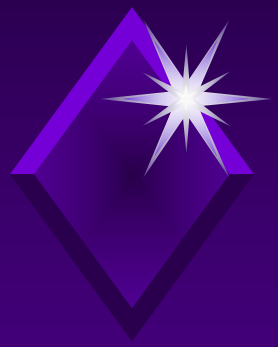
- 测试的人员组织

3）程序的测试：是指软件测试。是整个软件开发过程中交付用户使用前的最后阶段，是软件质量保证的关键。软件测试在软件生存周期中横跨两个阶段:通常在编写出每一个模块之后，就对它进行必要的测试（称为单元测试）。编码与单元测试属于软件生存周期中的同一阶段。该阶段的测试工作，由编程组内部人员进行交叉测试（避免编程人员测试自己的程序）。这一阶段结束后，进入软件生存周期的测试阶段，对软件系统进行各种综合的测试。测试工作由专门的测试组完成，负责整个测试的计划、组织工作测试组的其他成员由具有一定的分析、设计和编程经验的专业人员组成，人数根据具体情况可多可少，一般3～5人为宜。

软件测试的组织与管理

- 软件测试文件

软件测试文件描述要执行的软件测试及测试的结果。由于软件测试是一个很复杂的过程，同时也是设计软件开发其他一些阶段的工作对于保证软件的质量和它的运行有着重要意义，必须把对它们的要求、过程及测试结果以正式的文件形式写出。测试文件的编写是测试规范化的一个组成部分。测试文件不只在测试阶段才考虑，它在软件开发的需求分析阶段就开始着手，因为测试文件与用户有着密切的关系。在设计阶段的一些设计方案也应在测试文件中得到反映以利于设计的检验。测试文件对于测试阶段工作的指导与评价作用更是非常明显的。需要特别指出的是，在已开发的软件投入运行的维护阶段，常常还要进行再测试或回归测试，这时仍须用到测试文件 。



软件测试的组织与管理

- 软件测试文件

1）测试文件的类型

2）测试文件的使用

3）测试文件的编制

软件测试的组织与管理

- 测试文件的类型：

根据测试文件所起的作用不同，通常把测试文件分成两类，即测试计划和测试分析报告。测试计划详细规定测试的要求，包括测试的目的和内容、方法和步骤，以及测试的准则等。由于要测试的内容可能涉及到软件的需求和软件的设计，因此必须及早开始测试计划的编写工作。不应在着手测试时，才开始考虑测试计划。通常，测试计划的编写从需求分析阶段开始，到软件设计阶段结束时完成。测试报告用来对测试结果的分析说明，经过测试后，证实了软件具有的能力，以及它的缺陷和限制，并给出评价的结论性意见，这些意见即是对软件质量的评价，又是决定该软件能否交付用户使用的依据。由于要反映测试工作的情况，自然要在测试阶段内编写。

软件测试的组织与管理

- 测试文件的使用

测试文件的重要性表现在以下几个方面:

1）验证需求的正确性:测试文件中规定了用以验证软件需求的测试条件，研究这些测试条件对弄清用户需求的意图是十分有益的 。

2）检验测试资源:测试计划不仅要用文件的形式把测试过程规定下来还应说明测试工作必不可少的资源，进而检验这些资源是否可以得到，即它的可用性如何。如果某个测试计划已经编写出来，但所需资源仍未落实，那就必须及早解决 。

3）明确任务的风险:有了测试计划，就可以弄清楚测试可以做什么不能做什么。了解测试任务的风险有助于对潜伏的可能出现的问题事先作好思想上和物质上的准备

软件测试的组织与管理

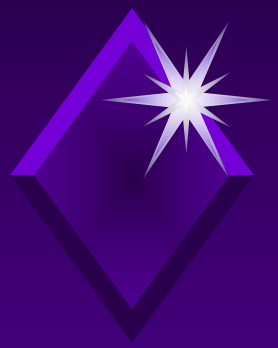
- 测试文件的使用

4）生成测试用例:测试用例的好坏决定着测试工作的效率，选择合适的测试用例是作好测试工作的关键。在测试文件编制过程中，按规定的要求精心设计测试用例有重要的意义 。

5）评价测试结果:测试文件包括测试用例，即若干测试数据及对应的预期测试结果。完成测试后，将测试结果与预期的结果进行比较，便可对已进行的测试提出评价意见。

6）再测试:测试文件规定的和说明的内容对维护阶段由于各种原因的需求进行再测试时，是非常有用的 。

7）决定测试的有效性:完成测试后，把测试结果写入文件，这对分析测试的有效性，甚至整个软件的可用性提供了依据。同时还可以证实有关方面的结论 。



软件测试的组织与管理

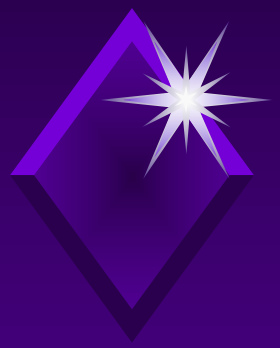
- 测试文件的编制

在软件的需求分析阶段，就开始测试文件的编制工作，各种测试文件的编写应按一定的格式进行 。具体格式可以根据各个公司的不同特点和标准制定固定的标准格式。

软件测试策略

- 软件测试策略：

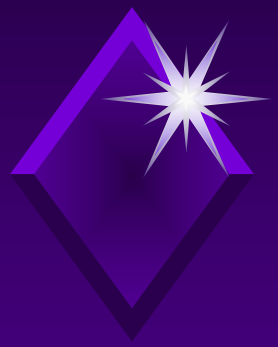
软件测试的策略、方法和技术是多种多样的。对于软件测试技术，可以从不同的角度加以分类：从是否需要执行被测软件的角度，可分为静态测试和动态测试。从测试是否针对系统的内部结构和具体实现算法的角度来看，可分为白盒测试和黑盒测试。



软件测试策略

Topics

- 静态方法与动态方法
- 功能测试与结构测试



软件测试策略

- 静态方法与动态方法

所谓静态方法是指不运行被测程序本身，仅通过分析或检查源程序的文法、结构、过程、接口等来检查程序的正确性。静态方法通过程序静态特性的分析，找出欠缺和可疑之处，例如不匹配的参数、不适当的循环嵌套和分支嵌套、不允许的递归、未使用过的变量、空指针的引用和可疑的计算等。静态测试结果可用于进一步的查错并为测试用例选取提供指导 。

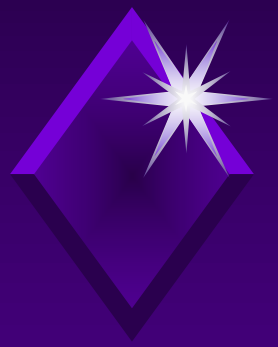
动态方法是指通过运行被测程序，检查运行结果与预期结果的差异并分析运行效率和健壮性等性能，这种方法由三部分组成：构造测试实例、执行程序、分析程序的输出结果。

软件测试策略

- 功能测试与结构测试——功能测试

功能测试是指在对程序进行的功能抽象的基础上，将程序划分成功能单元，然后在数据抽象的基础上，对每个功能单元生成测试数据进行测试。用这种方法进行测试时，被测程序被当作打不开的黑盒因而无法了解其内部构造，因此又称为黑盒测试。

黑盒测试也称功能测试或数据驱动测试，它是在已知产品所应具有的功能，通过测试来检测每个功能是否都能正常使用。在测试时，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口进行测试，只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当接收输入数据而产生正确的输出信息，并且保持外部信息的完整性 。



软件测试策略

- 功能测试与结构测试——功能测试

在功能测试中，被测软件的输入域和输出域往往是无限域，因此穷举测试通常是不可行的。必须以某种策略分析软件规格说明，从而得出测试用例集，尽可能全面而又高效地对软件进行测试。下面就说明几种功能测试的方法：

a。等价类划分

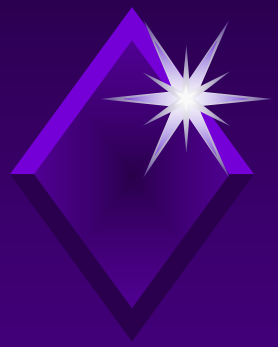
所谓等价类，就是指某个输入域的集合，集合中的每个输入对揭露程序错误来说是等效的，把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例，这就是等价类划分方法。它是功能测试的基本方法。

软件测试策略

- 功能测试与结构测试——功能测试

b。因果图法：因果图是一种形式语言，由自然语言写成的规范转换而成，这种形式语言实际上是一种使用简化记号表示数字逻辑图因果图法是帮助人们系统地选择一组高效测试用例的方法，此外它还能指出程序规范中的不完全性和二义性。

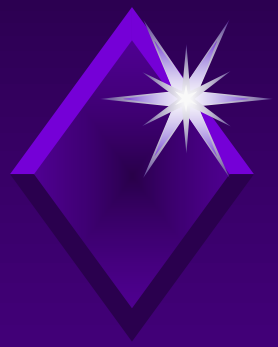
C。边值分析：实践证明，软件在输入、输出域的边界附近容易出现差错，边值分析是考虑边界条件而选取测试用例的一种功能测试方法。所谓边界条件，是相对于输入和输出等价类直接在其边缘上，稍高于和稍低于其边界的这些状态条件。边值分析是对等价类划分的有效补充。



软件测试策略

- 功能测试与结构测试——结构测试

结构测试是根据被测程序的内部结构设计测试用例的一类测试，又称为白盒测试。白盒测试也称结构测试或逻辑驱动测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能。其主要方法有逻辑驱动、基路测试等，主要用于软件验证。白盒法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。白盒法是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。但即使每条路径都测试了仍然可能有错误。



软件测试策略

- 软件测试策略

第一，穷举路径测试决不能查出程序违反了设计规范，即程序本身错误的程序。第二，穷举路径测试不可能查出程序中因遗漏路径而出错。第三，穷举路径测试可能发现不了一些与数据相关的错误与功能测试不同的是，结构测试涉及程序内部结构。尽管用户更倾向于基于程序规格说明的功能测试，但是结构测试能发现潜在的逻辑错误，而这种错误往往是功能测试发现不了的。它们各有利弊常常结合使用 。

软件测试策略

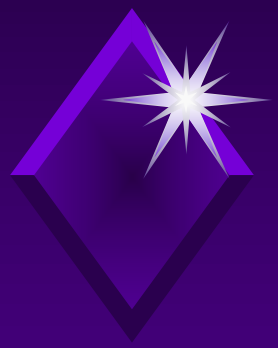
- 功能测试与结构测试——结构测试

结构测试方法：

a。采用逻辑覆盖的结构测试。逻辑覆盖又包含以下五种:语句覆盖判定覆盖、条件覆盖、判定与条件覆盖、路径覆盖。

b。域测试。这是一种基于程序结构的测试方法。这里的“域”是指程序的输入空间。域测试正是在分析输入空间的基础上，选择适当点以后进行测试的。

C。符号测试。符号测试是基于代数运算的一种结构测试方法。符号测试方法受分支问题、二义性问题和大程序问题的困扰，这些问题严重地影响着它的发展前景 。



软件测试策略

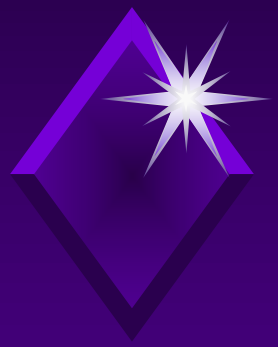
- 功能测试与结构测试——结构测试

结构测试方法：

d。数据流测试。数据流测试是指一个基于通过程序的控制流，从建立的数据目标状态的序列中发现异常的结构测试方法。

e。定义域测试。定义域测试的重点在分类方面，它还要判断出定义域的正确性。定义域测试与集合理论密切相关。

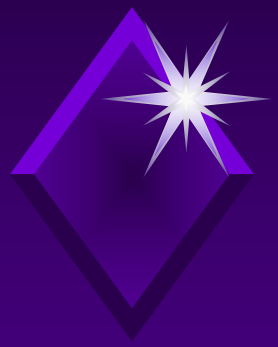
f。程序变异测试。这是一种基于程序错误的测试方法，它的目的是要说明程序中不含有某些特定的错误。



软件测试步骤

- 软件测试步骤：

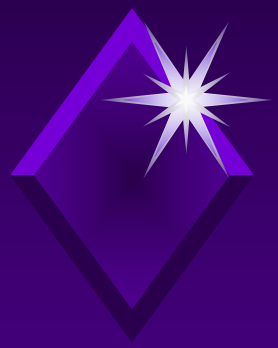
软件测试过程一般按四个步骤进行，即单元测试、集成测试、系统测试和确认测试



软件测试步骤

Topics

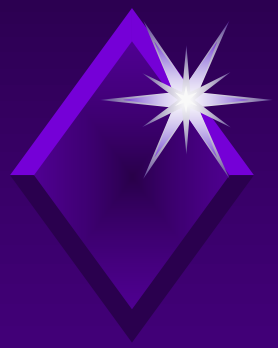
- 单元测试
- 集成测试
- 系统测试
- 确认测试



软件测试步骤

- 单元测试：

也称模块测试，这是针对软件设计的最小单位-模块进行正确性检验的测试工作，其目的在于发现各模块内部可能存在的各种差错。单元测试的依据是详细设计描述单元测试应对模块内所有重要的控制路径设计测试用例以便发现模块内部的错误。单元测试多采用结构测试（白盒测试）技术，系统内多个模块可以并行地进行测试。



软件测试步骤

- 单元测试：

a。单元测试的任务

(1)模块接口测试

(2)模块局部数据结构测试

(3)模块边界条件测试

(4)模块中所有独立执行通路测试

(5)模块的各条错误处理通路测试

软件测试步骤

- 单元测试：

a。单元测试的任务

模块接口测试是单元测试的基础。只有在数据能正确流入、流出模块的前提下，其他测试才有意义。测试接口正确与否应该考虑下列因素：

(1) 输入的实际参数与形式参数的个数是否相同
(2) 输入的实际参数与形式参数的属性是否匹配
(3) 输入的实际参数与形式参数的量纲是否一致
(4) 调用其他模块时所给实际参数的个数是否与被调模块的形参个数相同
(5) 调用其他模块时所给实际参数的属性是否与被调模块的形参属性匹配

软件测试步骤

- 单元测试：

a。单元测试的任务

(6) 调用其他模块时所给实际参数的量纲是否与被调模块的形参量纲一致；

(7) 调用预定义函数时所用参数的个数属性和次序是否正确

(8) 是否存在与当前入口点无关的参数引用；

(9) 是否修改了只读型参数；

(10) 各模块对全程变量的定义是否一致；

(11) 是否把某些约束作为参数传递

软件测试步骤

- 单元测试：

a。单元测试的任务

如果模块内包括外部输入输出，还应该考虑下列因素：

(1)文件属性是否正确

(2)OPEN/CLOSE语句是否正确

(3)格式说明与输入输出语句是否匹配

(4)缓冲区大小与记录长度是否匹配

(5)文件使用前是否已经打开

(6)是否处理了文件尾

(7)是否处理了输入/输出错误

(8)输出信息中是否有文字性错误

软件测试步骤

- 单元测试：

a。单元测试的任务

检查局部数据结构是为了保证临时存储在模块内的数据在程序执行过程中完整、正确。局部数据结构往往是错误的根源，应仔细设计测试用例，力求发现以下几类错误：

(1)不合适或不相容的类型说明；

(2)变量无初值

(3)变量初始化或省缺值有错

(4)不正确的变量名（拼错或不正确地截断）

(5)出现上溢、下溢和地址异常

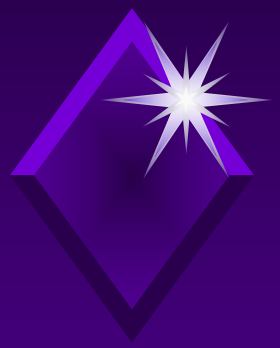
软件测试步骤

- 单元测试：

a。单元测试的任务

除局部数据结构外，如果可能，单元测试时还应该查清全局数据对模块的影响。 在模块中应对每一条独立执行路径进行测试，单元测试的基本任务是保证模块中每条语句至少执行一次。此时设计测试用例是为了发现因错误计算、不正确的比较和不适当的控制流造成的错误。此时基本路径测试和循环测试是最常用且最有效的测试技术。计算中常见的错误包括

(1) 误解或用错了算符优先级
(2) 混合类型运算
(3) 变量初值错
(4) 精度不够
(5) 表达式符号错



软件测试步骤

- 单元测试：

a。单元测试的任务

比较判断与控制流紧密相关，测试用例还应致力于发现下列错

(1)不同数据类型的对象之间进行比较

(2)错误地使用逻辑运算符或优先级

(3)因计算机表示的局限性，期望理论上相等而实际上不相等的两个量相等

(4)比较运算或变量出错

(5)循环终止条件不可能出现

(6)迭代发散时不能退出

(7)错误地修改了循环变量

软件测试步骤

- 单元测试：

a。单元测试的任务

一个好的设计应能预见各种出错条件，并预设各种出错处理通路，出错处理通路同样需要认真测试，测试应着重检查下列问题：

(1)输出的出错信息难以理解

(2)记录的错误与实际遇到的错误不相符

(3)在程序自定义的出错处理段运行之前，系统已介入

(4)异常处理不当

(5)错误陈述中未能提供足够的定位出错信息

边界条件测试是单元测试中最后也是最重要的一项任务。众所周知软件经常在边界上失效。采用边界值分析技术，针对边界值及其左右设计测试用例，很有可能发现新的错误.。

软件测试步骤

- 单元测试：

b。单元测试的过程

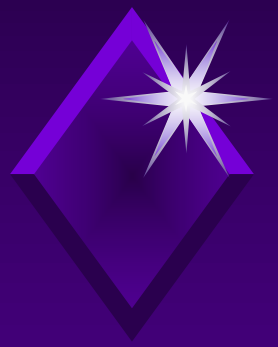
一般认为单元测试应紧接在编码之后，当源程序编制完成并通过复审和编译检查，便可开始单元测试。测试用例的设计应与复审工作相结合，根据设计信息选取测试数据，将增大发现上述各类错误的可能性。在确定测试用例的同时，应给出期望结果。

提高模块的内聚度可简化单元测试，如果每个模块只能完成一个，所需测试用例数目将显著减少，模块中的错误也更容易发现。

软件测试步骤

- 集成测试

也称组装测试，在单元测试之后，需要按照设计时作出的结构图，将它们联结起来，进行集成测试。集成测试是组装软件的系统测试技术，按设计要求把通过单元测试的各个模块组装在一起之后，进行综合测试以便发现与接口有关的各种错误 。把所有模块按设计要求一次全部组装起来，然后进行集成测试，这称为非增量式集成。这种方法容易出现混乱。因为测试时可能发现一大堆错误，为每个错误定位和纠正非常困难，并且在改正一个错误的同时又可能引入新的错误。新旧错误混杂，更难断定出错的原因和位置。与之相反的是增量式集成方法，程序一段一段地扩展，测试的范围一步一步地增大，错误易于定位和纠正，界面的测试亦可做到完全彻底。常用的有下面两种增量集成方法。



软件测试步骤

- 集成测试

a。自顶向下集成

自顶向下集成是构造程序结构的一种增量式方式，它从主控模块开始，按照软件的控制层次结构，以深度优先或广度优先的策略，逐步把各个模块集成在一起。深度优先策略首先是把主控制路径上的模块集成在一起，至于选择哪一条路径作为主控制路径，这多少带有些随意性，要根据问题的特性确定 。

软件测试步骤

- 集成测试

a。自顶向下集成

自顶向下集成测试的具体步骤为:

(1)以主控模块作为测试驱动模块，把对主控模块进行单元测试时引入的所有桩模块用实际模块替代

(2)依据所选的集成策略，每次只替代一个桩模块

(3)每集成一个模块立即测试一遍

(4)只有每组测试完成后，才着手替换下一个桩模块

(5)为避免引入新错误，须不断地进行回归测试。从第(2)步开始，循环执行上述步骤，直至整个程序结构构造完毕。

软件测试步骤

- 集成测试

a。自顶向下集成

自顶向下集成的优点在于能尽早地对程序的主要控制和决策机制进行检验，因此能较早地发现错误。缺点是在测试较高层模块时，低层处理采用桩模块替代，不能反映真实情况，重要数据也不能及时回送到上层模块，因此测试并不充分。解决这个问题有几种办法，第一种是把某些测试推迟到用真实模块替代桩模块之后进行

第二种是开发能模拟真实模块的桩模块

第三种是自底向上集成模块

第一种方法又回退为非增量式的集成方法，使错误难于定位和纠正并失去了在组装模块时进行一些特定测试的可能性;第二种方法无疑要大大增加开销;第三种方法更切实可行。

软件测试步骤

- 集成测试

b。自底向上集成

自底向上测试是从软件结构最低层的模块开始组装测试。因测试到较高层模块时，所需的下层模块功能均已具备，所以不再需要桩模块。

自底向上集成测试的步骤分为:(1)把低层模块组织成实现某个子功能的模块群;(2)开发一个测试驱动模块，控制测试数据的输入和测试结果的输出;(3)对每个模块群进行测试;(4)删除测试使用的驱动模块，用较高层模块把模块群组织成为完成更大功能的新模块群。从第(1)步开始循环执行上述各步骤，直至整个程序构造完毕。

软件测试步骤

- 集成测试

b。自底向上集成

自底向上集成方法不用桩模块，测试用例的设计亦相对简单，但缺点是程序最后一个模块加入时才具有整体形象。它与自顶向上集成测试方法优缺点相反。因此，在测试软件系统时，应根据软件的特点和工程的进度，选用适当的测试策略。有时混和使用两种策略更为有效，上层模块用自顶向下的方法，下层模块用自底向上的方法 此外，在集成测试中尤其要注意关键模块。所谓关键模块一般都具有下述一个或多个特征:(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。关键模块应尽早测试，并反复进行回归测试。

软件测试步骤

u 集成测试

c。确认测试

也称合格性测试，这是检验所开发的软件是否按用户要求运行。确认测试应检查软件能否按合同要求进行工作，即是否满足软件需求说明书中的确认标准。

1）.确认测试标准

实现软件确认要通过一系列黑盒测试。确认测试同样需要制订测试计划和过程，测试计划应规定测试的种类和测试进度，测试过程则定义一些特殊的测试用例，旨在说明软件与需求是否一致。无论是测试计划还是测试过程，都应该着重考虑软件是否满足合同规定的所有功能和性能，文档资料是否完整、准确，人机界面、可移植性、兼容性、可维护性等是否令用户满意 。

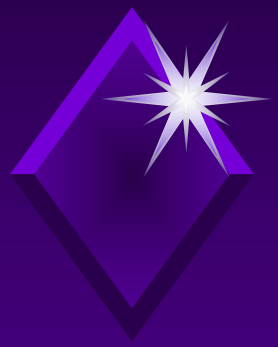
软件测试步骤

- 集成测试

c。确认测试

2）配置复审

确认测试的另一个重要环节是配置复审。复审的目的在于保证软件配置齐全、分类有序，并且包括软件维护所必须的细节。



软件测试步骤

u 集成测试

c。确认测试

3） α、β测试

事实上，软件开发人员不可能完全预见用户实际使用程序的情况。例如，用户可能错误地理解命令，或提供一些奇怪的数据组合，亦可能对设计者自认明了的输出信息迷惑不解。因此，软件是否真正满足最终用户的要求，应由用户进行一系列“验收测试”。验收测试既可以是非正式的测试，也可以是有计划、有系统的测试。有时，验收测试长达数周甚至数月，不断暴露错误，导致开发延期。一个软件产品，可能拥有众多用户，不可能由每个用户验收，此时多采用称为α、β测试的过程，以期发现那些似乎只有最终用户才能发现的问题 。

软件测试步骤

- 集成测试

c。确认测试

α测试是指软件开发公司组织内部人员模拟各类用户对即将面市的软件产品（称为α版本）进行测试，试图发现错误并修正。α测试的关键在于尽可能逼真地模拟实际运行环境和用户对软件产品的操作并尽最大努力涵盖所有可能的用户操作方式。经过α测试调整的软件产品称为β版本。紧随其后的β测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用β版本，并要求用户报告异常情况、提出批评意见。然后软件开发公司再对β版本进行改错和完善。

软件测试步骤

u 集成测试

d。系统测试

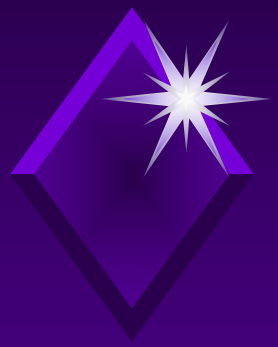
软件开发完成后，还要与系统中其他部分配套运行，进行系统测试包括恢复测试、安全测试、强度测试和性能测试等。在系统测试之前，软件工程师应完成下列工作:

(1)为测试软件系统的输入信息设计出错处理通路

(2)设计测试用例，模拟错误数据和软件界面可能发生的错误，记录测试结果，为系统测试提供经验和帮助

(3)参与系统测试的规划和设计，保证软件测试的合理性

系统测试应该由若干个不同测试组成，目的是充分运行系统，验证系统各部件是否都能工作并完成所赋予的任务。下面简单介绍几类系统测试。



软件测试步骤

- 集成测试

d。系统测试

(1)恢复测试

恢复测试主要检查系统的容错能力。当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。恢复测试首先要采用各种办法强迫系统失败，然后验证系统是否能尽快恢复。对于自动恢复需验证重新初始化、检查点、数据恢复和重新启动等机制的正确性;对于人工干预的恢复系统，还需估测平均修复时间，确定其是否在可接受的范围内。

软件测试步骤

u 集成测试

d。系统测试

(2)安全测试

安全测试检查系统对非法侵入的防范能力。安全测试期间，测试人员假扮非法入侵者，采用各种办法试图突破防线。

例如：

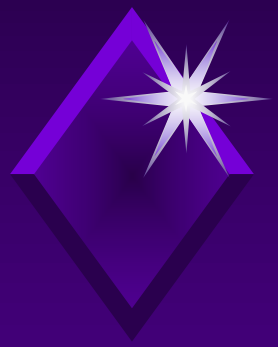
①想方设法截取或破译口令

②专门定做软件破坏系统的保护机制

③故意导致系统失败，企图趁恢复之机非法进入

④试图通过浏览非保密数据，推导所需信息

理论上讲，只要有足够的时间和资源，没有不可进入的系统。因此系统安全设计的准则是使非法侵入的代价超过被保护信息的价值此时非法侵入者已无利可图。



软件测试步骤

- 集成测试

d。系统测试

(3) 强度测试

强度测试检查程序对异常情况的抵抗能力。强度测试总是迫使系统在异常的资源配置下运行。例如，当中断的正常频率为每秒一至两个时，运行每秒产生十个中断的测试用例;定量地增长数据输入率检查输入子功能的反映能力;运行需要最大存储空间（或其他资源）的测试用例;运行可能导致虚存操作系统崩溃或磁盘数据剧烈抖动的测试用例等等 。

软件测试步骤

u 集成测试

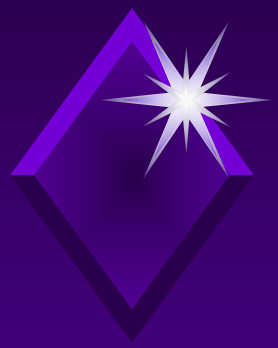
d。系统测试

(4) 性能测试

对于那些实时和嵌入式系统，软件部分既使能满足功能要求，也未必能够满足性能要求。虽然从单元测试起，每一测试步骤都包含性能测试，但只有当系统真正集成之后，在真实环境中才能全面、可靠地测试运行性能系统。性能测试是为了完成这一任务。性能测试有时与强度测试相结合，经常需要其他软硬件的配套支持。 只有经过上述测试过程测试后，软件才能基本满足开发要求。测试宣告结束，经验收后，将软件提交用户使用。

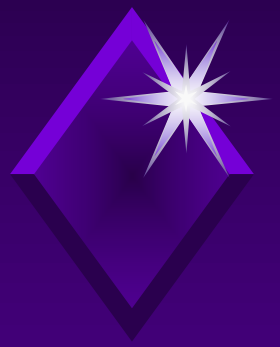
实例讲解

- 略



总结

软件质量是软件产品的生命之所在，软件测试作为保证软件质量的手段，愈来愈受到人们的重视。而如何提高软件产品质量，严格的测试是重要的一环。软件测试理论和方法在不断完善，测试工具也在蓬勃发展。测试已从简单的检查程序逻辑走向“确认、验证和测试又走向全面形式化的道路。 本文绍了软件测试的各种方法、测试过程的管理软件测试组织实施目的在于希望从事软件测试的同仁们在准备从事软件测试工作前，先明确什么是软件测试、软件测试的基本方法，只有打好理论基础才可以做好测试工作，并且要注重测试过程的管理，而不要一味的迷恋于测试工具。



谢 谢！