

<<软件工程>>

图书基本信息

书名：<<软件工程>>

13位ISBN编号：9787111362739

10位ISBN编号：711136273X

出版时间：2012-1

出版时间：机械工业出版社

作者：沙赫

页数：380

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>>

### 内容概要

本书是软件工程领域的经典著作，被加州大学伯克利分校等180多所美国高校选作教材。本书第8版继续保持了前七版的特色，采用传统方法与面向对象方法并重的方式，全面系统地介绍软件工程的理论与实践，并新增了第10章（第一部分的关键内容）和第18章（新兴技术）两章内容。全书分为两大部分，第一部分介绍软件工程概念，第二部分着重软件工程技术，教师可根据不同教学目的从任一部分开始讲授课程。

本书是高等院校软件工程课程的理想教材，同时也是专业软件开发人员和管理者的理想参考书。

作者简介

作者：(美国)沙赫 (Stephen R.Schach) 译者：邓迎春 韩松 等

## &lt;&lt;软件工程&gt;&gt;

## 书籍目录

出版者的话

译者序

前言

第1章软件工程的范畴

1.1历史方面

1.2经济方面

1.3维护性方面

1.3.1维护的传统和现代观点

1.3.2交付后维护的重要性

1.4需求、分析和设计方面

1.5小组编程方面

1.6为什么没有计划阶段

1.7为什么没有测试阶段

1.8为什么没有文档阶段

1.9面向对象范型

1.10正确看待面向对象范型

1.11术语

1.12道德问题

本章回顾

进一步阅读指导

习题

第一部分软件工程概念

第2章软件生命周期模型

2.1理论上的软件开发

2.2winburg小型实例研究

2.3winburg小型实例研究心得

2.4野鸭拖拉机公司小型实例研究

2.5迭代和递增

2.6修订的winburg小型实例研究

2.7迭代和递增的风险和其他方面

2.8迭代和递增的控制

2.9其他生命周期模型

2.9.1编码-修补生命周期模型

2.9.2瀑布生命周期模型

2.9.3快速原型开发生命周期模型

2.9.4开源生命周期模型

2.9.5敏捷过程

2.9.6同步-稳定生命周期模型

2.9.7螺旋生命周期模型

2.10生命周期模型的比较

本章回顾

进一步阅读指导

习题

第3章软件过程

3.1统一过程

## &lt;&lt;软件工程&gt;&gt;

3.2面向对象范型内的迭代和递增

3.3需求流

3.4分析流

3.5设计流

3.6实现流

3.7测试流

3.7.1需求制品

3.7.2分析制品

3.7.3设计制品

3.7.4实现制品

3.8交付后维护

3.9退役

3.10统一过程的各阶段

3.10.1开始阶段

3.10.2细化阶段

3.10.3构建阶段

3.10.4转换阶段

3.11一维与二维生命周期模型

3.12改进软件过程

3.13能力成熟度模型

3.14软件过程改进方面的其他努力

3.15软件过程改进的代价和收益

本章回顾

进一步阅读指导

习题

第4章软件小组

4.1小组组织

4.2民主小组方法

4.3传统的主程序员小组方法

4.3.1《纽约时报》项目

4.3.2传统的主程序员小组方法的不实用性

4.4主程序员小组和民主小组之外的编程小组

4.5同步-稳定小组

4.6敏捷过程小组

4.7开源编程小组

4.8人员能力成熟度模型

4.9选择合适的小组组织

本章回顾

进一步阅读指导

习题

第5章软件工程工具

5.1逐步求精法

5.2成本-效益分析法

5.3分治

5.4关注分离

5.5软件度量

5.6case

## <<软件工程>>

5.7 case的分类

5.8 case的范围

5.9 软件版本

5.9.1 修订版

5.9.2 变种版

5.10 配置控制

5.10.1 交付后维护期间的配置控制

5.10.2 基准

5.10.3 产品开发过程中的配置控制

5.11 建造工具

5.12 使用case技术提高生产力

本章回顾

进一步阅读指导

习题

第6章 测试

6.1 质量问题

6.1.1 软件质量保证

6.1.2 管理独立

6.2 非执行测试

6.2.1 走查

6.2.2 管理走查

6.2.3 审查

6.2.4 审查与走查的对比

6.2.5 评审的优缺点

6.2.6 审查的度量

6.3 执行测试

6.4 应该测试什么

6.4.1 实用性

6.4.2 可靠性

6.4.3 健壮性

6.4.4 性能

6.4.5 正确性

6.5 测试与正确性证明

6.5.1 正确性证明的例子

6.5.2 正确性证明小型实例研究

6.5.3 正确性证明和软件工程

6.6 谁应当完成执行测试

6.7 测试什么时候停止

本章回顾

进一步阅读指导

习题

第7章 从模块到对象

7.1 什么是模块

7.2 内聚

7.2.1 偶然性内聚

7.2.2 逻辑性内聚

7.2.3 时间性内聚

## &lt;&lt;软件工程&gt;&gt;

- 7.2.4过程性内聚
- 7.2.5通信性内聚
- 7.2.6功能性内聚
- 7.2.7信息性内聚
- 7.2.8内聚示例
- 7.3耦合
  - 7.3.1内容耦合
  - 7.3.2共用耦合
  - 7.3.3控制耦合
  - 7.3.4印记耦合
  - 7.3.5数据耦合
  - 7.3.6耦合示例
  - 7.3.7耦合的重要性
- 7.4数据封装
  - 7.4.1数据封装和产品开发
  - 7.4.2数据封装和产品维护
- 7.5抽象数据类型
- 7.6信息隐藏
- 7.7对象
- 7.8继承、多态和动态绑定
- 7.9面向对象范型
- 本章回顾
- 进一步阅读指导
- 习题
- 第8章可重用性和可移植性
  - 8.1重用的概念
  - 8.2重用的障碍
  - 8.3重用实例研究
    - 8.3.1raytheon导弹系统部
    - 8.3.2欧洲航天局
  - 8.4对象和重用
  - 8.5设计和实现期间的重用
    - 8.5.1设计重用
    - 8.5.2应用框架
    - 8.5.3设计模式
    - 8.5.4软件体系结构
    - 8.5.5基于组件的软件工程
  - 8.6其他设计模式
    - 8.6.1flic小型实例研究
    - 8.6.2适配器设计模式
    - 8.6.3桥设计模式
    - 8.6.4迭代器设计模式
    - 8.6.5抽象工厂设计模式
  - 8.7设计模式的种类
  - 8.8设计模式的优缺点
  - 8.9重用及互联网
  - 8.10重用和交付后维护

## &lt;&lt;软件工程&gt;&gt;

## 8.11可移植性

## 8.11.1硬件的不兼容性

## 8.11.2操作系统的不兼容性

## 8.11.3数值计算软件的不兼容性

## 8.11.4编译器的不兼容性

## 8.12为什么需要可移植性

## 8.13实现可移植性的技术

## 8.13.1可移植的系统软件

## 8.13.2可移植的应用软件

## 8.13.3可移植的数据

## 8.13.4模型驱动结构

## 本章回顾

## 进一步阅读指导

## 习题

## 第9章计划和估算

## 9.1计划和软件过程

## 9.2周期和成本估算

## 9.2.1产品规模的度量

## 9.2.2成本估算技术

## 9.2.3中间cocomo

## 9.2.4cocomo ii

## 9.2.5跟踪周期和成本估算

## 9.3软件项目管理计划的组成

## 9.4软件项目管理计划框架

## 9.5ieee 软件项目管理计划

## 9.6计划测试

## 9.7计划面向对象的项目

## 9.8培训需求

## 9.9文档标准

## 9.10用于计划和估算的case工具

## 9.11测试软件项目管理计划

## 本章回顾

## 进一步阅读指导

## 习题

## 第二部分软件生命周期的 workflow

## 第10章第一部分的关键内容

## 10.1软件开发：理论与实践

## 10.2迭代和递增

## 10.3统一过程

## 10.4 workflow概述

## 10.5软件小组

## 10.6成本-效益分析法

## 10.7度量

## 10.8case

## 10.9版本和配置

## 10.10测试术语

## 10.11执行测试和非执行测试



## &lt;&lt;软件工程&gt;&gt;

10.12模块性

10.13重用

10.14软件项目管理计划

本章回顾

习题

第11章需求

11.1确定客户需要什么

11.2需求流概述

11.3理解应用域

11.4业务模型

11.4.1访谈

11.4.2其他技术

11.4.3用例

11.5初始需求

11.6对应用域的初始理解：msg基金实例研究

11.7初始业务模型：msg基金实例研究

11.8初始需求：msg基金实例研究

11.9继续需求流：msg基金实例研究

11.10修订需求：msg基金实例研究

11.11测试流：msg基金实例研究

11.12传统的需求阶段

11.13快速原型开发

11.14人的因素

11.15重用快速原型

11.16需求流的case工具

11.17需求流的度量

11.18需求流面临的挑战

本章回顾

进一步阅读指导

习题

第12章传统的分析

12.1规格说明文档

12.2非形式化规格说明

12.3结构化系统分析

12.4结构化系统分析：msg基金实例研究

12.5其他半形式化技术

12.6建造实体-关系模型

12.7有穷状态机

12.8petri网

12.9z

12.9.1z：电梯问题实例研究

12.9.2z的分析

12.10其他的形式化技术

12.11传统分析技术的比较

12.12在传统分析阶段测试

12.13传统分析阶段的case工具

12.14传统分析阶段的度量

## &lt;&lt;软件工程&gt;&gt;

12.15软件项目管理计划：msg基金实例研究

12.16传统分析阶段面临的挑战

本章回顾

进一步阅读指导

习题

第13章面向对象分析

13.1分析流

13.2抽取实体类

13.3面向对象分析：电梯问题实例研究

13.4功能建模：电梯问题实例研究

13.5实体类建模：电梯问题实例研究

13.5.1名词抽取

13.5.2crc卡片

13.6动态建模：电梯问题实例研究

13.7测试流：面向对象分析

13.8抽取边界类和控制类

13.9初始功能模型：msg基金实例研究

13.10初始类图：msg基金实例研究

13.11初始动态模型：msg基金实例研究

13.12修订实体类：msg基金实例研究

13.13抽取边界类：msg基金实例研究

13.14抽取控制类：msg基金实例研究

13.15用例实现：msg基金实例研究

13.15.1estimate funds available for week用例

13.15.2manage an asset用例

13.15.3update estimated annual operating expenses用例

13.15.4produce a report用例

13.16类图递增：msg基金实例研究

13.17测试流：msg基金实例研究

13.18统一过程中的规格说明文档

13.19关于参与者和用例更详细的内容

13.20面向对象分析流的case工具

13.21面向对象分析流的度量

13.22面向对象分析流面临的挑战

本章回顾

进一步阅读指导

习题

第14章设计

14.1设计和抽象

14.2面向操作设计

14.3数据流分析

14.3.1小型实例研究：字数统计

14.3.2数据流分析扩展

14.4事务分析

14.5面向数据设计

14.6面向对象设计

14.7面向对象设计：电梯问题实例研究

## &lt;&lt;软件工程&gt;&gt;

- 14.8面向对象设计：msg基金实例研究
- 14.9设计流
- 14.10测试流：设计
- 14.11测试流：msg基金实例研究
- 14.12详细设计的形式化技术
- 14.13实时设计技术
- 14.14设计的case工具
- 14.15设计的度量
- 14.16设计流面临的挑战
- 本章回顾
- 进一步阅读指导
- 习题
- 第15章实现
- 15.1编程语言的选择
- 15.2第四代语言
- 15.3良好的编程实践
  - 15.3.1使用一致和有意义的变量名
  - 15.3.2自文档代码的问题
  - 15.3.3使用参数
  - 15.3.4为增加可读性的代码编排
  - 15.3.5嵌套的if语句
- 15.4编码标准
- 15.5代码重用
- 15.6集成
  - 15.6.1自顶向下的集成
  - 15.6.2自底向上的集成
  - 15.6.3三明治集成
  - 15.6.4面向对象产品的集成
  - 15.6.5集成的管理
- 15.7实现流
- 15.8实现流：msg基金实例研究
- 15.9测试流：实现
- 15.10测试用例选择
  - 15.10.1规格说明测试与代码测试
  - 15.10.2规格说明测试的可行性
  - 15.10.3代码测试的可行性
- 15.11黑盒单元测试技术
  - 15.11.1等价测试和边界值分析
  - 15.11.2功能测试
- 15.12黑盒测试用例：msg基金实例研究
- 15.13玻璃盒单元测试技术
  - 15.13.1结构测试：语句、分支和路径覆盖
  - 15.13.2复杂性度量
- 15.14代码走查和审查
- 15.15单元测试技术的比较
- 15.16净室
- 15.17测试对象时潜在的问题

## &lt;&lt;软件工程&gt;&gt;

- 15.18单元测试的管理方面
- 15.19何时该重实现而不是调试代码制品
- 15.20集成测试
- 15.21产品测试
- 15.22验收测试
- 15.23测试流：msg基金实例研究
- 15.24实现的case工具
  - 15.24.1软件开发全过程的case工具
  - 15.24.2集成化开发环境
  - 15.24.3商业应用环境
  - 15.24.4公共工具基础结构
  - 15.24.5环境的潜在问题
- 15.25测试流的case工具
- 15.26实现流的度量
- 15.27实现流面临的挑战
- 本章回顾
- 进一步阅读指导
- 习题
- 第16章交付后维护
  - 16.1开发与维护
  - 16.2为什么交付后维护是必要的
  - 16.3对交付后维护程序员的要求是什么
  - 16.4交付后维护小型实例研究
  - 16.5交付后维护的管理
    - 16.5.1缺陷报告
    - 16.5.2批准对产品的修改
    - 16.5.3确保可维护性
    - 16.5.4迭代维护造成的问题
  - 16.6面向对象软件的维护
  - 16.7交付后维护技能与开发技能
  - 16.8逆向工程
  - 16.9交付后维护期间的测试
  - 16.10交付后维护的case工具
  - 16.11交付后维护的度量
  - 16.12交付后维护：msg基金实例研究
  - 16.13交付后维护面临的挑战
- 本章回顾
- 进一步阅读指导
- 习题
- 第17章uml的进一步讨论
  - 17.1uml不是一种方法
  - 17.2类图
    - 17.2.1聚合
    - 17.2.2多重性
    - 17.2.3组合
    - 17.2.4泛化
    - 17.2.5关联

## &lt;&lt;软件工程&gt;&gt;

17.3注解  
17.4用例图  
17.5构造型  
17.6交互图  
17.7状态图  
17.8活动图  
17.9包  
17.10组件图  
17.11部署图  
17.12uml图回顾  
17.13uml和迭代  
本章回顾  
进一步阅读指导  
习题  
第18章新兴技术  
18.1面向层面技术  
18.2模型驱动技术  
18.3基于组件技术  
18.4面向服务技术  
18.5面向服务技术和基于组件技术的比较  
18.6社交计算  
18.7web工程  
18.8云技术  
18.9web 3.0  
18.10计算机安全  
18.11模型检查  
18.12目前和未来  
本章回顾  
进一步阅读指导  
附录  
附录a学期项目：巧克力爱好者匿名  
附录b软件工程资源  
附录c需求流：msg基金实例研究  
附录d结构化系统分析：msg基金实例研究  
附录e分析流：msg基金实例研究  
附录f软件项目管理计划：msg基金实例研究  
附录g设计流：msg基金实例研究  
附录h实现流：msg基金实例研究(c++版)  
附录i实现流：msg基金实例研究(java版)  
附录j测试流：msg基金实例研究

## &lt;&lt;软件工程&gt;&gt;

## 章节摘录

版权页：插图：2.4 野鸭拖拉机公司小型实例研究野鸭拖拉机公司在全美国的大部分地区销售拖拉机。该公司曾经要求它的软件部门开发一个新的软件产品，能够处理它的业务的各个方面。

例如，该产品必须能够处理销售、库存以及向销售人员支付佣金，还能提供所有必需的财会功能。

当这个软件产品正在实现的时候，野鸭拖拉机公司买下了加拿大的一家拖拉机公司。

野鸭拖拉机公司的管理层决定，为了省钱，把加拿大的业务并到美国的业务中去。

这意味着该软件完成前要进行修改：1) 必须修改它来处理增加的销售地区。

2) 必须扩展它来处理那些在加拿大有所不同的业务方面，如税费。

3) 它必须扩展以处理两种不同的货币，美元和加元。

野鸭拖拉机公司是一个快速增长的公司，它具有良好的业务前景。

接管加拿大拖拉机公司是一个积极的进展，这很可能带来未来几年更大的效益。

但是，从软件部门的观点来看，购买这家加拿大的拖拉机公司可能是一场灾难。

要不是进行需求、分析和设计的时候考虑到加入未来可能的扩展，加入加拿大销售地区的工作量可能非常大，可能抛弃迄今做的每件事而从零做起会更高效。

原因是改变这个阶段的产品与在该产品的生命周期的后期（见图1-5）修改它是相似的。

扩展软件处理与加拿大市场有关的方面以及加拿大货币可能同样困难。

即使软件是经过深思熟虑的，并且最初的设计确实是可扩展的，设计出的拼补在一起的产品不可能如最初就设计成满足美国和加拿大业务那样结合得好。

这会给将来的维护带来严重的隐患。

野鸭拖拉机公司软件部是移动目标问题的牺牲品。

就是说，在软件正在开发的时候，需求改变了。

它与改变的原因非常值得无关。

事实是，接管加拿大公司对于正在开发的软件质量是非常有害的。

在某些情况下，移动目标的原因不太良好。

有时一个组织内有权利的高层管理人会不断地改变关于正在开发的软件产品的功能的想法。

在另一种情形下，是特性的蔓延，即连续地向需求中加入小的甚至是琐碎的特性。

但是，不管是什么原因，频繁的改变，不管看起来多么微不足道，对于一个软件产品的健康状况都是有利的。

重要的是一个软件产品设计成一套尽可能独立的组件，以使对于该软件某个部分的改变不在代码明显无关的部分引入差错，称为退化（性）差错（regression fault）。

当做大量改变的时候，结果是在代码内部引起联动。

最后，存在许多的联动实际上都会引入一个或多个退化差错。

在这种时候，唯一能做的就是重新设计整个软件产品并重新实现它。

遗憾的是，对移动目标问题目前还没有解决办法。

对需求的积极性改变而言，业务不断增长的公司总是要改变的，这些改变必须反映在公司的重要任务软件产品中。

对于消极性改变，如果个人要求做这些改变的决定有较大可能，便无法阻止对正在进行的实现做修改，也无法阻止对该软件产品将来的维护性的损害。

<<软件工程>>

编辑推荐

《软件工程:面向对象和传统的方法(原书第8版)》是计算机科学丛书之一！

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>