

<<软件工程与计算>>

图书基本信息

书名：<<软件工程与计算>>

13位ISBN编号：9787111407508

10位ISBN编号：7111407504

出版时间：2012-12

出版时间：机械工业出版社

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<软件工程与计算>>

内容概要

《高等院校软件工程专业规划教材：软件工程与计算（卷2）：软件开发的技术基础》作为国家精品课程“软件工程与计算”系列课程的第二门课程配套教材，《高等院校软件工程专业规划教材：软件工程与计算（卷2）：软件开发的技术基础》以经典软件工程方法与技术为主线，软件开发技术与程序设计知识为教学重点，培养学生简单小组级别、中小规模软件系统的软件开发能力。

全书主要分为六部分。

第一部分介绍软件工程的基本框架。

第二部分介绍项目启动阶段的知识。

第三部分介绍软件需求开发的基础知识，包括软件需求工程的概要、软件需求的内涵、常见的需求分析方法、软件需求文档。

第四部分首先介绍软件设计的基础概念，之后沿着设计过程和设计技术两条主线，深入描述软件设计的相关知识。

第五部分介绍软件构造、测试、移交与维护等软件开发的下游工程的基础知识。

第六部分是对第一部分的延续，通过总结性回顾，进一步加深读者对软件工程的理解。

《高等院校软件工程专业规划教材：软件工程与计算（卷2）：软件开发的技术基础》可作为高等院校软件工程、计算机及相关专业本科生软件工程课程的教材，也可作为从事软件开发的相关技术人员的参考书。

<<软件工程与计算>>

书籍目录

前言 第一部分软件工程概论 第1章软件工程基础 1.1软件 1.1.1软件独立于硬件 1.1.2软件是一种工具 1.1.3软件的核心是程序 1.1.4软件开发远比编程要复杂 1.1.5应用软件基于现实又高于现实 1.2软件工程 1.2.1定义 1.2.2软件工程是一种工程活动 1.2.3软件工程的动机 1.2.4软件工程是科学性、实践性和工艺性并重的 1.2.5软件工程追求足够好，不是最好 1.2.6软件的产品是基于虚拟计算机的软件方案 1.2.7软件工程的最终目的 1.3软件工程概览 1.3.1软件工程知识域 1.3.2软件开发活动 1.3.3软件的角色分工 1.4习题 第2章软件工程的发展 2.1软件工程的发展脉络 2.220世纪50年代的软件工程 2.320世纪60年代的软件工程 2.420世纪70年代的软件工程 2.520世纪80年代的软件工程 2.620世纪90年代的软件工程 2.721世纪00年代的软件工程 2.8习题 第二部分项目启动 第3章示例项目描述 3.1背景 3.2目标 3.3系统用户 3.4用户访谈要点 3.5项目实践过程 第4章项目管理基础 4.1项目和项目管理 4.2团队组织与管理 4.2.1团队的特征 4.2.2团队结构 4.2.3团队建设 4.3软件质量保障 4.3.1软件质量 4.3.2质量保障 4.3.3评审 4.3.4质量度量 4.4软件配置管理 4.4.1配置管理动机 4.4.2配置项 4.4.3基线 4.4.4配置管理活动 4.4.5变更控制 4.5项目实践 4.6习题 第三部分需求开发阶段 第5章软件需求基础 5.1引言 5.2需求工程基础 5.2.1需求工程简介 5.2.2需求工程活动 5.2.3需求获取 5.2.4需求分析 5.2.5需求规格说明 5.2.6需求验证 5.2.7需求管理 5.3需求基础 5.3.1需求 5.3.2需求的层次性 5.3.3结合层次性的需求开发 5.3.4区分需求、问题域与规格说明 5.4需求分类 5.4.1需求谱系 5.4.2软件需求的分类 5.5项目实践 5.6习题 第6章需求分析方法 6.1需求分析基础 6.1.1需求分析的原因 6.1.2需求分析模型 6.2结构化分析 6.2.1结构化分析方法 6.2.2数据流图 6.2.3实体关系图 6.3面向对象分析 6.3.1面向对象分析方法 6.3.2用例 6.3.3用例图 6.3.4用例描述 6.3.5概念类图（领域模型） 6.3.6交互图（顺序图） 6.3.7状态图 6.4使用需求分析方法细化和明确需求 6.4.1细化和明确需求内容 6.4.2建立系统级需求 6.5项目实践 6.6习题 第7章需求文档化与验证 7.1文档化的原因 7.2需求文档基础 7.2.1需求文档的交流对象 7.2.2用例文档 7.2.3软件需求规格说明文档 7.3需求文档化要点 7.3.1技术文档写作要点 7.3.2需求书写要点 7.3.3软件需求规格说明文档书写要点 7.4评审软件需求规格说明文档 7.4.1需求验证与确认 7.4.2评审需求的注意事项 7.5以需求为基础开发系统测试用例 7.5.1开发测试用例套件 7.5.2开发测试用例 7.6度量需求 7.7将需求制品纳入配置管理 7.8项目实践 7.9习题 第四部分软件设计 第8章软件设计基础 8.1软件设计思想的发展 8.2软件设计的核心思想 8.3理解软件设计 8.3.1设计与软件设计 8.3.2工程设计与艺术设计 8.3.3理性主义和经验主义 8.3.4软件设计的演化性 8.3.5软件设计的决策性 8.3.6软件设计的约束满足和多样性 8.4软件设计的分层 8.5软件设计过程的主要活动 8.6软件设计的方法和模型 8.6.1软件设计的方法 8.6.2软件设计的模型 8.7软件设计描述 8.7.1设计视图和设计图 8.7.2设计视角和设计关注 8.7.3需求和涉众 8.7.4设计理由 8.7.5设计描述的模板 8.7.6软件设计文档书写要点 8.8项目实践 8.9习题 第9章软件体系结构基础 9.1软件体系结构的发展 9.2理解软件体系结构 9.2.1定义 9.2.2区分软件体系结构的抽象与实现 9.2.3部件 9.2.4连接件 9.2.5配置 9.3体系结构风格初步 9.3.1主程序／子程序 9.3.2面向对象式 9.3.3分层 9.3.4MVC 9.4项目实践 9.5习题 第10章软件体系结构设计 10.1体系结构设计过程 10.1.1分析关键需求和项目约束 10.1.2选择体系结构风格 10.1.3软件体系结构逻辑设计 10.1.4软件体系结构实现 10.1.5完善软件体系结构设计 10.1.6定义构件接口 10.2体系结构的原型构建 10.2.1包的创建 10.2.2重要文件的创建 10.2.3定义构件之间的接口 10.2.4关键需求的实现 10.3体系结构集成与测试 10.3.1集成的策略 10.3.2桩、驱动与集成测试用例 10.4软件体系结构设计文档描述 10.5体系结构评审 10.6项目实践 10.7习题 第11章人机交互设计 11.1引言 11.2人机交互设计的目标 11.3人机交互设计的人类因素 11.3.1精神模型 11.3.2差异性 11.4人机交互设计的计算机因素 11.4.1可视化设计 11.4.2常见界面类型 11.5人机交互设计的交互性 11.5.1导航 11.5.2反馈 11.5.3一些人机交互设计原则 11.6人机交互设计过程 11.6.1基本过程 11.6.2示例 11.7项目实践 11.8习题 第12章详细设计的基础 12.1详细设计概述 12.1.1详细设计出发点 12.1.2详细设计的上下文 12.2结构化设计 12.2.1结构化设计的思想 12.2.2结构化设计的过程 12.3面向对象设计 12.3.1面向对象设计的思想 12.3.2面向对象设计的过程 12.3.3通过职责建立静态模型 12.3.4通过协作建立动态模型 12.4为类间协作开发集成测试用例 12.5详细设计文档描述 12.6详细设计的评审 12.7项目实践 12.8习题 第13章详细设计中的模块化与信息隐藏 13.1模块化与信息隐藏思想 13.1.1设计质量 13.1.2模块化与信息隐藏思想的动机 13.1.3模块化与信息隐藏思想的发展 13.2模块化 13.2.1分解与模块化 13.2.2结构化设计中的耦合 13.2.3结构化设计中的内聚 13.2.4回顾：MSCS系统设计中的模块化思想 13.3

信息隐藏 13.3.1抽象与信息隐藏 13.3.2信息与隐藏 13.3.3模块说明 13.3.4回顾：MSCS系统设计中的信息思想 13.4习题 第14章详细设计中面向对象方法下的模块化 14.1面向对象中的模块 14.1.1类 14.1.2类之间的联系 14.2访问耦合 14.2.1访问耦合的分析 14.2.2降低访问耦合的方法 14.3继承耦合 14.3.1继承耦合的分析 14.3.2降低继承耦合的方法 14.4内聚 14.4.1面向对象中的内聚 14.4.2提高内聚的方法 14.5耦合与内聚的度量 14.5.1耦合的度量 14.5.2内聚的度量 14.6项目实践 14.7习题 第15章详细设计中面向对象方法下的信息隐藏 15.1封装类的职责 15.1.1类的职责 15.1.2封装——分离接口与实现 15.1.3封装实现细节 15.2为变更而设计 15.2.1封装变更／开闭原则 15.2.2多态 15.2.3依赖倒置原则 15.2.4总结 15.3项目实践 15.4习题 第16章详细设计的设计模式 16.1设计模式基础 16.2可修改性及其基本实现机制 16.3策略模式 16.3.1典型问题 16.3.2设计分析 16.3.3解决方案 16.3.4模式实例 16.4抽象工厂模式 16.4.1典型问题 16.4.2设计分析 16.4.3解决方案 16.4.4模式实例 16.5单件模式 16.5.1典型问题 16.5.2设计分析 16.5.3解决方案 16.5.4模式实例 16.6迭代器模式 16.6.1典型问题 16.6.2设计分析 16.6.3解决方案 16.6.4模式实例 16.7项目实践 16.8习题 第五部分 软件构造、测试、交付与维护 第17章软件构造 17.1概述 17.1.1软件构造的定义 17.1.2软件构造是设计的延续 17.2软件构造活动 17.2.1详细设计 17.2.2编程 17.2.3测试 17.2.4调试 17.2.5代码评审 17.2.6集成与构建 17.2.7构造管理 17.3软件构造实践方法 17.3.1重构 17.3.2测试驱动开发 17.3.3结对编程 17.4项目实践 17.5习题 第18章代码设计 18.1设计易读的代码 18.1.1格式 18.1.2命名 18.1.3注释 18.2设计易维护的代码 18.2.1小型任务 18.2.2复杂决策 18.2.3数据使用 18.2.4明确依赖关系 18.3设计可靠的代码 18.3.1契约式设计 18.3.2防御式编程 18.4使用模型辅助设计复杂代码 18.4.1决策表 18.4.2伪代码 18.4.3程序流程图 18.5为代码开发单元测试用例 18.5.1为方法开发测试用例 18.5.2使用MockObject测试类方法 18.5.3为类开发测试用例 18.6代码复杂度度量 18.7问题代码 18.8项目实践 18.9习题 第19章软件测试 19.1引言 19.1.1验证与确认 19.1.2软件测试的目标 19.1.3测试用例 19.1.4桩与驱动 19.1.5缺陷、错误与失败 19.2测试层次 19.2.1测试层次的划分 19.2.2单元测试 19.2.3集成测试 19.2.4系统测试 19.3测试技术 19.3.1测试用例的选择 19.3.2随机测试 19.3.3基于规格的技术——黑盒测试方法 19.3.4基于代码的技术——白盒测试方法 19.3.5特定测试技术 19.4测试活动 19.5测试度量 19.6项目实践 19.7习题 第20章软件交付 20.1安装与部署 20.1.1安装 20.1.2部署 20.2培训与文档支持 20.2.1培训 20.2.2文档支持 20.3项目评价 20.3.1项目评价的原因 20.3.2项目评价的内容 20.3.3项目评价的方法 20.3.4注意事项 20.4项目实践 20.5习题 第21章软件维护与演化 21.1软件维护 21.1.1软件可修改性与软件维护 21.1.2软件维护的类型 21.1.3软件维护的高代价性 21.1.4开发可维护的软件 21.1.5软件维护过程与活动 21.2软件演化 21.2.1演化与维护 21.2.2软件演化定律 21.2.3软件演化生命周期模型与演化活动 21.3软件维护与演化的常见技术 21.3.1遗留软件 21.3.2逆向工程 21.3.3再工程 21.4项目实践 21.5习题 第六部分软件过程模型与职业基础 第22章软件开发过程模型 22.1软件开发的典型阶段 22.1.1软件需求工程 22.1.2软件设计 22.1.3软件构造 22.1.4软件测试 22.1.5软件交付 22.1.6软件维护 22.2软件生命周期模型 22.3软件过程模型 22.4构建—修复模型 22.5瀑布模型 22.6增量迭代模型 22.7演化模型 22.8原型模型 22.9螺旋模型 22.10Rational统一过程 22.11敏捷过程 22.12习题 第23章软件工程职业基础 23.1软件工程职业 23.1.1软件行业的发展 23.1.2软件工程职业的出现 23.1.3软件工程师职业素质 23.2软件工程职业概况 23.2.1知识体系 23.2.2教育体系 23.2.3职业道德规范 23.2.4认证体系 23.2.5行业协会 23.3软件工程的行业标准 23.4习题 附录A软件需求规格说明文档模板 附录B文档注释规范 附录C软件工程道德和职业实践规范（5.2版）的八项原则 附录D连锁商店管理系统（MSCS）相关文档 参考文献

<<软件工程与计算>>

章节摘录

版权页：插图：“配置”是对“形式”的发展，定义了“部件”以及“连接件”之间的关联方式，将它们组织成系统的总体结构。

按照这个模型，[Shaw1996]给出了一个简洁的软件体系结构定义：“一个软件系统的体系结构规定了系统的计算部件和部件之间的交互。

”理解软件高层结构需要注意的第一个内容：连接件是一个与部件平等的单位。

在软件的详细设计中，交互与计算是交织在一起的——过程、对象和模块是第一等级的软件抽象实体，实现交互的程序调用、消息协作、导入／导出等则都是嵌入在第一等级软件抽象实体内部的，处于附属和派生的地位。

而在软件体系结构中，连接件将交互从计算中独立出来进行抽象和封装，这使得实现交互的连接件与部件一样，都是第一等级的元素单位。

理解软件高层结构需要注意的第二个内容：部件与连接件是比类、模块等软件单位更高层次的抽象。就像类既拥有抽象规格，又拥有“数据结构+算法”的具体实现一样，部件和连接件也既拥有抽象规格，又拥有“模块+连接”的具体实现。

9.2.2区分软件体系结构的抽象与实现 要正确理解软件体系结构就必须能够区分软件体系结构的抽象与实现。

以部件、连接件和配置为基本单位组织的模型就是软件体系结构的抽象，基本目的是描述软件系统的整体功能组织，不涉及程序设计语言提供的各种编程机制。

要最终建立软件产品，就必须考虑如何利用编程机制实现抽象的软件体系结构，这需要从部件、连接件、配置等单位向模块、构件、进程等传统单位进行转换。

模块、构件、进程等传统单位是依赖于编程机制的，它们组成的模型就被称为软件体系结构的实现。

软件体系结构设计是先使用抽象机制完成软件系统的总体功能部署，然后再将抽象模型等价转换为实现模型。

这既保证了软件系统的效用和质量（依靠抽象机制），又顺利实现了从总体结构设计到详细设计的过渡（依靠抽象机制向实现的转换）。

例如，[Allen1997]提到了一个简单的字符大写转换系统Capitalize，它将输入字符流中的候选字符转换为大写，并将其他字符转换为小写。

图9—1a就是以部件和连接件抽象规格形式表示的Capitalize体系结构。

split、upper、lower和merge四个部件（图中表现为矩形框），分别进行读取并分割字符、大写转换、小写转换与重新拼合并输出。

连接件是连接这四个部件的管道（图中表现为线），分别是split→upper、split→lower、upper→merge和lower→merge，它们负责完成字符流的传送。

整个结构的布局体现了软件体系结构的总体功能组织，这就是配置。

很明显，图9—1a所描述的软件体系结构模型是无法实现的，因为基本的程序设计语言机制中并没有能够实现数据流传输的手段，而且split、upper、lower和merge四个功能在执行中的先后顺序和衔接也需要规范，因为程序设计语言支持的是顺序结构，不是并行结构。

所以，图9—1a的抽象模型需要被转换为图9—1b的实现模型。

四个过滤器被实现为split、upper、lower和merge四个功能模块。

四个连接件的实现则与其抽象规格之间有较大的差别，它们都被实现为config和i/o library两个模块，其中config模块帮助各功能模块定位自己的输入和输出字符流，i/o library模块帮助各功能模块读和写字符流。

main模块负责控制各模块的执行顺序。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>