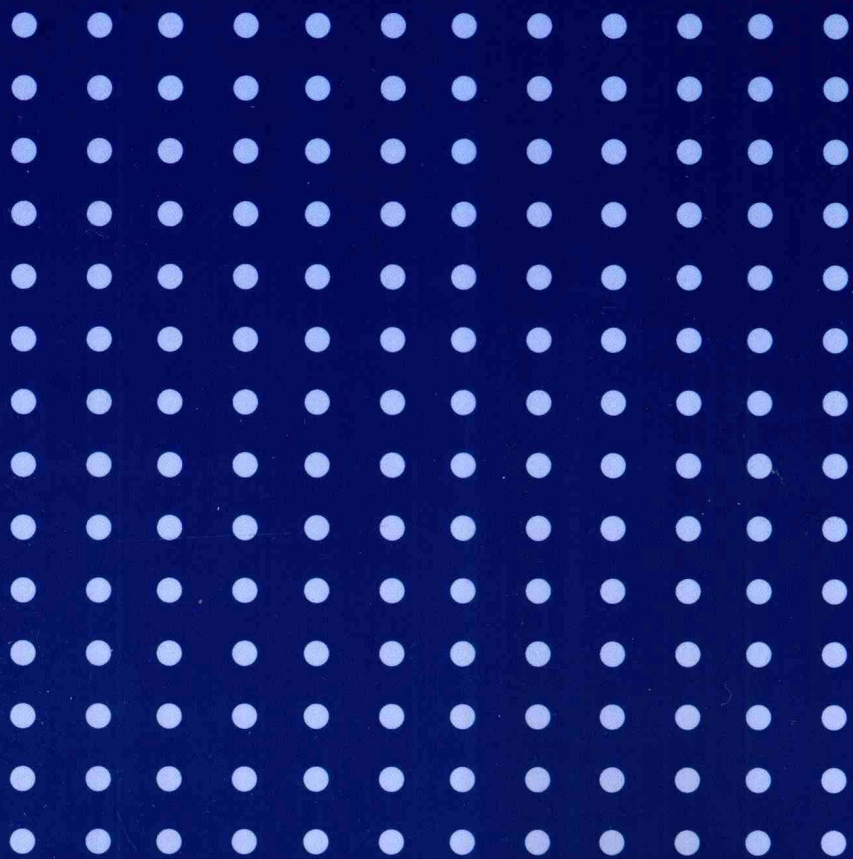


全国优秀畅销书  
全国高校出版社优秀畅销书

张海藩 编著

# 软件工程



清华大学出版社

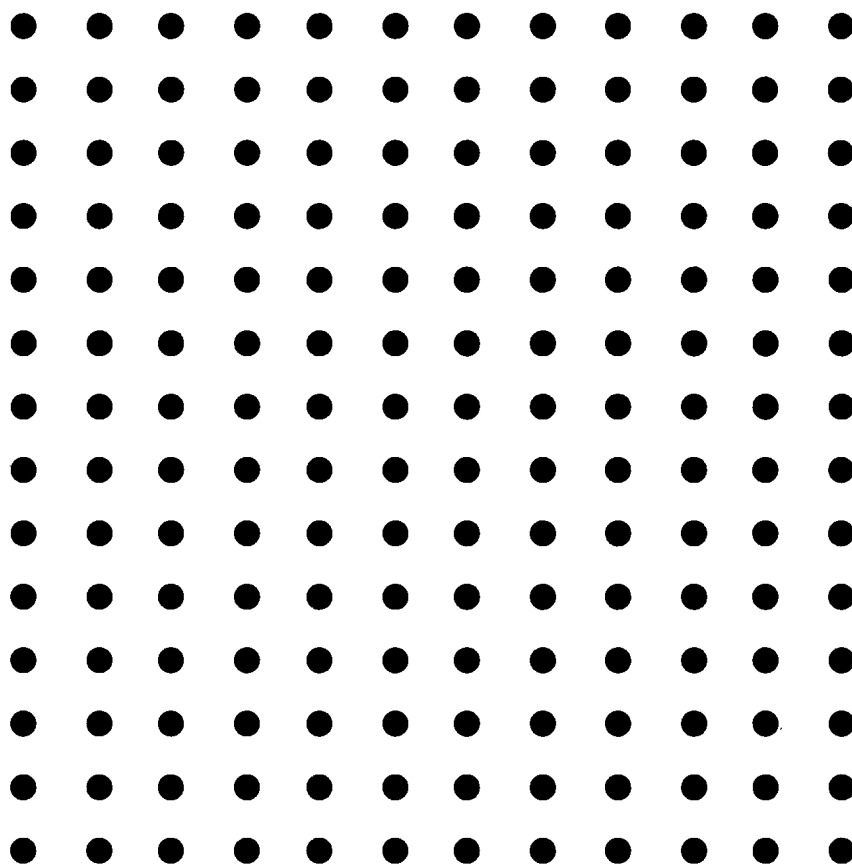


全国优秀畅销书

全国高校出版社优秀畅销书

# 软件工程

张海藩 编著



清华大学出版社

北京

## 内 容 简 介

为了满足“软件工程”课程学时较少的高等学校对一本适用教材的迫切需求,作者对先后荣获了全国普通高等学校工科电子类专业优秀教材一等奖和全国优秀畅销书奖,并被评为北京高等教育精品教材的《软件工程导论》作了精心改编,保留了原书中较重要、较新颖的内容,删除或简化了一些较陈旧或较次要的内容,写出了《软件工程》。

本书全面系统地讲述了软件工程的概 念、原理和典型的方法学,并介绍了软件项目的管理技术。本书正文共 13 章,第 1 章是概述,第 2 章至第 8 章顺序讲述软件生命周期各阶段的任务、过程、结构化方法和工具,第 9 章至第 12 章分别讲述面向对象方法学引论、面向对象分析、面向对象设计和面向对象实现,第 13 章介绍软件项目管理。正文后面有两个附录,分别讲述了用面向对象方法和结构化方法开发两个软件的过程,对读者深入理解软件工程学很有帮助,也是上机实习的好材料。

本书可作为高等院校“软件工程”课程的教材或教学参考书,也可供有一定实际经验的软件工作人员和需要开发应用软件的广大计算机用户阅读参考。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

## 图书在版编目(CIP)数据

软件工程 / 张海藩编著. —北京:清华大学出版社, 2009. 7

ISBN 978-7-302-19812-3

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

中国版本图书馆 CIP 数据核字(2009)第 045763 号

责任编辑:袁勤勇

责任校对:时翠兰

责任印制:何 莘

出版发行:清华大学出版社

地 址:北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

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

印 装 者:北京鑫海金澳胶印有限公司

经 销:全国新华书店

开 本:185×260

印 张:22

字 数:506 千字

版 次:2009 年 7 月第 1 版

印 次:2009 年 7 月第 1 次印刷

印 数:1~5000

定 价:29.50 元

---

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:010-62770177 转 3103 产品编号:029324-01

# 前 言

## PREFACE

作者编著的《软件工程导论》已经出版了 5 个版本，累计销售了近一百万册，先后荣获全国普通高等学校工科电子类专业优秀教材二等奖和一等奖，并被评为全国优秀畅销书（前 10 名）和北京高等教育精品教材，国内许多高校把它选作“软件工程”课的教材。

但是，也有一些“软件工程”课程学时较少的高校教师反映，《软件工程导论》内容较多，用它作教材感到授课学时比较紧张。为了满足这部分学校的需求，作者对《软件工程导论》作了精心改编，保留了原书中较重要、较新颖的内容，删除或简化了一些较陈旧或较次要的内容，写出了《软件工程》。该书在保持原书结构不变的前提下把篇幅减少了近六分之一，将显著减少授课所需的学时。

编者

2009 年 5 月

# 目 录

## CONTENTS

<b>第 1 章 软件工程学概述</b> .....	1
1.1 软件危机 .....	1
1.1.1 软件危机的介绍 .....	1
1.1.2 产生软件危机的原因 .....	2
1.1.3 消除软件危机的途径 .....	4
1.2 软件工程 .....	4
1.2.1 软件工程的介绍 .....	4
1.2.2 软件工程的基本原理 .....	5
1.2.3 软件工程方法学 .....	5
1.3 软件生命周期 .....	8
1.4 软件过程 .....	11
1.4.1 瀑布模型 .....	11
1.4.2 快速原型模型 .....	13
1.4.3 增量模型 .....	15
1.4.4 螺旋模型 .....	16
1.4.5 喷泉模型 .....	18
1.4.6 Rational 统一过程 .....	19
1.4.7 敏捷过程与极限编程 .....	22
1.4.8 微软过程 .....	26
1.5 小结 .....	28
习题 1 .....	29
<b>第 2 章 可行性研究</b> .....	31
2.1 可行性研究的任务 .....	31
2.2 可行性研究过程 .....	32
2.3 数据流图 .....	34
2.3.1 符号 .....	34

2.3.2	例子	35
2.3.3	命名	38
2.3.4	用途	39
2.4	数据字典	40
2.4.1	数据字典的内容	40
2.4.2	定义数据的方法	41
2.4.3	数据字典的用途	42
2.4.4	数据字典的实现	42
2.5	成本/效益分析	43
2.5.1	成本估计	43
2.5.2	成本/效益分析的方法	44
2.6	小结	46
	习题 2	47
<b>第 3 章</b>	<b>需求分析</b>	<b>49</b>
3.1	需求分析的任务	50
3.1.1	确定对系统的综合要求	50
3.1.2	分析系统的数据要求	51
3.1.3	导出系统的逻辑模型	52
3.1.4	修正系统开发计划	52
3.2	与用户沟通获取需求的方法	52
3.2.1	访谈	52
3.2.2	面向数据流自顶向下求精	53
3.2.3	简易的应用规格说明技术	54
3.2.4	快速建立软件原型	55
3.3	分析建模与规格说明	56
3.3.1	分析建模	56
3.3.2	软件需求规格说明	56
3.4	实体-联系图	56
3.4.1	数据对象	57
3.4.2	属性	57
3.4.3	联系	57
3.4.4	实体-联系图的符号	58
3.5	数据规范化	58
3.6	状态转换图	59
3.6.1	状态	59
3.6.2	事件	59
3.6.3	符号	60

3.6.4 例子 .....	60
3.7 其他图形工具 .....	61
3.7.1 层次方框图 .....	62
3.7.2 Warnier 图 .....	62
3.7.3 IPO 图 .....	63
3.8 验证软件需求 .....	64
3.8.1 从哪些方面验证软件需求的正确性 .....	64
3.8.2 验证软件需求的方法 .....	64
3.8.3 用于需求分析的软件工具 .....	65
3.9 小结 .....	66
习题 3 .....	67
<b>第 4 章 形式化说明技术 .....</b>	<b>69</b>
4.1 概述 .....	69
4.1.1 非形式化方法的缺点 .....	69
4.1.2 形式化方法的优点 .....	70
4.1.3 应用形式化方法的准则 .....	70
4.2 有穷状态机 .....	71
4.2.1 概念 .....	71
4.2.2 例子 .....	73
4.2.3 评价 .....	76
4.3 Petri 网 .....	76
4.3.1 概念 .....	76
4.3.2 例子 .....	78
4.4 Z 语言 .....	79
4.4.1 简介 .....	79
4.4.2 评价 .....	82
4.5 小结 .....	82
习题 4 .....	83
<b>第 5 章 总体设计 .....</b>	<b>85</b>
5.1 设计过程 .....	85
5.2 设计原理 .....	87
5.2.1 模块化 .....	87
5.2.2 抽象 .....	89
5.2.3 逐步求精 .....	89
5.2.4 信息隐藏和局部化 .....	90
5.2.5 模块独立 .....	91

5.3	启发规则	93
5.4	描绘软件结构的图形工具	95
5.4.1	层次图和 HIPO 图	95
5.4.2	结构图	96
5.5	小结	97
	习题 5	98
<b>第 6 章</b>	<b>详细设计</b>	<b>101</b>
6.1	结构程序设计	101
6.2	人机界面设计	103
6.2.1	设计问题	103
6.2.2	设计过程	105
6.2.3	人机界面设计指南	106
6.3	过程设计的工具	108
6.3.1	程序流程图	108
6.3.2	盒图	108
6.3.3	PAD 图	109
6.3.4	判定表	110
6.3.5	判定树	111
6.3.6	过程设计语言	112
6.4	程序复杂程度的定量度量	113
6.4.1	McCabe 方法	113
6.4.2	Halstead 方法	116
6.5	小结	116
	习题 6	117
<b>第 7 章</b>	<b>实现</b>	<b>121</b>
7.1	编码	122
7.1.1	选择程序设计语言	122
7.1.2	编码风格	122
7.2	软件测试基础	123
7.2.1	软件测试的目标	123
7.2.2	软件测试准则	124
7.2.3	测试方法	124
7.2.4	测试步骤	125
7.2.5	测试阶段的信息流	126
7.3	单元测试	126
7.3.1	测试重点	127



7.3.2	代码审查	128
7.3.3	计算机测试	128
7.4	集成测试	130
7.4.1	自顶向下集成	130
7.4.2	自底向上集成	132
7.4.3	不同集成测试策略的比较	132
7.4.4	回归测试	133
7.5	确认测试	133
7.5.1	确认测试的范围	134
7.5.2	软件配置复查	134
7.5.3	Alpha 和 Beta 测试	134
7.6	白盒测试技术	135
7.6.1	逻辑覆盖	135
7.6.2	控制结构测试	138
7.7	黑盒测试技术	144
7.7.1	等价划分	145
7.7.2	边界值分析	148
7.7.3	错误推测	148
7.8	调试	149
7.8.1	调试过程	149
7.8.2	调试途径	151
7.9	软件可靠性	152
7.9.1	基本概念	152
7.9.2	估算平均无故障时间的方法	153
7.10	小结	155
	习题 7	156
<b>第 8 章</b>	<b>维护</b>	<b>161</b>
8.1	软件维护的定义	161
8.2	软件维护的特点	162
8.2.1	结构化维护与非结构化维护差别巨大	162
8.2.2	维护的代价高昂	162
8.2.3	维护的问题很多	163
8.3	软件维护过程	164
8.4	软件的可维护性	166
8.4.1	决定软件可维护性的因素	166
8.4.2	文档	167
8.4.3	可维护性复审	168

8.5	预防性维护 .....	169
8.6	软件再工程过程 .....	170
8.7	小结 .....	172
	习题 8 .....	173
<b>第 9 章</b>	<b>面向对象方法学引论</b> .....	<b>175</b>
9.1	面向对象方法学概述 .....	175
9.1.1	面向对象方法学的要点 .....	175
9.1.2	面向对象方法学的优点 .....	177
9.2	面向对象的概念 .....	181
9.2.1	对象 .....	181
9.2.2	其他概念 .....	183
9.3	面向对象建模 .....	187
9.4	对象模型 .....	188
9.4.1	类图的基本符号 .....	189
9.4.2	表示关系的符号 .....	190
9.5	动态模型 .....	195
9.6	功能模型 .....	196
9.6.1	用例图 .....	196
9.6.2	用例建模 .....	199
9.7	3 种模型之间的关系 .....	200
9.8	小结 .....	201
	习题 9 .....	201
<b>第 10 章</b>	<b>面向对象分析</b> .....	<b>203</b>
10.1	面向对象分析的基本过程 .....	203
10.1.1	概述 .....	203
10.1.2	3 个子模型与 5 个层次 .....	204
10.2	需求陈述 .....	205
10.2.1	书写要点 .....	205
10.2.2	例子 .....	206
10.3	建立对象模型 .....	207
10.3.1	确定类与对象 .....	208
10.3.2	确定关联 .....	210
10.3.3	划分主题 .....	213
10.3.4	确定属性 .....	213
10.3.5	识别继承关系 .....	216
10.3.6	反复修改 .....	216

10.4	建立动态模型	219
10.4.1	编写脚本	219
10.4.2	设想用户界面	220
10.4.3	画事件跟踪图	221
10.4.4	画状态图	222
10.4.5	审查动态模型	223
10.5	建立功能模型	225
10.5.1	画出基本系统模型图	225
10.5.2	画出功能级数据流图	226
10.5.3	描述处理框功能	226
10.6	定义服务	227
10.7	小结	228
习题 10		228
<b>第 11 章</b>	<b>面向对象设计</b>	<b>231</b>
11.1	面向对象设计的准则	231
11.2	启发规则	233
11.3	软件重用	235
11.3.1	概述	235
11.3.2	类构件	237
11.3.3	软件重用的效益	238
11.4	系统分解	239
11.5	设计问题域子系统	242
11.6	设计人机交互子系统	245
11.7	设计任务管理子系统	247
11.8	设计数据管理子系统	249
11.8.1	数据存储管理模式选择	249
11.8.2	数据管理子系统设计	250
11.8.3	例子	252
11.9	设计类中的服务	252
11.9.1	确定类中应有的服务	252
11.9.2	设计实现服务的方法	253
11.10	设计关联	254
11.11	设计优化	255
11.11.1	确定优先级	255
11.11.2	提高效率的几项技术	256
11.11.3	调整继承关系	257

11.12 小结 .....	259
习题 11 .....	260
<b>第 12 章 面向对象实现 .....</b>	<b>261</b>
12.1 程序设计语言 .....	261
12.1.1 面向对象语言的优点 .....	261
12.1.2 面向对象语言的技术特点 .....	262
12.1.3 选择面向对象语言 .....	266
12.2 程序设计风格 .....	266
12.2.1 提高可重用性 .....	267
12.2.2 提高可扩充性 .....	269
12.2.3 提高健壮性 .....	269
12.3 测试策略 .....	270
12.3.1 面向对象的单元测试 .....	270
12.3.2 面向对象的集成测试 .....	271
12.3.3 面向对象的确认测试 .....	271
12.4 设计测试用例 .....	271
12.4.1 测试类的方法 .....	272
12.4.2 集成测试方法 .....	273
12.5 小结 .....	275
习题 12 .....	276
<b>第 13 章 软件项目管理 .....</b>	<b>277</b>
13.1 估算软件规模 .....	277
13.1.1 代码行技术 .....	277
13.1.2 功能点技术 .....	278
13.2 工作量估算 .....	280
13.2.1 静态单变量模型 .....	280
13.2.2 动态多变量模型 .....	280
13.2.3 COCOMO2 模型 .....	281
13.3 进度计划 .....	284
13.3.1 估算开发时间 .....	284
13.3.2 Gantt 图 .....	286
13.3.3 工程网络 .....	287
13.3.4 估算工程进度 .....	288
13.3.5 关键路径 .....	290
13.3.6 机动时间 .....	290

13.4	人员组织	292
13.4.1	民主制程序员组	292
13.4.2	主程序员组	293
13.4.3	现代程序员组	294
13.5	质量保证	296
13.5.1	软件质量	296
13.5.2	软件质量保证措施	298
13.6	软件配置管理	300
13.6.1	软件配置	301
13.6.2	软件配置管理过程	301
13.7	能力成熟度模型	303
13.8	小结	306
	习题 13	307
<b>附录 A</b>	<b>C++ 类库管理系统的分析与设计</b>	<b>309</b>
A.1	面向对象分析	309
A.1.1	需求	309
A.1.2	建立对象模型	310
A.2	面向对象设计	311
A.2.1	设计类库结构	311
A.2.2	设计问题域子系统	312
A.2.3	设计人机交互子系统	313
A.2.4	设计其他类	316
<b>附录 B</b>	<b>一个汉字行编辑程序的设计</b>	<b>317</b>
B.1	设计规格说明	317
B.1.1	外部编辑命令	317
B.1.2	编辑命令	318
B.1.3	输出信息	319
B.2	概要设计	320
B.2.1	正文文件	320
B.2.2	两个工作模式	321
B.2.3	数据元素	322
B.2.4	过程	323
B.3	概要设计结果	323
B.4	详细设计	326
B.4.1	数据元素	326
B.4.2	控制数据元素	328

B. 4. 3	编辑过程 .....	328
B. 4. 4	输入模式的过程 .....	329
B. 4. 5	编辑模式的过程 .....	330
B. 4. 6	编辑程序的详细结构 .....	333
<b>参考文献</b> .....		335

# 软件工程学概述

迄今为止,计算机系统已经经历了 4 个不同的发展阶段,但是,人们仍然没有彻底摆脱“软件危机”的困扰,软件已经成为限制计算机系统发展的瓶颈。

为了更有效地开发与维护软件,软件工作者在 20 世纪 60 年代后期开始认真研究消除软件危机的途径,从而逐渐形成了一门新兴的工程学科——计算机软件工程学(通常简称为“软件工程”)。

## 1.1 软件危机

在计算机系统发展的早期时代(20 世纪 60 年代中期以前),通用硬件相当普遍,软件却是为每个具体应用而专门编写的。这时的软件通常是规模较小的程序,编写者和使用者往往是同一个(或同一组)人。这种个体化的软件环境,使得软件设计通常是在人们头脑中进行的一个隐含的过程,除了程序清单之外,没有其他文档资料保存下来。

从 20 世纪 60 年代中期到 70 年代中期是计算机系统发展的第二个时期,这个时期的一个重要特征是出现了“软件作坊”,广泛使用产品软件。但是,“软件作坊”基本上仍然沿用早期形成的个体化软件开发方法。随着计算机应用的日益普及,软件数量急剧膨胀。在程序运行时发现的错误必须设法改正;用户有了新的需求时必须相应地修改程序;硬件或操作系统更新时,通常需要修改程序以适应新的环境。上述种种软件维护工作,以令人吃惊的比例耗费资源。更严重的是,许多程序的个体化特性使得它们最终成为不可维护的。“软件危机”就这样开始出现了!1968 年北大西洋公约组织的计算机科学家在联邦德国召开国际会议,讨论软件危机问题,在这次会议上正式提出并使用了“软件工程”这个名词,一门新兴的工程学科就此诞生了。

### 1.1.1 软件危机的介绍

软件危机是指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。这些问题绝不仅仅是不能正常运行的软件才具有的,实际上,几

乎所有软件都不同程度地存在这些问题。

概括地说,软件危机包含下述两方面的问题:如何开发软件,以满足对软件日益增长的需求;如何维护数量不断膨胀的已有软件。软件危机具有长期性和症状不明显的特征。

具体地说,软件危机主要有以下一些典型表现:

- 对软件开发成本和进度的估计常常很不准确;
- 经常出现用户对“已完成的”软件产品不满意的情况;
- 软件产品的质量往往达不到要求;
- 软件通常是很难维护的;
- 软件产品往往没有适当的文档资料;
- 软件成本在计算机系统总成本中所占的比例逐年上升;
- 软件开发生产率提高的速度远远不能满足社会对软件产品的日益增长的需求。

以上列举的仅仅是软件危机的一些明显的表现,与软件开发和维护有关的问题远远不止这些。

### 1.1.2 产生软件危机的原因

在软件开发和维护的过程中存在这么多严重问题,一方面与软件本身的特点有关,另一方面也和软件开发与维护的方法不正确有关。

软件不同于硬件,它是计算机系统逻辑部件而不是物理部件。由于软件缺乏“可见性”,在写出程序代码并在计算机上试运行之前,软件开发过程的进展情况较难衡量,软件的质量也较难评价,因此,管理和控制软件开发过程相当困难。此外,软件在运行过程中不会因为使用时间过长而被“用坏”,如果运行中发现了错误,很可能是遇到了一个在开发时期引入的在测试阶段没能检测出来的错误。因此,软件维护通常意味着改正或修改原来的设计,这就在客观上使得软件较难维护。

软件不同于一般程序,它的一个显著特点是规模庞大,而且程序复杂性将随着程序规模的增加而呈指数上升。为了在预定时间内开发出规模庞大的软件,必须由许多人分工合作,然而,如何保证每个人完成的工作合在一起确实能构成一个高质量的大型软件系统,更是一个极端复杂困难的问题,这不仅涉及许多技术问题,诸如分析方法、设计方法、形式说明方法、版本控制等,更重要的是必须有严格而科学的管理。

软件本身独有的特点确实给开发和维护带来一些客观困难,但是人们在开发和使用计算机系统的长期实践中,也确实积累和总结出了许多成功的经验。如果坚持不懈地使用经过实践考验证明是正确的方法,许多困难是完全可以克服的,过去也确实有一些成功的范例。但是,目前相当多的软件专业人员对软件开发和维护还有不少糊涂观念,在实践过程中或多或少地采用了错误的方法和技术,这可能是使软件问题发展成软件危机的主要原因。

与软件开发和维护有关的许多错误认识和做法的形成,可以归因于在计算机系统发展的早期阶段软件开发的个体化特点。错误的认识和做法主要表现为忽视软件需求分析的重要性,认为软件开发就是写程序并设法使之运行,轻视软件维护等。



事实上,对用户要求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的主要原因之一。只有用户才真正了解他们自己的需要,但是许多用户在开始时并不能准确具体地叙述他们的需要,软件开发人员需要做大量深入细致的调查研究工作,反复多次地和用户交流信息,才能真正全面、准确、具体地了解用户的要求。对问题和目标的正确认识是解决任何问题的前提和出发点,软件开发同样也不例外。急于求成,仓促上阵,对用户要求没有正确认识就匆忙着手编写程序,这就如同不打好地基就盖高楼一样,最终必然垮台。事实上,越早开始写程序,完成它所需要用的时间往往越长。

一个软件从定义、开发、使用和维护,直到最终被废弃,要经历一个漫长的时期,这就如同一个人要经过胎儿、儿童、青年、中年和老年,直到最终死亡的漫长时期一样。通常把软件经历的这个漫长的时期称为生命周期。软件开发最初的工作应是问题定义,也就是确定要求解决的问题是什么;然后要进行可行性研究,决定该问题是否存在一个可行的解决办法;接下来应该进行需求分析,也就是深入具体地了解用户的要求,在所开发的系统(不妨称之为目标系统)必须做什么这个问题上和用户取得完全一致的看法。经过上述软件定义时期的准备工作才能进入开发时期,而在开发时期,首先需要对软件进行设计(通常又分为概要设计和详细设计两个阶段),然后才能进入编写程序的阶段,程序编写完之后还必须经过大量的测试工作(需要的工作量通常占软件开发全部工作量的40%~50%)才能最终交付使用。所以,编写程序只是软件开发过程中的一个阶段,而且在典型的软件开发工程中,编写程序所需的工作量只占软件开发全部工作量的10%~20%。

另一方面还必须认识到程序只是完整的软件产品的一个组成部分,在上述软件生命周期的每个阶段都要得出最终产品的一个或几个组成部分(这些组成部分通常以文档资料的形式存在)。也就是说,一个软件产品必须由一个完整的配置组成,软件配置主要包括程序、文档和数据等成分。必须清除只重视程序而忽视软件配置其余成分的糊涂观念。

做好软件定义时期的工作,是降低软件成本提高软件质量的关键。如果软件开发人员在定义时期没有正确全面地理解用户需求,直到测试阶段或软件交付使用后才发现“已完成的”软件不完全符合用户的需要,这时再修改就为时已晚了。

严重的问题是,在软件开发的不同阶段进行修改需要付出的代价是很不相同的,在早期引入变动,涉及的面较少,因而代价也比较低;而在开发的中期,软件配置的许多成分已经完成,引入一个变动要对所有已完成的配置成分都做相应的修改,不仅工作量大,而且逻辑上也更复杂,因此付出的代价剧增;在软件“已经完成”时再引入变动,当然需要付出更高的代价。根据美国一些软件公司的统计资料,在后期引入一个变动比在早期引入相同变动所需付出的代价高2~3个数量级。图1.1定性地描绘了在不同时期引入一个变动需要付出的代价的变化趋势。

通过上面的论述不难认识到,轻视维护是一个最大的错误。许多软件产品的使用寿命长达10年甚至20年,在这样漫长的时

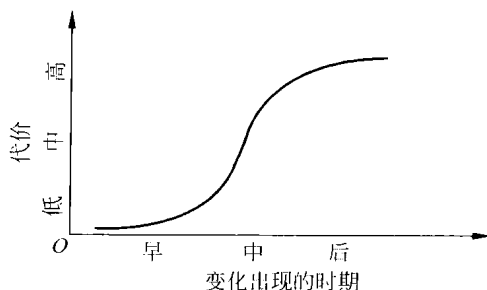


图 1.1 引入同一变动付出的代价随时间变化的趋势