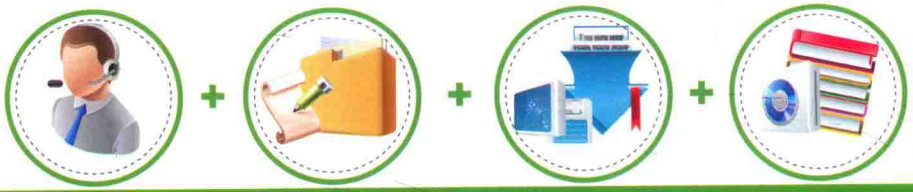
全国高等院校应用型创新规划教材 计算机系列

软件工程
与设计模式

白文荣 主编



- 以基础理论—实用技术—实训为主线
- 按照教与学的实际需要取材谋篇
- 精心设置有“小型案例实训”，旨在培养学生的实践能力
- 配备丰富的免费教学资源——电子教案、习题答案和项目开发实例源代码

清华大学出版社

全国高等院校应用型创新规划教材·计算机系列

软件工程与设计模式

白文荣　主编

清华大学出版社
北京

内 容 简 介

本书是作者在多年从事软件工程、软件设计模式课程教学实践基础上编写的。全书共分为 8 章，通过大量的实例，介绍了实用软件工程学的原理及设计模式的相关知识，根据软件开发“工程化”思想，系统地讲授了软件工程学、软件设计过程、23 种先进的设计模式、软件测试方法、软件项目管理、应用技术和实用工具等相关知识。全书采用社会所需实际案例为基线，以案例、项目式教学思路贯穿始终，根据需要安排了多个任务和子任务，读者可以通过实践掌握课程所学内容。书后配有适量的思考题和练习题，使读者能够在学习过程中提高操作能力和实际应用能力。

本书可作为高等院校学生学习软件工程、软件设计模式、软件体系结构设计等课程的教材，也可以作为读者自学的参考书。

图书在版编目(CIP)数据

软件工程与设计模式/白文荣主编. —北京：清华大学出版社，2017
(全国高等院校应用型创新规划教材 • 计算机系列)
ISBN 978-7-302-45714-5

Ⅰ. ①软…　Ⅱ. ①白…　Ⅲ. ①软件工程—高等学校—教材　Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2016)第 288798 号

责任编辑：秦　甲　杨作梅
封面设计：杨玉兰
责任校对：吴春华
责任印制：王静怡
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn, 010-62791865
印 装 者：北京市人民文学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张：**24.75　　**字　数：**599 千字
版　　次：2017 年 1 月第 1 版　　**印　次：**2017 年 1 月第 1 次印刷
印　　数：1～2500
定　　价：49.00 元

产品编号：070549-01

前　言

21 世纪是信息社会，信息技术已经渗透到社会的各行各业。随着计算机应用技术的不断发展，软件工程学也渗入软件研发的各个环节中。实用软件工程是一门将理论和知识应用于实践的工程，它借鉴了传统工程的原则和方法，讲解了常用的 23 种设计模式，以求高效地开发高质量软件。近年来，大多数高等院校，无论是理工科还是文科专业都将软件工程作为计算机应用技术类课程的必修课或选修课。

软件工程是软件开发组织根据所要开发的软件特点及项目自身的特点，选择适合的软件设计模式，把各种软件工程学原理的特性和软件设计模式有机地结合起来，充分利用它们的优点，回避缺陷，有效地提高软件质量的过程。

本书是软件开发方法体系的完整体现，有别于传统软件工程学，增加了许多实际软件开发过程中需要的实用方法技术，填补了传统软件工程的设计薄弱环节。

全书共分为 8 章，各章的主要内容安排如下。

第 1 章主要介绍软件工程学的基本概念、理论和基础知识。

第 2 章主要介绍软件设计过程基本原理、软件设计建模工具 UML，以及软件设计环境 Visio、PowerDesigner、Violet 等。

第 3 章主要介绍 23 种先进的设计模式，体现 23 种设计模式在软件开发过程中的重要设计地位和作用。

第 4 章主要介绍在面向对象程序设计语言 Java 中实现 23 种设计模式的方法和实践。

第 5 章主要介绍面向对象技术。面向对象技术是一种以对象为基础，以事件或消息驱动对象执行相应的消息处理函数的程序设计技术。

第 6 章主要介绍各种行之有效的软件测试方法和技术。

第 7 章主要介绍 23 种设计模式在实际应用中的原则和分类。

第 8 章主要介绍软件工程控制活动中的质量保证、配置管理和项目管理的基本知识。

本书以学习、应用为目的，以案例贯穿始终，系统地讲授了软件工程学和软件设计模式，各章均以项目分配任务形式编排，有助于提高学生的实操和实际应用能力。

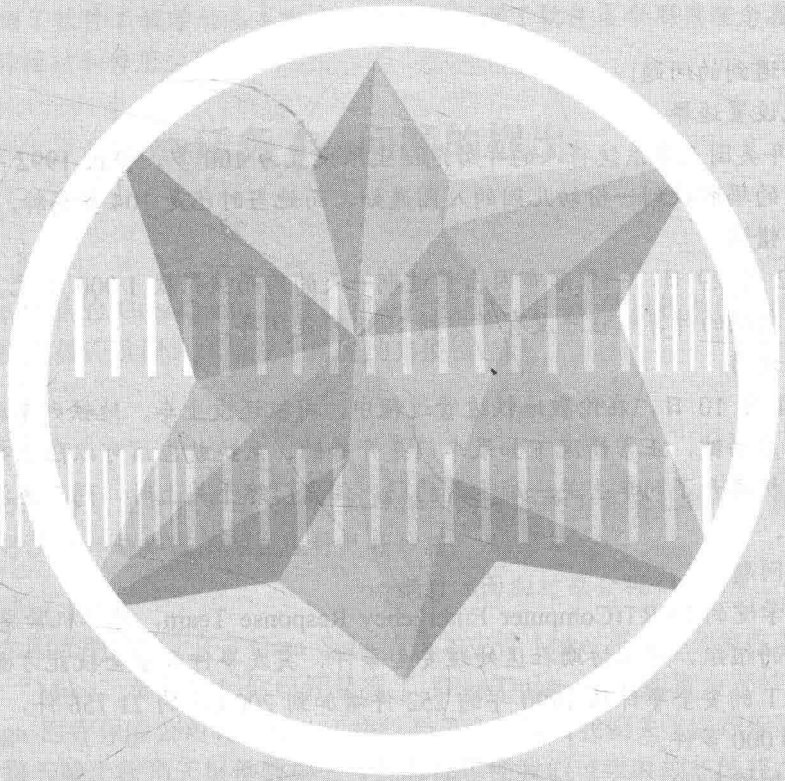
本书由白文荣主编，在本书策划和编写的过程中，得到了清华大学出版社的大力支持，在此表示衷心的感谢。

由于作者水平有限，书中难免存在错误和不足之处，敬请广大读者批评指正。

为了方便教师教学和学生自主学习，本书配有电子教案、案例源代码、安装软件等，若有需要，可从清华大学出版社网站下载。

编　者

目录



项目 1

软件工程概述

项目导入

看看下面遇到的问题:

1) 系统设置边界

在 1900 年美国人事系统将人的年龄范围上限设置为 100 岁，导致 1992 年，来自明尼苏达州怀俄明的玛丽收到一份幼儿园的入园通知，而她当时已是 104 岁高龄。

2) 闰年错误

1988 年 2 月 29 日，一家超市因出售过期一天的肉而被罚款 1 000 美元。因为在肉的标签上打印保质期的计算机程序没有考虑到 1988 年是闰年。

3) 接口误用

1990 年 4 月 10 日，在伦敦地铁运营过程中，司机还没上车，地铁列车就驶离车站。当时司机按了启动键，正常情况下如果车门是开着的，系统就应该可以阻止列车启动。当时的问题是司机离开了列车去关一扇卡着的门，但当门终于关上时，列车还没有等到司机上车就开动了。

4) 安全问题

软件工程学院的 CERT(Computer Emergency Response Team，计算机紧急反应小组)是一个政府资助的组织，用来协助社区处理安全事件、突发事件和安全技能方面的问题。美国报道的 CERT 的安全事件从 1990 年的 252 件增加到 2000 年的 21 756 件，而到 2001 年已增加到了 40 000 多件。

5) 拖延和超支(1)

在 1995 年，由于新丹佛尔国际机场自动行李系统的错误，造成旅客行李箱的损坏。机场则被迫推迟 16 个月再开放，且大部分采用手工行李系统，产生 32 亿美元超支。

6) 拖延和超支(2)

2002 年的 Swanick 空运控制系统，包括英格兰和威尔士全部空运线路。在系统交付时，已延期 6 年且严重超支(实际花费 6.23 亿英傍，原计划花费 3.5 亿英镑)，其中两次主要的系统升级是在运输操作员培训已经开始后才交付的。

7) 按期交付

1984 年，经过 18 个月的开发，一个耗资 2 亿美元的系统交付给了威斯康星州的一家健康保险公司。但是该系统无法正常工作，只好追加了 6 000 万美元，又花了 3 年才解决了问题。

8) 不必要的复杂性

麦克道尔道格拉斯的 C-17 货机因为控制系统的软件问题，而超支 5 亿美元。C-17 含有 19 台机载计算机，80 个微处理器以及 6 种不同的编程语言。

小结：上述各种失误的产生都与软件问题有关。有时，开发者没有考虑到偶发事件(如一个人的年龄超过 100 岁，影响保质期的闰年)。有时，开发者没有考虑到系统的异常情况(如接口误用)。还有些系统失误的情形是由于管理失误(拖延和超支交付、按期交付不正确的系统以及不必要的复杂性)造成的。

项目分析

本项目通过医疗自动报价系统错误案例分析，讲解了软件工程学的重要性、软件设计

的必要性，介绍了软件工程学的基本概念、原理，强调了软件生命周期理念在软件开发过程中的重要设计地位和作用。

任务 1　问题的提出

任务要求

根据 Mike 团队的 Giga-Quote 项目内容分析整个项目推进的过程及出现的错误问题，搜集资料，分析人员、过程、产品三个角度的典型错误。

知识储备

业余软件工程师总在寻找奇迹——用某种惊人的方法或工具来让软件开发变得轻而易举，但职业软件工程师都知道不存在这种灵丹妙药。

——摘自《面向对象分析和设计》，Grady Booch

4 月份一个阳光明媚的上午，Giga Safe 的技术主管 Mike 正在办公室吃着午餐，看着窗外的美景。“Mike，恭喜你！你已经获得了 Giga-Quote 项目的资金了！”他是 Bill，Mike 所在 Giga 医疗保险公司的老板，“执行委员会很欣赏我们关于医疗保险自动报价的设想，也欣赏每天晚上将当天报价数据上传到总部以便我们可以时刻在线获得最新销售线索的想法。现在，我还有个会，过一会儿我们再细谈，你的项目建议书真是太棒了！”Mike 几个月前就为 Giga-Quote 项目写了一个项目建议书，但他的项目建议书只是单机版软件，不具备与总部通信的能力。哦，也好，这倒给了他在现代 GUI 环境下领导开发客户服务器项目的机会——这是他向往已久的事。按他的项目建议书，他几乎有 1 年的时间来做这个项目，应该有足够的时间加入一些新功能。Mike 拿起电话，拨了他妻子的电话号码：“Honey，让我们今晚到外面共进晚餐庆祝……”第二天早上，Bill 约见 Mike 讨论这个项目：“OK，Bill，什么时候开始？这个项目似乎并不像以前我写的项目建议书那样。”Bill 感到不自在，Mike 没有参加项目建议书的修改，但那是因为没有时间让他参与，执行委员会一听就决定接受这个建议。“执行委员会欣赏构建医疗保险自动报价系统这个设想，但他们想将地区报价数据自动传到主机系统中。而且他们想在明年 1 月份新费率生效前，系统就能准备就绪。他们将软件完成日期从明年 5 月 1 日提前到今年 11 月 1 日，将项目工期压缩至 6 个月。”Mike 估计这项工作将需要 12 个月，他认为没有多大可能在 6 个月内完成，他告诉 Bill：“执行委员会加入了一块很大的通信需求，却将计划时间从 12 个月砍到 6 个月了吗？” Bill 耸耸肩：“我知道这是一个挑战，但你是具有创造能力的，我认为你能够实现。他们批准了你提出的预算，加入数据通信功能也并不是那么难。你要求 36 个人月，没有问题。在这个项目上，你可以雇佣任何你想要的人，也可以扩大项目组规模。”Bill 告诉他去找些开发人员谈一谈，找出一条可以按时交付软件的方法。Mike 找到了另一个技术领导 Carl，他们寻找缩短计划的方法。Carl 问：“你为什么不用 C++和面向对象的设计方法？它可以比 C 有更高的效率，这样可以将项目计划缩短 1～2 个月。”Mike 认为有道理，Carl 也知道一种报表生成工具，据悉可以削减一半的开发时

间，这个项目有许多报表，所以，这两项改变可以将项目计划缩短到 9 个月。由于他们拥有更新、更快的硬件设备，这样可以再削减 3 周时间。如果能够雇佣到顶尖的开发人员，他们可以将项目计划缩短到大约 7 个月时间，这已经足够接近了。Mike 给 Bill 带回了他的发现。“看，”Bill 说，“将计划压缩到 7 个月是不错，但还不足以满足项目要求，执行委员会要求 6 个月是最后的期限。他们不给我们选择的余地，我可以给你最新的硬件设备，但你和你的项目组必须找到一些方法或者加班加点拼命工作，将项目计划压缩到 6 个月以内。”

考虑到起初的估计仅仅是粗略的预测，Mike 认为，在 6 个月内完成项目是可能实现的。“OK，Bill，我将在这个项目中雇佣 3 个合同制的顶尖高手，也许我们能够找到具有将数据从 PC 上传到主机的经验的人。” 到了 5 月 1 日，Mike 已经组建了项目组，Jill、Sue 和 Tomas 是公司内部固定的开发成员，但他们无法完成一些特定任务。Mike 找了 2 个合同制成员 Keiko 和 Chip，Keiko 有开发主机和 PC 之间通信接口的经验，Chip 是 Jill 和 Tomas 面试的，建议不宜雇佣，但 Mike 急于用人，Chip 有通信经验，并能够迅速到岗，所以 Mike 还是雇用了他。

在项目组的第一次会议上，Bill 向项目组阐明了 Giga-Quote 项目对 GigaSafe 公司的战略意义：“公司的顶级人物时刻关注着我们，如果项目取得成功，我们会获得丰厚的奖赏。”他保证说他将信守承诺。

Bill 鼓气激励之后，Mike 与项目组成员坐下，布置项目计划。执行委员会或多或少已经提出了一些项目功能要求，其余的功能说明应在接下来的 3 周内完成，然后，他们花 6 周时间进行设计，余下的 4 个月进行构建与测试。粗略估计整个产品大约会有 3 万行 C++代码。周围的每个人都点头称是。他们雄心勃勃，但不知道以后会发生什么。第 2 周，Mike 约见了测试负责人 Stacy，她说他们应该不晚于 9 月 1 日提交测试版本，并于 10 月 1 日交付功能完备并通过测试的版本，Mike 同意了她的计划。项目组很快完成了需求分析报告并进入了设计阶段，似乎很好地发挥了 C++的功能优势。6 月 15 日，项目组提前完成设计工作，开始了疯狂的编码，以满足 9 月 1 日发布第一个测试版本的要求。项目进展并非一帆风顺，Jill 和 Tomas 都不喜欢 Chip，Sue 也抱怨 Chip 不让任何人靠近他的代码，Mike 将这些归结于由于人们长时间工作所导致的个性冲突。然而到了 8 月初，他们报告说只完成了 85%～90%的工作。8 月中旬，保险核算部门发布了下一年度的费率，项目组发现他们的系统必须进行调整才能完全适用于新的费率结构。新的费率方法要求有提出问题的功能，如提出像锻炼习惯、饮食习惯、吸烟习惯、娱乐活动及其他一些以前不包括在费率计算公式之内的因素之类的问题，他们认为 C++的特性可以让他们不受这些变化的影响，他们已经把在费率表中插入一些新数据这样的问题提前考虑进去了。

但是，他们必须改变输入对话框、数据库设计以及数据存取对象和通信对象，以适应新的结构。由于项目组处于设计修改的混乱之中，Mike 告诉 Stacy 他们可能比预计推迟几天交付第 1 个测试版本。

9 月 1 日项目组没有准备好测试版本，Mike 向 Stacy 保证再有 1～2 天就可以交付了。转眼间数周过去了，预计 10 月 1 日交付测试通过的完全版本的日期到来并过去了，项目组还是没能提交第 1 个测试版本。

Stacy 和 Bill 召开了一个讨论项目计划的会议，“我们还没有从开发组中拿到测试版

本，”她说，“原计划我们 9 月 1 日拿到第 1 个测试版，由于到现在我们还没有拿到，他们至少已经落后于计划整整 1 个月了，我想他们一定遇到麻烦了。”“他们确实遇到麻烦了，”Bill 说，“让我们与项目组谈谈，我已经承诺，11 月 1 日让 600 个代理点都能拿到软件，我们必须在新费率执行前及时拿出最后版本。”

Bill 召开了项目组会议，“这是一个富有朝气的团队。你们应该兑现你们的承诺，”他说，“我不知道什么地方出了问题，但我希望每个人能够努力工作，并按时交付产品。你们还可以得到奖金。但你们必须为之努力工作，从现在起到软件完成，我要求你们每周工作 6 天，每天工作 10 小时。” Jill 和 Tomas 抱怨 Mike 不必像对待孩子一样对待他们，但他们同意按 Bill 要求的时间工作。项目组将计划推迟了 2 周，承诺 11 月 15 日交付功能完整的版本，在明年 1 月 1 日新费率生效前，还能有 6 周的测试时间。4 周后，项目组于 11 月 1 日发布了第 1 个测试版本，然后开会讨论遗留的问题。

Tomas 负责开发报表生成部分，他碰到了一个问题，“报价汇总表包括一张简单的柱状图，我使用报表生成器生成一张柱状图时，它只能单独地放在一页上，而销售部门的要求是将文字和柱状图放在同一页，我已经计划在报表的文字部分添加柱状图联想标记，引出报告的柱状图。这里有明显的标志，在第 1 版发布以后我会再来仔细研究处理它。”

Mike 回答：“我看不到所谓‘以后’是什么时候，Bill 已经清楚地提出了要求，我们必须拿出产品，没有时间使代码尽善尽美，现在没有任何推延的可能，把标志部分做出来!”此时，Chip 报告说他的通信编码已经完成了 95%，还在继续进行，有许多需要测试的东西。Mike 发觉了 Jill 和 Tomas 鄙夷的眼神，但他决定不予理睬。直到 11 月 15 日，项目组仍在努力工作，14 日和 15 日几乎工作得通宵达旦，但他们还是没能完成 11 月 15 日应当交付的版本。到了 16 日上午，项目组已经筋疲力尽了。最感丧气的是 Bill，Stacy 已经告诉了他 11 月 15 日之前开发组没能交付功能完整的测试版本。

就在上周，另一个项目经理 Claire 了解到项目的进展后说，他们将不会按时交付测试版本。Bill 认为 Claire 极端保守，他不喜欢她。他向执行委员会汇报说项目在正常进行，他保证项目组正按计划推进最后的版本。Bill 告诉 Mike 召集项目组会议，他照办了。项目组看起来就像打了败仗一样，一个半月每周 60 小时的工作几乎压垮了他们。当 Mike 问什么时间能交付测试版本时，Tomas 说，“今天我的代码可以脱手，可以称之为‘功能完整’，但我可能还需要 3 周的整理工作。”Mike 问 Tomas 的“整理工作”是什么意思，“我还没有将公司的标志放到每页上，我还没有将代理点的名称和电话号码打印到每页的下角，还有一些其他类似这样的事情。我认为已经做完了 99%。” Jill 也说：“我也确实没有 100%完成，我以前的项目组经常叫我做一些技术支持工作，我大约每天得为他们工作 2 个小时。另外，到现在我还不知道要给代理点提供在报表中加入它们名称和电话的功能，我还没有设计输入这些数据的窗口，同时，我还得做一些处理这些窗口的工作，我以为在‘完成功能完整’的里程标志时间点提供的版本不需要这些功能。”

现在，Mike 也感到丧气了。Mike 围绕着桌子一个一个地问每个人，他们是否可以在 3 周内完成所承担的工作，每个人都表示，如果努力工作，他们可以完成。

经过长时间、令人不舒服的讨论之后，Mike 和 Bill 同意将项目计划顺延 3 周到 12 月 5 日，同时要求项目组每天工作长达 12 小时，而不再是 10 小时。计划的修改意味着测试和区域代理点的培训必须同步进行，这是 1 月 1 日发布软件的唯一办法。Stacy 抱怨没有

给 QA 足够的时间测试软件，但 Bill 驳回了她的说法。12 月 5 日中午前，Giga-Quote 项目组将功能完整的 Giga-Quote 程序提交测试。剩下的工作就是需要一个长时间的休息，从 9 月 1 日以来，他们几乎一直在工作。两天后，Stacy 发布了第 1 个问题报告，该死的报告打破了轻松的休息。两天中，测试组在 Giga-Quote 程序中发现了 200 多个问题，包括必须处理的一类严重错误 23 个。“我看不到任何于明年 1 月 1 日将软件发给各代理点的希望，”她说，“测试组可能需要较长时间重新编写测试用例，并且，我们每小时还在发现新的错误。”

第二天上午 8 点，Mike 召开了项目组成员会议。开发人员都脾气暴躁，火气十足，他们说虽然存在严重错误，但报告中的许多问题并不是真正的错误，有些纯粹是操作上的误解，Tomas 指出，例如第 143 号错误：“测试报告中的第 143 号说在报价汇总表中，柱状图要求在页面的右侧而不应放在页面左侧，这很难说是一类严重错误，这是典型的测试问题反应过度。”

Mike 分发了测试问题报告，他要求开发人员检查测试中出现的问题，并估算修正每个错误所需要的时间。当项目组下午再次碰面时，消息不太好，“现实一点，”Sue 说，“我估计处理已经发现的问题至少需要 3 周的时间，加上我还要完成数据库一致性工作，从现在起，我总共需要 4 周时间。”

Tomas 已经将第 143 号错误发回到测试组，申请将错误级别从一类错误改为三类错误——“修饰性错误”。测试人员回答说，Giga-Quote 所提交的汇总报告必须与主机的政策更新程序所产生的类似报告的形式相似，它们也应该与公司使用多年的市场印刷材料相符合，公司 600 多个代理点已习惯将销售柱状图放在页面的右侧，所以它必须放在右侧，因此，将其归为一类严重错误。

Tomas 说，“要将柱状图放在右侧，我必须得从头重写那个报表，那意味着我必须自己编写低层代码才能实现报表要求的图文格式。”Mike 小心翼翼地问，那样做粗略估计需要多少时间，Tomas 说至少得花 10 天的时间，在知道确切时间前，还需要仔细研究一下。那天回家前，Mike 告诉 Stacy 和 Bill，项目组整个假期都要工作才能在 1 月 7 日处理完已发现的所有错误。Bill 说他真的希望这是最后一次了，在开始远在去年夏天就已经计划的为期 1 个月的 Caribbean 休假前，他批准了项目计划顺延 4 周。接下来的 1 个月时间里，Mike 又把项目组召集在了一起。4 个多月来，项目组成员拼命地工作，他认为已经无法再进一步逼他们了。他们每天要在办公室待上 12 个小时，但他们会花许多时间看杂志、支付账单、电话聊天，一旦问起他们处理碰到的错误需要多长时间时，他们就火冒三丈。他们每处理一个错误，测试人员就会发现 2 个新的错误，一些本来估计花几分钟就可以解决的问题由于牵扯项目各方，变成需花数天时间才能解决。很快他们意识到他们无法在 1 月 7 日前处理完所有错误。

1 月 7 日，Bill 休假返回，Mike 告诉他开发组还需要 4 周时间，“糟糕透了。”Bill 说：“我的 600 多个区域代理点已经对你们这些搞计算机的家伙愤怒不已了。执行委员会正在考虑取消这个项目。无论如何，必须在 3 周内处理完所有的问题。”

Mike 召开项目组会议，讨论解决办法。Mike 告诉了大家 Bill 的态度，要求他们给出最终可以发布产品的时间，是仅仅 1 周，还是 1 个月。大家沉默不语，没人冒险猜测什么时间能够发布最终产品，Mike 不知该如何向 Bill 汇报。会后，Chip 告诉 Mike，他已经接

受了另外一家公司的合约，合同于 2 月 3 日开始。Mike 开始感到项目可能会被取消。

Mike 找到了负责编写 PC—主机通信模块中主机一侧程序的编程员 Kip，让他帮助处理本项目中 PC 一侧的通信代码。经过与 Chip 所编代码 1 周的“鏖战”，Kip 认识到程序中存在一些深层缺陷，这意味着程序很难正确运行。Kip 被迫重新设计，并重新编写 PC—主机通信中的 PC 侧程序。

2 月中旬，Bill 由于一直忙于开会，决定由 Claire 来继续监控 Giga-Quote 项目的进展。星期五，Claire 约见了 Mike。“这个项目已经失控了，”她说，“我一直没有从 Bill 处获得有关几个月来可靠的项目估算计划，这是 6 个月的项目，现在已经拖延 3 个月了，还没有结论。我已经看了问题分析报告，而且，项目组还没有解决这些问题，你们一直长时间工作，但收效欠佳。我要求你们周末休息，然后，我要你们做一个详细的、一步一步的包括所有事情的报告，我所说的所有事情是指项目剩下的事情。我不要求你们强制将项目按人为的计划执行，我想知道是否还需要另外 9 个月时间，下个星期二给我一份项目结束的报告。报告不必那么讲究，但必须是完整的。”开发组成员很高兴有了周末，并有足够的精力投入下周准备报告的工作中。星期二，报告准时出现在 Claire 的桌子上。她与已看过问题分析报告的软件工程顾问 Charles 一道分析这篇报告。Charles 建议项目组的重点应放在少数有问题倾向的模块上，对所有已处理的错误迅速开始设计和编码检查，项目组按正常作息时间工作，以确保他们能准确地找到问题，并按要求正确地解决。3 周后，也就是 3 月的第一周，所有发现的错误第一次全部被剔除，项目组士气高涨，项目进展稳定。到 5 月 15 日，顾问建议，软件可以交付进行整体测试和可靠性测试了。由于 GigaSafe 半年费率增长在 7 月 1 日生效，Claire 确定软件正式发布的日期是 6 月 1 日。

Giga-Quote 程序按计划于 6 月 1 日发布到各区域点，GigaSafe 向开发组每位成员颁发了 250 美元的奖金，以感谢他们辛勤的工作。几周以后，Tomas 要求长期休假，Jill 也跳槽到另外一家公司去了。

任务实施

最终 Giga-Quote 产品花了 13 个月才得以交付，而不是计划的 6 个月，时间超过计划 100%。开发人员的工作量，包括加班，总计 98 个人月，超过计划 36 人月的 170%。最终的产品不包括空行和注释行大约 4 万 C++代码行，超出 Mike 当初的粗略估计量 33%。作为分发到 600 多个地点的应用产品，Giga-Quote 是个介于商业软件产品和封装商品软件产品之间的混合体，这种类型和规模的产品正常应该在 11.5 月、使用 71 个人月完成，这个项目在这两个方面均超过了平均值。

一些无效的开发实践经常被许多人选用，这些可预测会带来坏结果的行为成为典型错误。然而大多数这类错误都具有表面诱惑力。以下为从人员、过程、产品三个角度的典型错误分析。

1)　人员方面的问题

- 管理层挫伤人员的积极性。
- 管理层对有问题的员工失控。
- 个人英雄主义。

- 项目后期加入人员。
- 办公环境拥挤嘈杂。
- 开发人员与客户之间产生摩擦。
- 不现实的预期。
- 缺乏有效的项目支持。
- 缺乏各种角色的齐心协力。
- 缺乏用户介入。
- 充满想象。

2) 过程方面的问题

- 过于乐观的计划。
- 缺乏足够的风险管理。
- 承包人导致的失败。
- 缺乏详细、具体的计划。
- 在压力下放弃计划。
- 在模糊的项目前期浪费时间。
- 前期活动不合要求。
- 设计低劣。
- 缺少质量保证措施。
- 缺少管理控制。
- 太早或过于频繁的集成。
- 项目估算时遗漏了必要的任务。
- 追赶计划。
- 鲁莽编码。

3) 产品方面的问题

- 需求的镀金。
- 功能蔓延。
- 开发人员的镀金。
- 产品功能的随意变更。
- 研究导向的开发。

任务 2　软件工程概述

任务要求

在该任务中，要求学生掌握软件危机、软件、软件工程的基本概念及原理等理论要点，从学科角度、社会需求角度理解学习软件工程学的必要性。

知识储备

软件工程是一门将理论和知识应用于实践的工程，它借鉴了传统工程的原则和方法，

以求高效地开发高质量软件。除了工程科学，软件工程还综合应用了计算机科学、数学和管理科学。计算机科学和数学用于构造模型与算法，工程科学用于制定规范、分析设计、评估成本及确定权衡，管理科学用于计划、资源、质量和成本的管理。软件工程这一概念，主要是针对 20 世纪 60 年代“软件危机”而提出的。它首次出现在 1968 年的 NATO (北大西洋公约组织)会议上。自这一概念提出以来，围绕软件项目，开展了有关开发模型、方法以及支持工具的研究。其主要成果有：提出了瀑布模型，开发了一些结构化程序设计语言(如 PASCAL 语言，Ada 语言)、结构化方法等。并且，围绕项目管理提出了费用估算、文档复审等方法和工具。综观 20 世纪 60 年代末至 80 年代初，其主要特征是，前期着重研究系统实现技术，后期开始强调开发管理和软件质量等。自 80 年代“软件工厂”这一概念提出以来，主要围绕软件过程以及软件复用，开展了有关软件生产技术和软件生产管理的研究与实践。其主要成果有：提出了应用广泛的面向对象语言以及相关的面向对象方法，大力开展了计算机辅助软件工程(CASE)的研究与实践等。

在计算机软件生产的发展过程中，产生了软件危机，为了解决软件危机，产生了软件工程。下面从软件危机入手，介绍软件工程学的基本概念。

1. 软件危机

每天全世界都有许多软件存在超出预算、推迟交付时间、带有残存错误就交付使用、不满足用户需求等诸多问题。软件设计上的欠妥和错误为这个文明社会带来了很多不愉快，甚至是灾难性的结局。故在计算机软件的开发和维护过程中所遇到的一系列严重问题称为软件危机。其实软件危机并不是指问题严重到危机的程度，只是为了引起人们对其的重视，才称为软件危机。

产生软件危机的根本原因是软件本身的特点。软件是计算机系统中的一个逻辑部件，它具有抽象性、可复制性、不会磨损性、依赖性、开发效率低、开发费用高等特点。产生软件危机的原因如下。

(1) 软件本身的特点(如软件规模庞大)导致开发和维护困难。

(2) 软件开发的方法不正确。

(3) 开发人员与管理人员重视开发而轻视问题的定义和软件维护。

(4) 软件开发技术落后。

(5) 软件管理技术差。

为了解决软件危机，既要有技术措施(好的方法和工具)，也要有组织管理措施。软件工程正是从技术和管理两方面来研究如何更好地开发和维护计算机软件的。

2. 软件工程学的基本概念

要知道什么是软件工程，首先要知道什么是软件。

1)　软件

在信息处理和计算机领域，一般认为软件是计算机程序、各种相关的文档和数据的集合；实际上，计算机软件是指计算机程序、方法、规则、相关的文档资料以及在计算机上运行程序时所必需的数据。

软件=程序+数据+文档