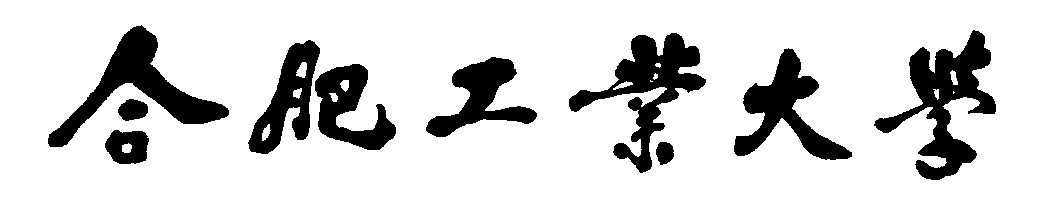


****

计算机与信息学院

软件测试期末报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计科21-3班 |
| 姓名 | 陈嘉乐 |
| 学号 | 2021218152 |
| 任 课 教 师 | 于磊 |
| 完成日期 | 2024.4.25 |

目录

[第一章 绪论 3](#_Toc165047906)

[第二章 软件测试策略 3](#_Toc165047907)

[第三章 黑盒测试 4](#_Toc165047908)

[第四章 白盒测试 5](#_Toc165047909)

[第五章 集成测试 6](#_Toc165047910)

[第六章 系统测试 6](#_Toc165047911)

[第七章 测试组织和管理 7](#_Toc165047912)

[第九章 软件评审 7](#_Toc165047913)

[第十章 软件质量与软件质量管理 8](#_Toc165047914)

[课程体会与心得 8](#_Toc165047915)

# 第一章 绪论

本章主要介绍了软件测试的原则、基本定义、发展和生命周期等主要内容。

软件的生存期包括计划阶段、需求分析、设计构思、编码实现、测试验证以及后续的维护。而在这整个流程中，软件测试扮演着至关重要的角色，它是保障软件质量不可或缺的一环。软件测试不仅致力于验证软件的准确性，更深入挖掘软件潜在的错误与不足。在测试原则方面，它坚守六大核心准则，包括认识到测试的局限、激发创新思维、侧重预防错误，以及全面评估测试所伴随的风险等。此外，随着技术的不断进步，软件测试也在持续发展与完善。

据Boehm统计，软件测试占软件开发总成本的30%～50%。若考虑维护阶段，测试成本比例或降低，但维护中的测试工作不容忽视。甚至有人估计软件工作中有50%以上的时间和成本花在测试上。

值得一提的是，软件测试还与其他多个领域紧密相连，如图论、程序正确性验证等，这些领域共同构成了软件测试的综合性知识体系。而在实际操作中，信息流的管理尤为关键，它涵盖了软件配置详情、测试需求以及测试工具的选择与使用，确保测试工作的精准与高效。

# 第二章 软件测试策略

本章主要介绍了软件测试的设计和管理。测试设计内容包括过程设计、类型设计和数据设计三大部分。测试管理内容包括流程管理和数据管理两大部分。

软件测试是软件开发的一个阶段，而软件测试同样可以划分成不同的阶段，因此它也具备了一个完整的生命周期，这个周期涵盖了计划、设计、开发、执行以及评估这五个核心阶段。

首先是计划阶段，通过系统分析制定针对性的测试策略和规范，并设计充足的测试用例，同时安排测试进度和资源。接着是设计阶段，重点在于规划测试过程和测试用例，包括确定测试内容、顺序，以及满足发现Bug和评估系统性能的需求。进入开发阶段，为提高效率，需开发测试工具以支持自动化测试脚本、数据生成及流程。随后是执行阶段，通过手动或自动方式执行测试用例。最后是评估阶段，对测试结果进行综合评价，统计错误数量，从而提供对被测系统的准确评价，确保测试的完整性和有效性。这五个阶段相互衔接，共同构成了软件测试的完整流程。

软件测试的步骤可以被形象地比喻为一条螺旋线，其过程包括单元测试、集成测试、确认测试和系统测试这四个关键步骤。首先是单元测试，它针对程序的最小组成单位——单元进行，旨在检验单元内部是否存在错误。单元测试通常由编码人员自己完成，采用白盒测试方法，从程序的内部结构出发设计测试用例，以确保每个单元都符合预期的功能要求。随后是集成测试，它将已完成的单元按照设计结构连接起来，检验模块间接口的正确性和全局功能。接下来是确认测试，该阶段主要检查系统是否满足规格说明和用户需求，确保软件的整体功能正确无误。最后是系统测试，它涵盖了恢复测试、安全测试、强度测试、容量测试、性能测试等多个方面，以全面评估系统的稳定性、可靠性和性能表现。此外，还包括α测试和β测试，分别模拟开发环境和实际使用环境下的测试场景，以及安装测试和可使用性测试，确保软件安装顺利且易于使用。这些步骤共同构成了软件测试的完整流程，旨在确保软件质量，提升用户体验。

软件测试的方法可大致归为静态与动态两大类别。静态分析聚焦于对被测程序的特性进行深入细致的剖析，从而获取程序内在的逻辑和特性。相对而言，动态分析则更注重实际的执行过程，其中黑盒测试和白盒测试是两种主要的测试形式。黑盒测试主要关注程序的输入与输出，而不过多关心其内部逻辑；而白盒测试则深入程序的内部结构，检查其代码逻辑的正确性。

此外，软件测试还涵盖了人工测试和自动测试这两种形式。人工测试更多地依赖于测试人员的专业知识和丰富经验，他们凭借自身的技能和洞察力来进行测试，而不完全依赖于计算机。而自动测试则通过编写计算机程序来自动执行测试任务，这种方式可以大大提高测试的效率，并减少人为因素导致的错误。

软件审查则是一个全面而系统的工作流程，主要包括制定计划、预审、准备、审查会、返工和终审等步骤，确保软件开发的规范性和质量。它的作用体现在利用审查数据指导和改进开发工作，同时作为软件开发进程的控制手段，在关键控制点实施阶段审查，以促进软件开发的顺利进行。

# 第三章 黑盒测试

本章主要介绍了黑盒测试的五大方法，具体如下：

①等价类划分：等价类划分是黑盒测试的典型方法，它专注于根据程序的需求规格说明书来设计测试用例，而不考虑程序的内部结构。这一方法首先分析需求，将输入和输出要求区分并分解，然后划分成有效和无效等价类。有效等价类包含有意义的输入数据，而无效等价类则包含不合理或无意义的输入数据。确定等价类时，需遵循一系列原则，如根据取值范围或值的个数、输入值的集合或“必须如何”的条件来划分。最后，根据等价类表确定测试用例，确保所有有效和无效等价类都被覆盖。

②因果图：首先，分析程序规格说明，确定原因（输入条件或等价类）和结果（输出条件）。其次，根据语义内容绘制因果图，标明原因与结果之间的关系。然后，考虑语法或环境限制，在因果图上标注约束条件。接着，将因果图转换为判定表。最后，将判定表中的每一列情况编写成测试用例。此外，输入条件存在E（异）、I（或）、O（唯一）、R（要求）四类约束，而输出条件则有M（强制）约束。

③正交实验设计法：它是一种科学的实验设计方法，通过正交表从大量实验点中挑选出适量、有代表性的点来合理安排实验。这种方法确保所有因子和水平在实验中均匀分配与搭配，变化规律均匀。在软件功能测试中，首先要提取功能说明，构造因子—状态表，然后加权筛选生成因素分析表，最后利用正交表构造测试数据集。通过逻辑命令将输入与输出结合，绘制布尔图，从而分析各因子状态。

④边值分析：在设计测试用例时，针对单变量边界值的选取，应依据输入或输出条件的取值范围或个数，采用五点法或七点法选择边界值。对于有序集合的输入或输出域，需特别选取集合的首尾元素作为测试用例。同时，还需分析规格说明以识别其他潜在边界条件。在多个变量组合的情况下，根据可靠性理论，可固定其他变量为正常值，对单个变量应用五点法或七点法来选取边界值，并重复此过程直至每个变量都被考虑。

⑤判定表驱动测试：判定表是一种工具，用于表达多个逻辑条件与操作之间的关联。在功能测试中，当软件规格说明包含复杂的条件与操作关系时，判定表能清晰地展示这些关系。使用判定表设计测试用例的条件包括：规格说明易于转换、条件与规则顺序不影响操作、满足规则后无需检验其他规则等。这些条件确保了判定表在测试中的有效性。

最后，功能测试是模块测试的关键方法，与结构测试不同，它主要关注模块的输入输出信息是否符合功能需求，而不深究内部实现逻辑。

# 第四章 白盒测试

本章主要介绍了白盒测试的九种具体分析方法，如下：

①程序结构分析：涉及控制流图、控制流图矩阵及结构要求。控制流图用结点和流线表示程序流程，可转化为矩阵形式便于处理。程序结构需避免不存在标号转向、无用标号、无法到达或停机的语句。结构分析包括流程图逐步细化，结点可扩充代换。

②数据流分析：涉及变量在程序中的定义和引用。定义是修改变量值的语句，引用是使用变量值的语句。可达性定义是变量在控制流图中被定义的路径。通过分析可达定义表，可识别引用未定义变量的错误。同时，也要检查是否存在未曾使用的定义，即定义的变量在后续程序中未被引用。这些方法有助于发现数据流相关的问题。

③信息流分析

④逻辑覆盖：几种常用的逻辑覆盖测试方法有语句覆盖、判断覆盖、条件覆盖、判断条件覆盖和路径覆盖。

⑤域测试：域测试是基于程序输入空间分析的测试方法，它通过分析输入空间选择测试点，以验证输入元素是否产生正确结果。但因其使用限制和涉及多维空间概念，推广和实用性受限。子空间划分由程序分支语句谓词决定，测试点选择基于输入域分析。

⑥符号测试：是一种执行代数运算的测试方法，代表一类测试，无需具体数值数据。它结合了测试和验证特点，既运行程序检验可靠性，又通过单次测试代表多类普通测试结果。关键在于开发处理符号运算的编译器或解释器。实际应用包括符号执行、分支路径处理、化简路径条件、求解测试数据，最后进行测试。若路径条件无解，则路径不可执行。

⑦路径分析：程序路径分析是白盒测试的核心，涉及路径表示与计算。路径可用弧序列、结点序列或路径表达式描述，后者以控制流图弧为运算对象，包括乘和加运算。路径数计算涉及表达式计算、程序复杂度评估和独立路径数确定。独立路径指从入口到出口每次至少执行一新语句，在控制流图中表现为至少经过一个未走过的弧。

⑧程序插装：是一种基本的测试手段，通过在被测程序中插入操作来实现测试目的。插装程序能记录并输出各程序点上语句的实际执行次数，设计时需考虑探测信息、设置探测点的位置和数量。断言语句是在程序特定部位插入的判断变量特性的语句，用以证实程序的运行特性，是程序正确性证明的实用方法。

⑨程序变异：是一种错误驱动测试，针对某类特定程序错误的测试方法，分为程序弱变异和强变异。

# 第五章 集成测试

本章概括了集成测试的必要性，包括解决组件接口、功能、时序和资源竞争等问题的重要性。同时介绍了程序结构分析的内容，并详细描述了十种不同的集成测试方法，包括它们的优缺点。最后，强调了基于调用图的集成方式在减少驱动和桩模块开发工作方面的应用。

①一次性集成是一种基于分解的集成测试方法，通过少数测试运行来检测整个系统以验证稳定性。它将所有模块集合在一起进行集成测试，而不考虑模块间的相依性或风险。适用于系统稳定且只加入少数构件，或系统较小且每个模块已充分测试的情况。一次性集成测试通常不建议进行，因为可能存在众多接口错误导致测试失败，并难以确定错误来源。

②自顶向下集成是一种广泛采用的软件测试方法，从顶层控制对象开始，逐次增加构件，以验证系统稳定性。通过控制层次顺序，以桩模块替代下层单元进行测试，逐步集成直至所有构件被测试。适用于增量式系统开发、并行软硬件开发等场景。其优点在于早期发现错误、减少驱动模块开发费用，但桩模块开发代价大，底层需求变化可能导致顶层修改，底层构件测试困难。因此，在选择自顶向下集成时需权衡其优缺点。

③自底向上集成是一种从底层构件开始，按相依性次序向上集成，以验证系统稳定性的测试方法。它适用于子系统的迭代和增量开发，适合具有稳定接口定义的构件系统。优点包括并行测试和集成、减少桩模块需求，但驱动模块开发耗费大，高层构件测试可能不充分。因此，在选择自底向上集成时，也需权衡其优缺点。

# 第六章 系统测试

本章概述了系统测试的目的，即通过与需求定义比较来发现软件与系统定义不符合或矛盾的地方。之后详细介绍了非功能测试、性能测试及其他测试（如α测试、β测试等）的目的、内容和要点，并提供了详细的测试检查表。此外，文档还穿插介绍了多种测试方法。

非功能测试包括安装测试、兼容性测试、安全性测试和恢复测试。

性能测试是评估产品或组件性能需求满足度的关键测试，通过评测和微调软件性能行为，确保性能需求得到满足。测试内容涵盖精确性、完成时间、数据处理量、协调性、软硬件限制、负载潜力及内存占用等方面，对软件质量保证至关重要。其包括负载测试。压力测试和容量测试。

除此之外，还有α测试是软件开发公司内部模拟用户测试α版本软件，力求发现并修正错误，以逼近实际运行环境和用户操作；β测试关注产品功能、可用性、可靠性和性能，同时评估支持性；界面测试则检查软件界面的规范性、合理性和方便性，确保用户友好和易用。

# 第七章 测试组织和管理

本章主要阐述了系统测试的组织、准备、实施和总结等多个环节，以及缺陷的跟踪和管理，旨在通过有效的测试组织和管理结合质量保证手段，提高软件质量。

首先，测试组织和管理的重要性不言而喻，通过合理分配测试资源、明确测试职责和工作流程，以及采用同行评审等质量保证手段，可以确保测试工作的顺利进行，并最大程度地减少测试过程中的错误和遗漏。

在测试准备阶段，我们关注测试需求分析、计划编写、环境搭建以及测试用例的设计等关键环节。进入测试实施阶段，我们注重测试用例的执行、数据记录、沟通和验证等核心环节。测试人员需严格按照测试用例进行测试，记录详细的测试数据，并与开发人员保持密切的沟通，确保测试结果的准确性和完整性。测试总结阶段则是对整个测试过程进行回顾和总结，包括测试数据整理、测试用例修订、用例库维护和配置管理等工作。

此外，缺陷的跟踪和管理也是测试工作中不可或缺的一部分。通过有效的缺陷描述、追踪和统计分析，我们可以确保每个缺陷都得到及时解决，同时识别和预防缺陷的频繁发生，进一步提高软件的质量和稳定性。

最后，测试成熟度模型CMM作为一种有效的指导工具，可以帮助软件开发组织完善和改进软件开发过程。通过对照CMM的五个级别，我们可以评估当前测试工作的成熟度和水平，并制定相应的改进计划，不断提升测试工作的专业性和高效性。

# 第九章 软件评审

本章主要介绍了软件评审的定义、发展历史、实例、意义和实施方法。

按IEEE的说法，评审是软件开发组之外的人员或小组对于软件需求、设计或代码进行详细审查的一种正式评价方法。其目的在于发现软件中的缺陷，找出违背执行标准的情况以及其他问题。

软件评审的意义：①及时排除软件开发过程中引入的缺陷②提高软件生产率，降低消除缺陷的成本③评审可为项目监控提供信息④评审可找出测试无法发现的缺陷⑤通过评审发现缺陷学习新知识

软件评审的实施涉及多个关键环节，包括正式评审、需求评审、设计评审和代码评审。正式评审需组建评审组，由组长负责策划、准备、主持及后续工作，确保评审公正客观。评审准备需明确被评审产品、评审目标等，并遵循评审原则。需求评审针对软件需求进行审查，设计评审旨在发现设计中的缺陷，包括初步、关键和程序设计评审。代码评审在编码后针对代码和文档进行，确保实现细节正确。这些评审活动全面保障软件开发质量，为成功上线奠定坚实基础。

# 第十章 软件质量与软件质量管理

本章主要介绍了软件质量管理的内容。质量管理体系包括质量策划、控制、保证与改进四大方面，旨在确保软件质量达标。由于系统事故频发，常需分析事故性质与原因，通过若干途径、对错误类型分析和分类来解决软件质量问题。

对于估计程序中的隐藏错误数量，可以利用散播模型和回归模型，前者是从估算池塘中养鱼数或是野生动物数所用方法得来的，后者是测定前面的一些数据，描出数据的曲线，然后推测出曲线的未来部分。软件质量包括内部、外部和使用质量三类。

此外，软件质量管理还涉及多个相关过程，例如：软件质量保证过程、软件验证过程、软件确认过程、软件评审过程、审核过程、问题解决过程等。这些过程和任务共同保障了软件产品的稳定与高效。

# 课程体会与心得

通过为期8周的《软件测试》课程的深入学习，我深刻认识到之前对软件测试的理解过于片面，甚至一度将其与程序调试混为一谈。如今，我深刻地认识到了软件测试的重要性以及它在软件开发过程中的关键作用，更加明晰了软件测试的真正含义——它是在特定条件下执行软件，以发现潜在问题，进而提升软件质量的关键环节。

通过这门课程的学习，我系统掌握了软件测试的理论知识和实践技能，对软件测试的流程和方法有了更为深入的了解。软件测试并不仅仅是简单地找出软件中的错误和缺陷，更重要的是确保软件能够满足用户的需求和期望。在软件开发的各个阶段，软件测试都扮演着至关重要的角色，它既是质量的保障，也是用户信任的基石。

在课程中，老师带领我们深入探讨了黑盒测试与白盒测试的核心原理及其适用场景。黑盒测试着眼于软件的功能实现与用户体验的优化，通过模拟用户的日常操作来验证软件是否满足预期功能；而白盒测试则侧重于软件的内部逻辑与结构，通过细致审查代码来保障软件的稳健性和可维护性。这两种测试方法相辅相成，共同构建起了软件测试的坚实基础。

除了理论学习，我还通过实践操作，逐步熟悉了软件测试的全流程。这包括精心制定测试计划、设计具有针对性的测试用例、搭建高效的测试环境、严格执行测试并记录结果。我亲自编写了测试用例，并借助先进的测试工具实施自动化测试，同时学会了撰写详尽的测试报告以及进行缺陷跟踪。这些实践经验让我对软件测试有了更为直观且深刻的理解。

在软件测试管理方面，我亦有所收获。我学习了如何有效管理测试用例、追踪软件缺陷以及促进团队之间的协作。这些管理技巧不仅显著提升了测试工作的效率与质量，更加强了团队内部的沟通与协作，使我们的软件测试工作更加高效有序。

我深知软件测试是一个持续学习和进步的过程。随着软件技术的不断更新和发展，新的测试方法和工具不断涌现。随着现代互联网企业规模的不断扩大，软件项目的复杂性、团队规模以及客户对质量的要求都在持续上升。在这样的背景下，软件测试的重要性日益凸显，测试方法和理论研究也呈现出蓬勃发展的态势。因此，作为一名软件测试人员，我将保持学习的态度，不断提升自己的技能水平，以适应不断变化的软件测试领域。

总的来说，这次软件测试课程的学习让我受益匪浅。我不仅掌握了软件测试的基本知识和技能，更对软件测试的重要性和意义有了更为深刻的认识。我相信，在未来的工作中，我将能够运用所学知识，为软件的质量和稳定性贡献自己的力量。