

软件质量管理指南

51Testing软件测试网 组编 张 瑾 编著



测试实践丛书

软件质量管理指南

51 Testing 软件测试网 组编
张 瑾 编著

书名:	责编:
社名:	校次:
开本:	正文页码:
文前页码:	排版员:
日期:	排版公司:
主管签字:	

电子工业出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书从软件质量管理的流程和技术方法等方面对软件质量管理体系进行了详尽的讲述，并对日常工作案例进行剖析，使广大软件质量管理人员能够更加清楚地了解和掌握软件质量管理的精髓。

本书以 CMMI 软件能力成熟度模型为主线，穿插了 PMP 项目管理和软件测试技术的相关知识，从而形成了一套完整的软件质量管理理论。因此，本书是软件企业进行过程改进或 CMMI 认证的辅导资料，同样也可以作为 PMP 和“信息类项目管理师”考试的补充材料。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

软件质量管理指南 / 51Testing 软件测试网组编；张瑾编著. —北京：电子工业出版社，2009.7
（测试实践丛书）
ISBN 978-7-121-09010-3

I. 软… II. ①5… ②张… III. 软件质量—质量管理—指南 IV. TP311.5-62

中国版本图书馆 CIP 数据核字（2009）第 091790 号

责任编辑：江 立

印 刷：北京智力达印刷有限公司

装 订：北京中新伟业印刷有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：787×980 1/16 印张：24 字数：472 千字

印 次：2009 年 7 月第 1 次印刷

印 数：4000 册 定价：49.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

《51Testing 软件测试网作品系列》

编辑委员会名单

编 委：

王 威（具备多年软件开发经验和软件测试工作经验，51Testing 软件测试培训高级讲师）

王 琰（具有丰富的通信终端产品的测试以及管理工作经验，51Testing 软件测试培训高级讲师）

王海龙（Mercury 认证 CPC，对自动化测试有深入的研究和丰富的实战经验，51Testing 软件测试培训高级讲师）

朴春龙（Mercury 认证 CPC，自动化测试专家，51Testing 软件测试培训高级讲师）

吴晓红（具备多年的软件测试工作经验，对测试技术与测试流程及测试管理有丰富的经验及深刻的认识，51Testing 软件测试培训高级讲师）

宋 锋（多年软件开发和软件测试工作经验，具备丰富的项目实战经验，51Testing 软件测试培训高级讲师）

陈 霁（ISO 内审员，积累了丰富的测试和管理工作经验，51Testing 软件测试培训高级讲师）

陈文广（谙熟软件测试流程，擅长自动化测试和性能测试，51Testing 软件测试培训高级讲师）

周 峰（信息产业部认证系统分析员，51Testing 软件测试培训高级讲师）

周春江（具备多年通信协议和通信终端设备的测试工作经验，51Testing 软件测试培训高级讲师）

徐林林（熟悉大型应用软件的开发和测试流程；熟悉性能测试流程、方法和工具（如 LoadRunner 等），51Testing 软件测试培训高级讲师）

商 莉（多年从事软件开发、软件测试及质量保证方面的管理工作，51Testing 软件测试培训高级讲师）

编辑部成员：张晓晓 蒋玉鹏

前 言

软件质量管理是个全组织、多角色共同参与的、复杂的系统过程，好的软件质量是各级软件管理人员孜孜追求的最高梦想。

软件质量管理体系的知识涵盖了软件工程、CMMI 软件能力成熟度模型、PMP 项目管理以及软件测试技术的理论。其中，软件工程主要介绍了各种生命周期模型，这是软件研发和质量管理的基礎，也是 CMMI 软件能力成熟度模型和 PMP 项目管理理论中非重点介绍的内容；PMP 项目管理理论适用于任何行业的项目管理工作，它详细介绍了制定项目估算、预算的方法，以及制定项目进度计划的各种技术，这些是 CMMI 软件能力成熟度模型和软件工程的重要补充；CMMI 软件能力成熟度模型是当今最流行的一种对软件企业成熟度的评判标准，它所涵盖的内容之广及体系之完整都是前所未有的。CMMI 将软件的管理过程拆分为多个 PA（过程域），并详细介绍了每个 PA 所需要完成的工作、流程以及流程中必备的产出物，它是软件质量管理中的核心部分。但 CMMI 软件能力成熟度模型着重于过程的定义，有些具体的操作方法和技術就必须参考 PMP 项目管理理论或软件测试理论的相关知识。软件测试一直以来都被很多人误解为等同于软件质量管理，多样的软件测试技术正是 CMMI 软件能力成熟度模型 VER（验证）的重要补充内容。总的来说，软件工程中生命周期模型好比盖房子时打下的地基，CMMI 软件能力成熟度模型就是房子的框架结构，PMP 项目管理以及软件测试技术的理论就是填充房子的砖石，而盖好的这座房子就是软件质量管理体系。

本书以 CMMI 软件能力成熟度模型为主线，第 1 章对软件质量管理体系进行了概述，第 2~4 章介绍了软件质量管理所必备的常用技术“验证”、“确认”和“同行评审”；第 5~8 章介绍软件质量管理的基础管理流程“质量保证”、“配置管理”、“度量管理”和“风险管理”的知识；第 9~11 章介绍软件项目管理相关的“项目集成管理”、“项目计划”和“项目监控”的知识；第 12~14 章介绍了软件质量管理体系的“需求工程”、“决策分析”和“产品集成”的理论；第 15 章重点介绍了如何进行持续的质量改进，第 16 章为广大读者讲解了微软最新的软件项目工具“Team Foundation Server”的基本使用方法。

为了让广大读者更好地理解软件质量管理的理论，本书在每章的结束都针对软件项目开发过程中的常见问题进行案例分析，目的是为了将软件质量管理体系的知识与实际项目进行联系，更好地让软件各级管理人员进行理解和应用。

本书总结了当今软件质量管理所需要的全部知识，其中重点介绍的 CMMI 软件能力成熟度模型可以为软件公司高层管理人员和过程改进小组（EPG）的工作提供帮助；PMP 项目管理的相关技术可以为软件公司的项目管理人员提供日常的项目指导并作为 PMP 考试的参考资料；每章的案例分析也采取了“信息类项目管理师”的考试形式，希望可以为参加“信息类项目管理师”考试的朋友提供帮助。

这些年来我一直希望可以将总结的软件质量管理的知识和理论与大家分享，本书能够顺利出版首先要感谢 51Testing 所提供的机会，也要感谢各位编辑的辛勤劳动。同时还要感谢长期以来支持我的朋友和我的妻子蔡觅女士，你们是我成长的最大动力！

作 者

2009 年 5 月 28 日于苏州

目 录

CONTENTS

软件质量管理指南

第 1 章 软件质量管理体系概述	1
1.1 软件质量复杂度的来源	2
1.2 “过程”在软件研发中的重要性	3
1.3 小结	7
1.4 思考题	7
第 2 章 软件质量管理的检查方式——验证	8
2.1 软件验证的最佳实践	9
2.2 软件质量大师的观点	13
2.3 常用的验证方法	15
2.3.1 边界值测试	16
2.3.2 白盒测试	17
2.3.3 等价类分法	23
2.3.4 压力测试	24
2.4 案例分析——如何计算系统的并发用户数	31
2.5 小结	32
2.6 思考题	32
第 3 章 软件质量管理的信任机制——确认	33
3.1 软件确认管理的概述	34
3.2 软件确认流程及最佳实践	35
3.2.1 确认的准备工作	35
3.2.2 执行确认	41
3.3 软件确认过程中常见问题及案例分析	41
3.3.1 为什么开发和测试之间总是反复	42
3.3.2 确认是对需求变更的约束	43
3.4 小结	44
3.5 思考题	44

目 录

CONTENTS

软件质量管理指南

第 4 章 软件质量管理的预防手段——同行评审	45
4.1 软件同行评审的概述	45
4.2 软件同行评审流程及最佳实践	47
4.2.1 同行评审计划阶段	47
4.2.2 同行评审启动阶段	52
4.2.3 同行评审执行阶段	55
4.2.4 同行评审收尾阶段	56
4.2.5 同行评审流程裁剪指南	56
4.2.6 同行评审最佳实践	58
4.3 软件同行评审常见问题及案例分析	59
4.3.1 案例 1——如何提高同行评审的效果	59
4.3.2 案例 2——如何计算同行评审的投资回报率	61
4.3.3 案例 3——如何更好地执行同行评审	62
4.4 小结	62
4.5 思考题	62
第 5 章 软件质量管理的审计体系——质量保证	63
5.1 软件质量保证概述	64
5.1.1 PPQA 与 SQC 的区别	65
5.1.2 软件质量保证人员的素质和责任	68
5.1.3 软件质量保证人员与其他岗位的关系	70
5.2 软件质量保证流程及最佳实践	71
5.2.1 对软件研发过程的审计	71
5.2.2 对软件工作产品的审计	73
5.3 软件质量保证常见问题及案例分析	76
5.4 小结	79
5.5 思考题	79

软件质量管理指南

第 6 章 软件质量管理的基石——配置管理	80
6.1 软件配置管理概述	81
6.2 软件配置管理流程及最佳实践	82
6.2.1 建立基线	83
6.2.2 配置库及工具	91
6.2.3 跟踪和控制变更	104
6.2.4 建立基线完整性	107
6.3 利用分支与合并进行软件配置管理工作	111
6.4 小结	112
6.5 思考题	113
第 7 章 软件质量管理的客观洞察力——度量管理	114
7.1 软件度量管理概述	115
7.1.1 测量的基础知识	115
7.1.2 度量的基础知识	116
7.2 软件度量管理流程及最佳实践	117
7.2.1 软件度量的目标	117
7.2.2 软件度量的实体与属性	120
7.2.3 软件度量的方法	121
7.2.4 软件度量的指示器	123
7.2.5 软件度量管理的流程	126
7.3 软件度量管理常见问题及案例分析	126
7.3.1 如何提高软件度量的准确性	126
7.3.2 从哪里可以收集到度量所需的数据	128
7.4 小结	129
7.5 思考题	129
第 8 章 软件质量管理的预警措施——风险管理	130
8.1 软件风险管理的概述	131

目 录

CONTENTS

软件质量管理指南

8.1.1 风险的类型	131
8.1.2 风险的来源	132
8.1.3 风险的应对策略	134
8.2 软件风险管理流程及最佳实践	134
8.2.1 建立组织级风险库	135
8.2.2 识别项目风险、定义风险的属性	137
8.2.3 分析风险并对风险进行排序	138
8.2.4 风险的跟踪	141
8.3 软件风险管理常见问题及案例分析	142
8.3.1 为什么风险识别总不准确	142
8.3.2 为什么项目计划总是不准确	143
8.3.3 如何在项目中进行风险跟踪	143
8.4 小结	144
8.5 思考题	144
第 9 章 软件质量管理的统筹规划——	
项目集成管理	145
9.1 项目整体策划的流程及最佳实践	147
9.2 XP 极限式开发模型与 CMMI 的比较	154
9.3 小结	155
9.4 思考题	155
第 10 章 软件质量管理的策划——项目计划	156
10.1 软件项目计划概述	159
10.2 软件计划的流程及最佳实践	161
10.2.1 对项目进行整体估算	161
10.2.2 对项目范围进行管理	166
10.2.3 建立时间进度计划	190
10.2.4 建立项目费用预算	200

软件质量管理指南

10.2.5 计划项目的其他内容	203
10.2.6 建立项目计划的基准	204
10.3 软件计划的常见问题及案例分析	204
10.3.1 假设是项目计划的根本条件	204
10.3.2 关键路径的计算之案例 1	205
10.3.3 关键路径的计算之案例 2	206
10.4 小结	206
10.5 思考题	206
第 11 章 软件质量管理的监督手段——项目监控	207
11.1 软件监控管理流程及最佳实践	208
11.1.1 监控项目的主要参数	208
11.1.2 监控项目的次要参数	211
11.1.3 监控项目的方法	214
11.2 软件监控管理常见问题及案例分析	218
11.2.1 项目参数之间存在相互的影响和依赖	218
11.2.2 挣值法在软件项目中的应用	219
11.3 小结	221
11.4 思考题	221
第 12 章 软件质量管理的根源——需求工程	222
12.1 软件需求工程概述	223
12.2 软件需求开发的流程及最佳实践	224
12.2.1 需求调研的方法	225
12.2.2 软件需求分析的概述	228
12.2.3 软件需求规格化	231
12.2.4 需求验证及确认方法	238
12.3 软件需求管理的流程及最佳实践	244
12.4 软件需求工程常见问题及案例分析	248

目 录

CONTENTS

软件质量管理指南

12.4.1 对需求关键干系人分析的重要性	248
12.4.2 调研时需求关键干系人不能被代替	249
12.4.3 需求文档规范化与 XP 极限式开发的理念 是否矛盾	249
12.4.4 利用需求跟踪矩阵来应对项目变更	250
12.5 小结	251
12.6 思考题	251
第 13 章 软件质量管理的群体决议机制—— 决策分析	252
13.1 软件决策分析概述	253
13.2 软件决策分析流程及最佳实践	254
13.3 软件决策分析常见问题及案例分析	259
13.3.1 决策树的使用方法	259
13.3.2 加权打分的决策方法	261
13.4 小结	263
13.5 思考题	263
第 14 章 软件质量管理的构建机制——产品集成	264
14.1 软件产品集成管理概述	265
14.2 软件产品构建的流程及最佳实践	266
14.2.1 软件产品集成的准备工作	266
14.2.2 确保软件产品接口的完整性	272
14.2.3 集成并交付产品	276
14.2.4 通过日构建来实现持续集成	279
14.2.5 日构建工具 NAnt 的使用	283
14.3 软件产品集成管理常见问题及案例分析	295
14.3.1 在配置管理下如何开展产品集成	295
14.3.2 产品集成的顺序与项目进度计划的关系	296

软件质量管理指南

14.4 小结	296
14.5 思考题	297
第 15 章 软件质量的持续改进	298
15.1 组织过程改进的焦点	299
15.1.1 确定过程改进的需要	299
15.1.2 计划和执行过程改进	304
15.2 定义组织标准过程的最佳实践	309
15.2.1 建立组织级标准过程和组织财富库	309
15.2.2 定义生命周期模型	311
15.2.3 定义裁剪指南	319
15.2.4 建立组织级度量库和工作环境	326
15.3 组织培训的最佳实践	328
15.3.1 建立组织培训的能力	328
15.3.2 实施培训	332
15.4 案例分析在实际工作中如何把握质量改进的时机	334
15.5 小结	334
15.6 思考题	334
第 16 章 TFS 在软件研发中的应用	335
16.1 TFS 的拓扑结构	336
16.2 TFS 团队项目功能简介	340
16.2.1 创建工作项	341
16.2.2 添加查询视图	345
16.2.3 源代码管理	347
16.2.4 项目门户	351
16.3 Team Build	358
16.4 小结	360
附录 A 思考题答案	361

1

第 1 章

软件质量管理体系概述

质量是指“产品具备满足明确或隐含需求能力的所有特性的总和”，一提到如何提高软件的质量，很多软件从业人员想到的就是加强软件测试的力度、增派软件测试的人手，但事实上这种做法的效果并不理想，主要原因是没有理解软件质量管理的内涵。软件的缺陷早已在软件研发的过程中生成，软件测试只是一种弥补软件质量缺陷的手段，在项目结束日期确定的情况下对软件产品进行“无穷尽”的测试是不具备条件的，何况“无穷尽”的测试也是不科学、不现实的。

软件质量管理是一套复杂的系统工程，软件质量的好坏就好比木桶盛水的多少，木桶所盛的水越多，软件产品的质量就越好。如图 1-1 所示，木桶所盛水的多少却是取决于木桶中最短的那块木板的高度。

那么软件质量管理体系是由哪些“木板”组成的呢？以 CMMI 和软件工程理论为依据，它们分别是：

- 讲述软件检查方式的验证管理（VER）和讲述软件预防手段的同行评审（Peer Review）
- 讲述软件信任机制的确认管理（VAL）
- 讲述软件审计体系的质量保证（PPQA）
- 作为软件管理基石的配置管理（CM）

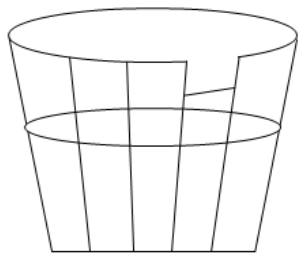


图 1-1 木桶理论

- 讲述软件客观洞察力的度量管理（MA）
- 讲述软件预警措施的风险管理（RSKM）
- 讲述软件统筹规划的项目集成管理（IPM）
- 讲述软件详细策划的项目计划（PP）
- 讲述软件监控手段的项目监控（PMC）
- 讲述软件需求工程的需求开发（RD）和需求管理（ReqM）
- 软件群体决策机制的决策分析（DAR）
- 讲述软件构建机制的产品集成（PI）
- 项目供应商管理（SAM）
- 项目的技术解决方案（TS）
- 讲述软件持续改进的组织过程的焦点（OPF）、组织过程的定义（OPD）和组织培训（OT）

1.1 软件质量复杂度的来源

由此可见，软件质量的提高绝非只是通过增强软件测试可以达到的，那为什么提高软件产品的质量会如此复杂呢？这就要对软件项目的特点进行分析。软件项目同样具备项目的临时性和独特性的特征，也会受到时间、成本、资源的约束，在软件项目进行的过程中也会对需求进行逐步的细化。但软件项目与其他类型项目相比的复杂性却不在于此，而是具有软件行业特点的以下因素：

- 软件项目交付给客户的最终产品是一种看不见、摸不着、无形的、需要人脑理解的“逻辑”产品。
- 所谓“隔行如隔山”，软件产品的业务逻辑集中体现了客户所从事工作的最佳实践，软件研发人员需要跨行业学习并理解相关的知识。
- 软件产品是一种主观的、无形的“逻辑”产品，在软件项目中“变化是永恒的，不变是短暂的”。正如《大话西游之大圣娶亲》中唐僧说过的一句话：“妖要是有了仁慈之心，就不再是妖，而是人妖。”连妖都会变心，那更何况是凡夫俗子的人了。
- 软件项目的需求 60%以上都是隐形的需求，有时客户所提供的原始需求中简短的几个字就可以延伸成为一个规模不小的模块。
- 工业化是建立在“生产线”基础上的，但是软件行业“生产线”的概念却不明显，原因是软件行业是知识密集型的行业，人的大脑就是生产线上的设备，所以不能使

用传统方式来对它进行规范，因此必须加强开发标准流程的定义和规范。

- 软件开发人员对文档的重视程度不够，通常认为只有写代码才是他们的本职工作。就像很久以前的一句话“人类没有联想，世界将会怎样”，受过高等教育的软件开发人员的思维都是极其活跃的，可是一旦项目没有了文档，那么又该用什么来统一软件项目团队成员对最终交付产品的认识呢？
- 软件开发人员对单元测试的重视程度不够，认为测试工作都是软件测试人员的事情。在当今主流的开发模型中对单元测试的要求越来越高，这正是先进软件质量理念“尽早测试”的集中体现。
- 软件研发人员往往会忽略了软件开发的真实目的。由客户投资来研发的软件其目的不是为了要这些代码，而是希望利用这些代码辅助实现其商业目的。这正是所谓“软件代码不值钱，值钱的是软件所实现的业务逻辑”。

1.2 “过程”在软件研发中的重要性

为了提高软件产品的质量，彻底解决软件项目的难点，唯一的方法就是通过提高软件研发过程的质量来带动软件产品质量的提高。例如当今山寨手机、山寨春晚、山寨明星等比比皆是，“山寨”已经成为了一种文化，“山寨”一方面体现了时尚和潮流，另外一方面却体现了质量的低下。假如你要去买一辆汽车，在4S店内销售人员正在向你推荐两款车型，其中一款是代表了世界先进工艺和生产线的知名品牌“本田”，一款是由技术精湛的技师通过车床、锤子等工具纯手工工艺打造的品牌“木田”，如果这两款车价格相同，你会选择哪一款呢？相信所有人都会选择“本田”而不是“木田”。因为“本田”汽车是由高度自动化的生产线生产，生产过程中的每个工艺都受到了严格的控制，而“木田”却是由纯手工工艺打造，每个工艺的好坏完全取决于技师当时的心情和工作状态，没有人可以保证生产的所有工艺中汽车质量都是完全合格的。因此选择汽车的过程就是对汽车厂商生产过程质量的评判，软件行业同样如此。

软件研发过程的质量是指对软件项目已定义的生命周期模型、各个过程的流程、模板、准则、项目计划及其从属计划等的遵循程度。遵循的程度越高，软件研发过程的质量就越高，软件产品的质量才会越高。项目计划是昨天对未来的规划，它是项目实施所遵循的重要依据，虽然软件项目的变更通常非常频繁，但是也不能因此而不做项目计划，这也就是项目管理名言“没有计划也就没有变化”的来源。

很多软件研发人员不是不愿意遵循项目已定义的流程和计划，而是恐惧项目的变更，因为项目的变更打乱了项目的计划。项目的变更是个既复杂又简单的过程，之所以复杂是

因为变更就像多米诺骨牌那样对项目会产生连锁反应，如何准确判断变更的影响范围，对变更进行估算和风险管理是复杂的直接原因。简单的原因是项目管理人员只要遵循已定义的变更流程去执行就能够很好地控制变更。有人认为执行变更流程的成本会很高，不如直接行动的效果明显，但现实中的结果通常是相反的。变更的流程不仅复杂而且还要和各个部门进行协商，虽然变更还没有被执行，但是经过了变更的流程后就已经可以准确预测到变更后的结果，这也就是通过软件开发过程的质量来保证软件产品质量的最佳实践。

当今对软件开发过程质量的评判主要以 SEI (Software Engineering Institute) 颁布的 CMMI (Capability Maturity Model Integration) for Development 为标准。它主要是对软件开发各个过程的能力进行衡量，以判断软件企业研发管理的成熟度。CMMI 是对软件开发过程中的各个环节进行量体裁衣，“适合”二字体现了它的精髓。

追求产品的高质量是所有管理人员的共同心愿，正如全球著名的质量管理大师、“零缺陷”之父、伟大的管理思想家克劳斯比对质量提出的 4 条基本原则所说的那样：

- 原则 1：质量是符合要求，而不是最好

“好、最好、卓越”都是主观描述，而不能衡量，因此质量的最高境界是符合要求。

- 原则 2：预防产生质量，检验不能提高质量

产品的质量是靠系统性的预防而不是检验。检验是在过程结束后将产品的缺陷挑出来，可此时产品的缺陷已经产生。企业应该将资源投入在预防工作中而不是花费在错误的补救上。

- 原则 3：零缺陷

每个工作环节的标准必须是零缺陷，不能出现某些时候满足标准或者是大部分的工作满足标准。

- 原则 4：用不符合项的价值来衡量质量

质量可以用纠正不符合项所产生的价值来衡量。

除了以上 4 个质量的基本原则外，克劳斯比还极富艺术性地提出了“质量是芭蕾舞，而不是曲棍球”的理论。曲棍球是一种依靠临场发挥并对抗激烈的体育运动，在比赛的过程中如果因为本方的失误导致对手进球，那么只要依靠努力多进对方几个球还是会取得最终的胜利。而芭蕾舞在演出前都是经过精心策划和反复排练的，舞蹈演员的每个动作、音乐的每个节拍都必须准确无误，在表演时的任何差错都是无法补救的，也就是说芭蕾舞追求的是“零缺陷”的境界。在软件研发的过程中各级管理人员往往采用“曲棍球”式的管理模式，时不时地下去检查一圈，发现问题解决问题，时间久了回想一下也许会发现很多问题都在重复发生。而采用“芭蕾舞”式的管理人员却比较专注于制订更合理、更严谨的管理过程来减少缺陷的发生。

通过以上的介绍，大家可以了解到提高软件质量的正确途径是对软件研发过程质量的改进，CMMI 是目前为止最好的方法论。但 CMMI for Development 是一个内容庞大的体系，大家可以通过以下例子对其进行理解。

假设你是一个 5 星级酒店餐饮部的经理，酒店会承揽很多宴会，为了更好地为与会人员服务，以满足不同情况下的用餐需求，你应该如何运用 CMMI 第 3 级的理论进行管理呢？

首先应该制订餐饮部门的标准作业流程，其步骤如下：

STEP 01 制订标准的宴会生命周期。

STEP 02 识别标准就餐过程中所使用的工具、材料和各种资源。

STEP 03 收集以往会议就餐的持续时间、成本、服务员人数等历史数据供日后进行参考。

STEP 04 向餐饮部的所有员工收集以往工作中发生的意外情况，并以此为依据制订风险数据库。

STEP 05 定义餐饮部的工作环境标准，其中包括餐饮部的日常行政管理规范。

宴会的标准生命周期可以制订多套方案，以满足中餐、西餐、自助餐的不同需求。以中餐为例：

- 接订单
- 制订菜单
- 设计菜式
- 备料
- 制造菜肴
- 厨师长检查
- 摆桌上菜
- 餐后小点
- 收拾碗碟
- 总结经验教训

然后制订餐饮部的裁剪指南：

- 定义餐饮规模的分类。
- 根据不同餐饮的分类定义对不同餐饮标准生命周期中各个流程的裁剪指南，例如：较小规模的宴会可以省略餐前点，大型的宴会可以增加来宾发言。
- 根据不同的餐饮标准生命周期的各个流程定义产出物。例如：宴会开始时的产出物有来宾签到簿，宴会结束时的产出物有赠送来宾的礼品。
- 定义度量的裁剪指南。不同规模的宴会可能需要度量的内容也不同，对于大型宴会可能要分别统计男女来宾的数量，对于小型宴会可能只要统计来宾的总数。

为了进一步提高今后的服务质量，你还需要组织酒店的高级管理人员成立一个过程改进小组，定期对宴会的服务质量进行评估，找出优势和劣势，从而发现改进的机会（OPF）。当发现了服务质量的改进机会后，就可以更新或制订标准餐饮作业流程（OPD），最后通过培训的手段将新的餐饮作业流程对员工进行培训（OT）。

在有了一套完整的标准餐饮作业流程后，就可以开始承接各种类型的宴会。以瀑布式的使用寿命模型为参考，一个宴会可以分为以下几个阶段：

- ① 立项：接到宴会的订单。
- ② 需求调研：详细询问宴会的规模、类型、参与宴会人员的特点、喜欢的菜式等。
- ③ 项目计划：根据需求调研的内容对宴会进行估算，并制订详细的项目计划和从属计划。
- ④ 设计阶段：行政总厨会根据需求调研的内容来设计每道菜的样式。
- ⑤ 开发阶段：采购人员去备料，厨房开始烹饪。
- ⑥ 系统测试阶段：行政总厨会对每个厨师烧制的菜肴进行检查。
- ⑦ 用户验收阶段：摆桌，宴会开始进行。
- ⑧ 结项阶段：收拾碗筷，相关人员一起对本次宴会进行总结。

上述瀑布模型与 CMMI 理论相结合就可以得到更加细化的内容，如下所示：

① 立项：接到宴会的订单。根据订单的大小，依据餐饮部的裁剪指南对生命周期模型、流程、产出物等进行裁剪，生成针对本次宴会已定义的流程（IPM），酒店的相关领导要对此流程进行评审和确认（VAL）。在宴会启动前建立配置管理的数据库（CM），保证所有环节具有可追溯性和完整性。

② 需求调研：详细询问宴会的规模、类型、参与宴会人员的特点、喜欢的菜式等（RD），对需求进行分析，如果客户对某些内容进行变更，那么要按照需求变更的流程对它进行评估和审批。通过需求跟踪矩阵的制订对变更的影响范围进行判断（ReqM），最后还要与客户对宴会的需求进行确认（VAL）。

③ 项目计划：根据需求调研的内容对宴会进行估算，并制订详细的项目计划和从属计划（PP），从属计划包括采购计划（SAM）、宴会的进程等内容。另外还需要对本次宴会的风险进行识别和管理（RSKM），对于客户需求的某些高档配料应该对供应商的人选进行决策和选择（DAR）。这些内容也需要与客户以及酒店的各个部门进行确认（VAL）。

④ 设计阶段：行政总厨会根据需求调研的内容来设计每道菜的样式（TS），餐饮部的经理要对设计阶段的工作进行监控（PMC、MA）。

⑤ 开发阶段：采购人员去备料，厨房开始烹饪（TS）并拼盘（PI）。

⑥ 系统测试阶段：行政总厨会对每个厨师烧制的菜肴进行检查（VER）。

⑦ 用户验收阶段：摆桌，宴会开始进行（VAL）。

⑧ 结项阶段：收拾碗筷，相关人员一起对本次宴会进行总结。

在以上各个阶段中，酒店的质量管理部门都会对宴会的质量进行第三方审计和监控（PPQA）。

1.3 小结

本章对软件质量体系进行了概括性的描述，软件质量的提高不是依靠软件测试实现的，而是在于软件研发整体过程的提高。软件质量管理的范围不仅包含了项目管理的内容，而且还覆盖了软件工程的所有部分。

1.4 思考题

1. CMMI 第3级的18个PA分别是什么？
2. 软件质量的提高是通过软件测试还是软件研发各过程的整体提高？为什么？

2

第 2 章

软件质量管理的检查方式——验证

软件产品质量的检查工作是由软件测试人员和软件开发人员共同负责并完成的，软件测试人员一般使用的检查方法是功能测试、回归测试等黑盒测试的技术，软件开发人员通常使用的是单元测试、集成测试等白盒测试的技术。不管他们采用的是哪种测试技术，他们所做的工作都可以称为对软件产品的验证工作。验证的目的在于确保工作产品符合其指定的需求。如图 2-1 所示，软件验证的过程可以抽象为以下 3 个部分：验证的准备工作、验证的执行工作、纠正措施。

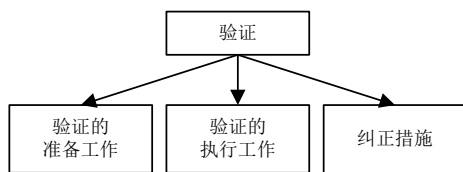


图 2-1 验证工作的 3 个组成部分

软件测试人员在进行验证的过程中也要把握尺度，通常所遵循原则如下：

① 软件各种测试方法的目的是为了发现存在的缺陷，但永远不能证明软件系统不存在缺陷。因为软件产品对于测试人员来说就是一个“黑盒子”，而软件的缺陷却已经在生产的过程中产生了。

- ② 要想在有限的时间和资源下进行无穷尽的测试是不可能的，因此软件测试也要适可而止。
- ③ 测试要尽早介入，由于软件的复杂性和抽象性，在软件生命周期各个阶段都可能产生错误，所以不应把软件测试仅仅看做是软件开发的一个独立阶段的工作，而应当把它贯穿到软件开发的各个阶段中。
- ④ 帕雷托的 80/20 原则，80%的缺陷是因为 20%的错误原因导致的。
- ⑤ 相同的测试用例反复使用是没有效果的，因为测试人员也会产生“审美疲劳”。

2.1 软件验证的最佳实践

验证的对象是软件产品或软件工作产品，验证的过程是将其与对应的客户需求、产品需求和产品组件需求进行比较，以判断它是否符合预期要求。验证是一种渐进式过程，因为它发生在整个研发的过程中，从对需求的验证开始，直到为已开发完成的软件产品进行验证。

验证的准备工作可以抽象为 3 个部分，如图 2-2 所示，它们是识别验证的工作产品和方法、建立验证的环境，以及建立验证的流程和准则。

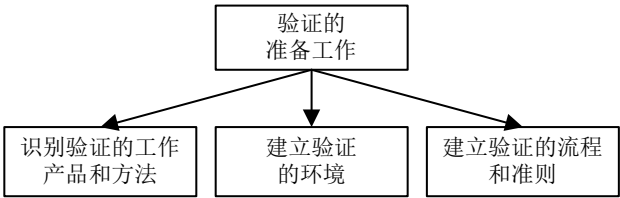


图 2-2 验证的准备工作

1. 识别验证的工作产品和方法

验证和确认有些类似，但强调的重点不同。验证确保“你正确地做了”；确认确保“你做了正确的事”。验证的准备工作首先就是选择待验证的工作产品，这个工作可以在项目过程定义后进行，工作产品验证清单如表 2-1 所示。如图 2-3 所示，在项目过程定义中会依据项目的类型、选择的生命周期模型对项目产出物进行裁剪。软件测试人员在此基础上就可以选择该项目待验证的工作产品，并为每一个工作产品确定验证的方法，常用的验证方法如下：

- 单元测试
- 集成测试
- 性能测试

- 同行评审
- 系统测试
- 验收测试

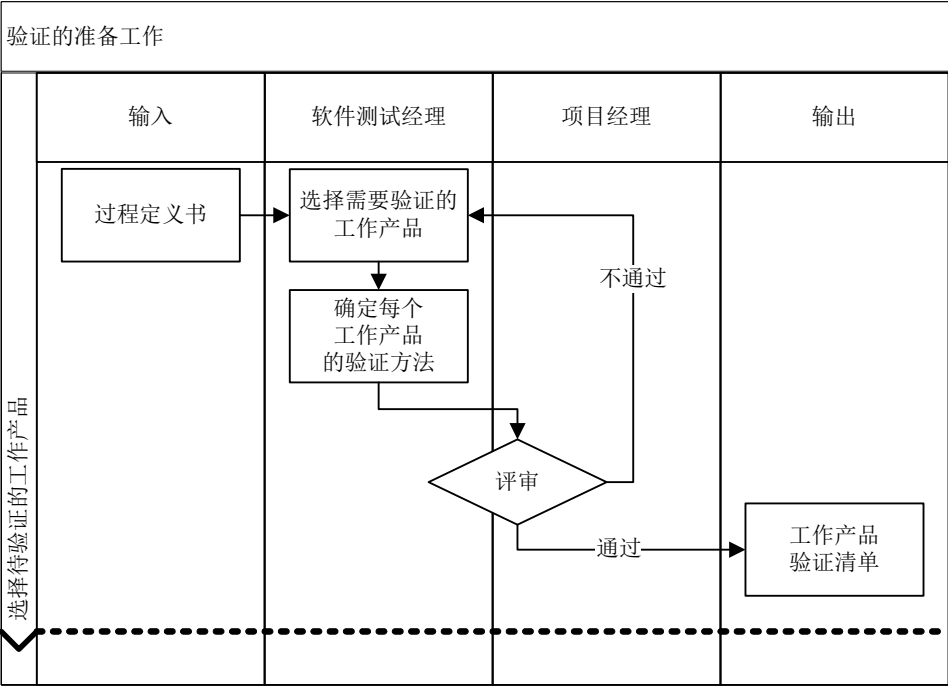


图 2-3 选择待验证的工作产品

表 2-1 工作产品验证清单

软件生命周期模型	阶 段	工作产品	是否需要验证	验证的方法
瀑布式模型	项目立项阶段	项目立项书	—	—
		需求调研计划	√	同行评审
	需求调研阶段	软件需求说明书	√	同行评审
		系统需求规格说明书	√	同行评审
	项目计划阶段	项目过程定义书	√	同行评审
		项目估算表	√	同行评审
		项目计划	√	同行评审
		配置项列表	√	同行评审
		项目进度计划	√	同行评审

续表

软件生命周期模型	阶 段	工作产品	是否需要验证	验证的方法
瀑布式模型	项目计划阶段	配置管理计划	√	同行评审
		质量保证计划	√	同行评审
		风险管理计划	√	同行评审
		度量计划	√	同行评审
		配置库结构和权限表	√	同行评审
		系统测试计划	√	同行评审
		功能点估算记录	√	同行评审
		采购计划	√	同行评审
		WBS	√	同行评审
		产品集成计划	√	同行评审
	软件设计阶段	概要设计说明书	√	同行评审
		详细设计说明书	√	同行评审
		接口与组件对照表	√	同行评审
		产品集成方案	√	同行评审
	编码开发阶段	产品代码	√	单元测试 集成测试 性能测试 系统测试
		代码交付确认单	—	—
	系统测试阶段	系统测试用例	√	同行评审
		系统测试总结报告	—	—
	产品验收阶段	用户验收测试用例	√	同行评审
		验收测试报告	—	—
	项目结项阶段	项目总结及度量分析总结报告	—	—
		过程改进解决方案	√	同行评审
		过程改进建议	—	—
.....

2. 建立验证的环境

不管使用哪一种测试工具或测试技术都要有特定的验证环境，例如单元测试要使用与开发相同的环境，系统测试要使用与用户日常工作相同的环境，同行评审要使用会议室等办公环境。而验证环境搭建的好坏会直接影响验证的效果，例如：通过网络会议对需求文档进行评审时，如果网络时断时续，那么评审的效果一定会大打折扣。因此在进行验证前

先制订验证环境的检查表，如表 2-2 所示，并对验证环境进行检查。

表 2-2 系统测试环境检查表

项目名称		北极光		项目经理	刘大力
QA 名称		王秋香		审计时间	2009-3-4
序号	检查项		审计结果	本次审计是否适用	备 注
1	系统测试服务器的 CPU 是否为 Intel 3.0 GHz，内存是否为 2GB，硬盘是否为 250GB		√	√	
2	系统测试服务器的操作系统是否是 Linux		√	√	
3	系统测试服务器的数据库是否是 MySQL		√	√	
4	待测试的代码是否已经放置到 D:\Code\目录下		√	√	

在识别待验证工作产品的同时，验证环境也随之被识别出来。有些验证环境可以重用，例如：单元测试环境和集成测试环境可以重用，而有些工作产品则需要购买或自行开发。项目经理在制订项目计划时需要对此进行考量。

3. 建立验证的流程和准则

所有的验证方法都会有验证的准则，例如：对代码进行验证的准则是单元测试用例，系统测试验证的准则是系统测试用例。所有验证准则都必须包含 3 个要素：输入、操作步骤和期望的输出。例如：单元测试的输入是单元测试数据，操作步骤就是一行行的单元测试代码，期望输出的结果可以通过断言来实现；自动化测试工具 QTP 的输入是 DataTable 中的测试数据，操作的步骤是 QTP 脚本中的每行代码，期望的输出可以通过验证点来实现。

• 传统的 V 模型的验证流程

传统的 V 模型如图 2-4 所示，该模型已经退出了历史舞台，很少有软件公司会按照该模型的流程对软件工作产品进行验证，因为它与现代软件质量“尽早开始测试”的理论相矛盾，但是该模型却为每种验证方法指明了验证所参考的依据和标准，这是它最难能可贵的地方。

• V 模型的拓展 W 模型

W 验证模型是 V 模型的一种实际拓展，它体现了尽早测试和不断测试的原理，更加符合现代软件质量的思想。W 模型由 Evolutif 公司在 1999 年提出，如图 2-5 所示，它强调验证工作伴随整个软件开发的周期，而且测试的对象不仅仅局限于代码，需求文档、设计文档等工作产品同样需要进行测试。

但 W 模型中需求文档、设计文档、程序代码等验证活动仍然以串行方式进行，对于复杂多变的软件开发情况来说，W 模型有时也会面临困扰。

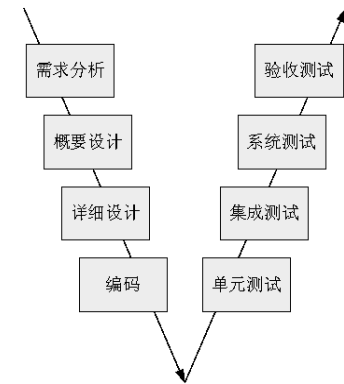


图 2-4 V 模型的验证流程

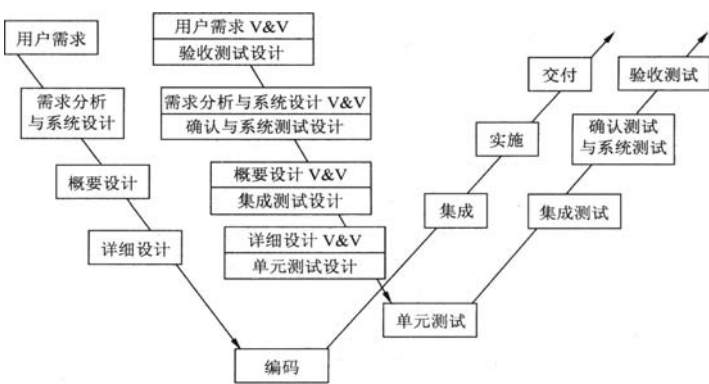


图 2-5 W 模型的验证流程

• H 模型的验证流程

H 模型将软件验证过程看做一个独立于软件开发的流程，并贯穿整个软件产品研发的生命周期。当某个需要验证的时间和资源都就绪时，软件验证的工作也就随之展开，其流程如图 2-6 所示。

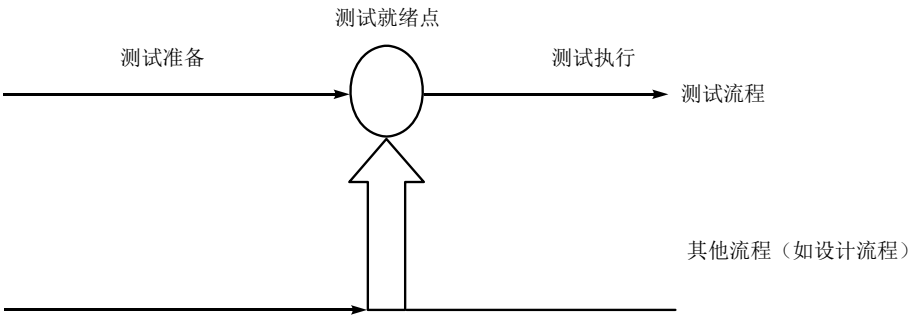


图 2-6 H 模型的验证流程

2.2 软件质量大师的观点

测试是为了保证及提高质量，每个测试人员每天都为质量问题忙碌，大师是怎样阐述提高质量的呢？

1. 戴明理论

1900 年 10 月 4 日在美国爱荷华州，W.爱德华·戴明（W.Edwards Deming）博士诞生，1928 年他在耶鲁大学取得物理博士学位。1950 年戴明博士去日本讲学，并在以后的 40 多年里一直在日本从事质量管理指导工作，日本企业称之为“质量之神”。1980 年 6 月 24 日美国 NBC 播放了举世闻名的纪录片“*If Japan Can, Why Can't We?*”，戴明博士自此成名，并获得“第三次工业革命之父”的美誉。

“PDCA 戴明环”理论是戴明的重要成就之一，它将管理过程分为四个阶段，形成封闭的环路，达到持续改进的效果，如图 2-7 所示。

- ① Plan（计划）：指提高质量的改进计划。
- ② Do（执行）：执行计划。
- ③ Check（检验）：通过检查来判断是否达到期望的结果。
- ④ Action（纠正）：实施纠正行动。

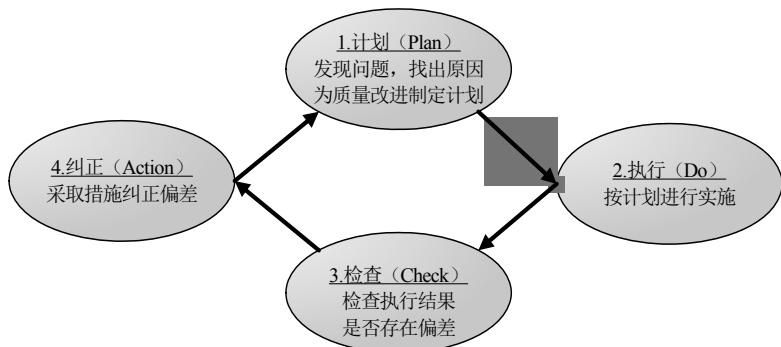


图 2-7 PDCA 戴明环

2. 朱兰理论

约瑟夫·M·朱兰是继戴明之后对日本质量革命具有重大影响的大师，他是最早将帕累托原理引入质量管理的人，被称为品质领域“首席建筑师”。他在 1951 年出版了被称为“质量管理领域中的圣经”的《朱兰质量手册》（Juran's Quality Handbook）。

朱兰认为应该由管理负责的质量问题至少占 80%，剩下 20% 的质量问题才是技术问题。他的“朱兰三部曲”是一直被世人称颂的。

质量计划——为了建立有能力满足质量标准的工作程序，质量计划是必需的。

质量控制——为了掌握何时应该采取措施纠正质量问题就必须开展质量控制。

质量改进——质量改进有利于发现更好的管理工作方式。

3. Crosby 理论

飞利浦·克劳士比（Philip Crosby）一生致力于“质量管理”哲学的发展和应用。1979年他出版了以建议“组织向零缺陷努力”的著称《质量是免费的》（Quality is Free）。他被誉为当代“伟大的管理思想家”、“零缺陷之父”、“世界质量先生”。

飞利浦·克劳士比认为质量就是符合要求的，而不是最好的；质量系统是用来预防的，而不是用来检验的；质量的标准是零缺陷，而不是差不多就好（Close enough is good enough）。

4. 田口宏一理论

田口宏一提出“坚固设计方法”，他认为“质量是设计出来的，而不是检查出来的”。在软件工程中这个观点也非常实用，软件产品的质量是要从源头的需求调研、软件设计抓起，而不能完全依靠测试人员。

“质量最好是通过减少与目标的偏差来获得”，“产品应该对不可控的因素具有免疫力”也是田口宏一提出的观点，而且他还认为可以通过损失函数（Loss Function）来对产品质量进行衡量。

5. 6 σ 理论

在20世纪70年代，摩托罗拉在统计学原理上建立了6 σ 理论。希腊字符西格马 σ 代表“标准差”，在商业活动中，它代表流程与完美的差距。6 σ 的值是99.99966%，也就是说每一百万个产品中有3.4个是次品。从数据上可以看出该管理理论是以追求完美无瑕为最终目标，它在非常多的大型企业中得到运用。

例如在一家普通的制品厂中，要生产一批圆柱形的零件，其中有一个重要参数就是零件的直径。在连续的生产过程中，每个零件的直径是不可能是一模一样的。如果规定的标准 requirements 是1.5分米，那么通过生产线生产出来的铅笔直径可能有1.48分米、1.51分米或者1.49分米等几种不同的数值。材料经过同一个流程，输出的结果会有微小的差异，这原本也是在所难免的。如果零件直径波动在允许的误差范围内，那么零件就是合格的，否则就是次品。对零件直径的控制可以看出，不是看所有产品的平均值，而是看误差的范围究竟有多大。

2.3 常用的验证方法

测试的方法很多，这里介绍几种常用的测试技术，希望测试人员可以学习和领悟到其中的测试技巧。

2.3.1 边界值测试

变量和函数都有自己的边界，能够准确找到各种边界值是软件测试人员所需要具备的基本素质。变量的边界值是其最大值和最小值，函数的边界是一个区域。

例如：函数 F 有两个参数 X 和 Y ，函数 F 的有效值范围就是参数 X 和 Y 的有效值所组成的区间，如图 2-8 所示。

$$a \leq X \leq b$$
$$c \leq Y \leq d$$

边界值测试分为健壮性边界测试和健壮性最坏情况下边界测试。

1. 健壮性边界测试

健壮性边界测试就是取最大值、略大值、正常值、最小值、略小值进行测试，以验证在这五种情况下是否正确，如图 2-9 所示。

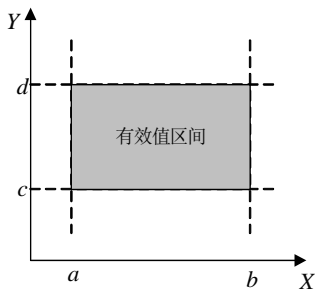


图 2-8 函数有效值的范围

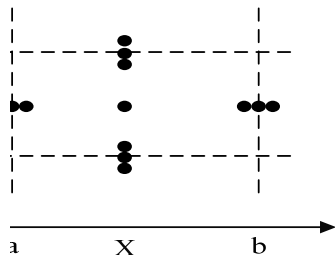


图 2-9 健壮性边界测试

2. 健壮性最坏情况下边界测试

如图 2-10 所示，健壮性最坏情况下边界测试除了要对以上五种境况进行测试外，还要对其进行笛卡儿积计算，以覆盖更多的边界情况。

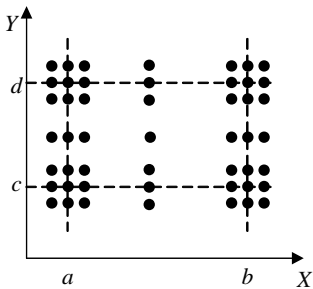


图 2-10 健壮性最坏情况下边界测试

2.3.2 白盒测试

白盒测试是与黑盒测试相对的概念，是对黑盒测试完美的补充。它是指软件测试人员针对可见代码进行的一种测试，因此白盒测试是面向代码的测试、结构化的测试。它根据程序的控制流结构来设计测试用例，对代码中的所有路径进行逐个测试。

为什么要进行白盒测试？这个问题比较复杂，这里做一个简单的比喻，例如一个病人去医院看病，那么医生就相当于软件的测试人员，他们要帮助病人尽快地找到病因。软件程序中的缺陷就好像人身体里的疾病，给人们带来很多困扰和不便，病人来到医院后首先要去挂号，这也相当于在进行软件测试前要先发出一份测试的通知。测试人员进行的黑盒测试就好比医生在接诊时常常使用“望、闻、问、切”的方法，也许某些时候可以发现一些情况，但这些只是从表面现象进行判断，都不能够达到十分准确。要想真正确诊就必须对人体内在的元素进行化验和检查，这也就是软件白盒测试的原理。软件测试人员只有通过代码才能有更多针对性的方法来查找原因，而且可以更加准确地对缺陷进行定位。

白盒测试的任务如下：

- ① 对模块接口进行测试。
- ② 对模块的内部数据结构进行测试。
- ③ 对模块边界条件进行测试。
- ④ 对模块内部的业务逻辑进行分析，独立测试所有路径。
- ⑤ 检查代码是否符合《编码命名规范》。

白盒测试需要注意的事项有以下几点：

- ① 逻辑结构是否完整
- ② 数据类型转换是否安全
- ③ 算法是否合理
- ④ 有无对异常进行捕捉并处理
- ⑤ 是否对必要的资源进行释放
- ⑥ 集合类型的边界是否安全
- ⑦ 对所有逻辑值均需测试 `true` 和 `false`
- ⑧ 输入参数的数量和类型是否匹配

白盒测试讲究的是代码覆盖的情况，目标是把所有代码和各种可能性都完全进行独立的测试。代码中间往往嵌套着各种业务逻辑，由各种流控制语句组合实现，这就可能会有多种情况发生。程序的流程图是对其逻辑进行分析的最好工具，先看一下如图 2-11 所示的代码范例，以及根据其逻辑画出的流程图，如图 2-12 所示。

```
1 using System;
2
3 namespace StatementCoverage
4 {
5     public class Class1
6     {
7         public void mytest(int k, int j, int l)
8         {
9             int x=0, y=0;
10
11             if(k>3 && j<15)
12                 x=y+1;
13
14             if(k==6 || l>10)
15                 x=x+y;
16         }
17     }
18 }
19
20
```

图 2-11 代码范例

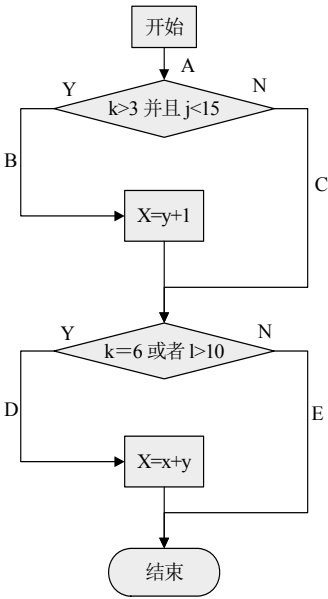



图 2-12 流程图

1. 语句覆盖

语句覆盖（Statement Coverage）从字面上理解就是在白盒测试时程序中每一行代码都检查一遍。

语句覆盖要求设计足够多的测试用例（当然越少越好），使得程序中每条语句至少被执行一次。但它无法实现对所有逻辑分支进行覆盖，因此它是一种最基本的覆盖原则。

**提示**

将表 2-3 中的测试数据带入程序进行运算，该用例执行了代码中的所有语句，显然白盒测试的目的达到了。但仔细研究一下程序就会发现一个问题：如果将一个判断中的“&&”改为“||”，或者将第二个判断中的“||”改为“&&”，该测试用例仍可以覆盖所有的语句。因此语句覆盖无法判断逻辑是否存在的问题，所以语句覆盖也是最弱的一种覆盖原则。


表 2-3 语句覆盖的测试用例

编号	K	J	L	程序执行的路径
a	6	10	15	A→B→D

2. 判断覆盖

在语句覆盖中可以发现即使所有代码都经过测试，但却无法判断逻辑是否正确，因此引入判断覆盖（Decision Coverage）的概念，判断覆盖也称为分支覆盖。所以要对每个 if 语句的逻辑进行独立测试，每个逻辑也就对应了语句中的 if 和 else 分支。

判断覆盖要求设计足够多的测试用例，使得代码中每个判断至少取一次真值，取一次假值，即程序中的每个 if 语句的分支至少执行一次。

**提示**

将表 2-4 中的两组测试数据分别代入程序运行，可以看到两个 if 语句分别都取了一次真值和一次假值。判断覆盖不但满足了逻辑的判断，而且还做到了语句覆盖，因此判断覆盖比语句覆盖稍强一些。

表 2-4 判断覆盖的测试用例

编号	K	J	L	程序执行的路径
a	6	13	10	A→B→D
b	2	20	10	A→C→E

接下来将代码做一点改变，假如将第二个判断条件中“l>10”改成“l<10”，该测试用例仍可以执行并达到预期的测试目的。由此可见判断覆盖无法对判断条件的组合进行检查，不一定能发现某个条件是否存在错误。

3. 条件覆盖

可以发现判断覆盖虽然弥补了语句覆盖的缺点，但它自身还是存在一定缺陷。条件覆盖（Condition Coverage）要求设计足够多的测试用例，使得判断中每个条件的每种可能性至少运行一遍。通过对范例进行分析发现，它是由两个 if 语句构成，每个 if 语句包含了两个判断条件，因此如表 2-5 所示，可以设计一种测试用例，对这些判断条件的每种可能性进行覆盖。

表 2-5 条件覆盖的测试用例 1

编号	K	J	L	程序执行的路径	覆盖条件	覆盖分支
a	6	13	10	A→B→D	T1、T2、T3、-T4	BD
b	3	14	10	A→C→E	-T1、T2、-T3、-T4	CE
c	6	15	11	A→C→D	T1、-T2、T3、T4	CD



首先要对代码的每个 if 语句的条件进行分析，如表 2-6 和 2-7 所示。

表 2-6 条件覆盖的第一个判断

k>3	取真值	T1
k<=3	取假值	-T1
j<15	取真值	T2
j>=15	取假值	-T2

表 2-7 条件覆盖的第二个判断

K=6	取真值	T3
K!=6	取假值	-T3
L>10	取真值	T4
L<=10	取假值	-T4

然后将表 2-8 中的测试数据分别代入程序进行运行。可以看出使用三个用例，即可将四个判断条件的八种情况全部覆盖。不但做到了语句覆盖，而且也覆盖了所有的分支，并且每个分支中每个判断条件的真、假两种可能性也全部进行了测试。

表 2-8 条件覆盖的测试用例 1

编号	K	J	L	程序执行的路径	覆盖条件	覆盖分支
a	3	13	11	A→C→D	-T1、T2、-T3、T4	CD
b	6	16	9	A→C→D	T1、-T2、T3、-T4	CD



再将表 2-9 中的测试数据代入程序进行运行，可以看出使用两个用例，也可以将两个 if 语句中四个判断条件的八种情况全部覆盖。

但该测试用例未能覆盖全部分支，只对第一个条件取假的分支和第二个条件取真的分支进行覆盖，也就是说只覆盖了四个分支中的两个，因此条件覆盖也存在缺陷。

表 2-9 条件覆盖的测试用例 2

编号	K	J	L	程序执行的路径	覆盖条件	覆盖分支
a	6	13	10	A→B→D	-T1、T2、T3、-T4	BD
b	3	14	10	A→C→E	-T1、T2、-T3、-T4	CE
C	6	15	11	A→C→D	T1、-T2、T3、T4	CD

4. 判断-条件覆盖

用户可以知道条件覆盖并不能取代判断覆盖，它们各有优势，这里就将它们进行结合，提供一种“判断-条件覆盖”。

用户需要设计足够多的测试用例，使得判断中每个条件的所有可能性至少出现一次，每个分支本身所有可能的结果也至少出现一次。判断-条件覆盖满足了判断覆盖的原则，同时也满足了条件覆盖准则，弥补了二者的不足。


**提示** 将表 2-10 中两组测试数据代入程序进行运算。可以看出不但满足了分支覆盖的目的，而且也满足了各分支中每个条件的所有可能性。

表 2-10 判断-条件覆盖的测试用例

编号	K	J	L	程序执行的路径	覆盖条件	覆盖分支
a	6	13	15	A→B→D	T1、T2、T3、T4	BD
b	3	16	10	A→C→E	-T1、-T2、-T3、-T4	CE

如果再对代码进行认真分析，会得到在第一个表达式中只有两个条件均满足的情况下才为真，因此当表达式 $K=3$ 为假时，J 的值就不再起作用。但是对于第二个表达式，只要有一个条件满足时，该表达式就为真。因此当表达式 $K=6$ 为真时，L 的值就不再起作用。因此，判断-条件覆盖不一定能检查出表达式中的条件错误。

5. 条件组合覆盖

既然判断-条件覆盖都无法达到最终的效果，那就将“条件覆盖”、“判断覆盖”、“判断-条件覆盖”的目的进行组合，推出另一种覆盖方式——“条件组合覆盖”。

用户需要设计足够多的测试用例，使得每个判断中各条件结果可能性的组合至少出现一次。

各位对代码进行分析，有两个 if 语句，每个 if 语句都包含两个判断条件，因此对同一 if 语句中的判断条件的可能性进行组合，每个判断可以得到四种组合方式。


**提示** 首先要对代码中每个 if 语句的条件进行组合，如表 2-11 和表 2-12 所示。

表 2-11 条件组合覆盖的第一个判断

条件组合 1#	$k>3, j<15$	T1、T2	第一个判断为真
条件组合 2#	$k>3, j\geq 15$	T1、-T2	第一个判断为假

续表

条件组合 3#	$k \leq 3, j < 15$	-T1、T2	第一个判断为假
条件组合 4#	$k \leq 3, j \geq 15$	-T1、-T2	第一个判断为假

表 2-12 条件组合覆盖的第二个判断

条件组合 5#	$k = 6, l > 10$	T3, T4	第二个判断为真
条件组合 6#	$k = 6, l \leq 10$	T3, -T4	第二个判断为真
条件组合 7#	$k \neq 6, l > 10$	-T3, T4	第二个判断为真
条件组合 8#	$k \neq 6, l \leq 10$	-T3, -T4	第二个判断为假

将表 2-13 中的四组测试数据代入程序进行运算。可以看到所有的条件组合均被覆盖。该程序有四条路径，但本用例覆盖了所有条件分支，也覆盖了各种条件的组合，但却少了一条路径 A→B→E。路径能否完全覆盖，在白盒测试中却是一个非常重要的问题。

表 2-13 条件组合覆盖的测试用例

编号	K	J	L	程序执行的路径	覆盖条件	覆盖组合	覆盖分支
A	6	13	15	A→B→D	T1、T2、T3、T4	#1、#5	BD
B	6	16	10	A→C→D	T1、-T2、T3、-T4	#2、#6	CD
C	3	13	12	A→C→D	-T1、T2、-T3、T4	#3、#7	CD
D	3	15	10	A→C→E	-T1、-T2、-T3、-T4	#4、#8	CE

6. 路径覆盖

经过以上五种覆盖方式可以看到其各有千秋。但路径是软件测试最需要关注的地方，因此路径覆盖将是软件测试人员需要十分注意并且常用的一种覆盖方式。用户需要设计足够的测试用例，使其覆盖程序中所有可能的路径，该测试方法可以对程序进行彻底的测试，比前面五种的覆盖面都要全面。

如表 2-14 所示对该测试代码进行分析，它一共有四条路径。

- ① A→B→D
- ② A→C→E
- ③ A→C→D
- ④ A→B→E

表 2-14 条件组合覆盖的测试用例

编号	K	J	L	程序执行的路径	覆盖条件
a	6	13	15	A→B→D	T1、T2、T3、T4

续表

编号	K	J	L	程序执行的路径	覆盖条件
B	4	16	10	A→C→E	T1、-T2、-T3、-T4
c	3	13	12	A→C→D	-T1、T2、-T3、T4
d	4	13	10	A→B→E	T1、T2、-T3、T4



在实际测试工作中，路径其实是一个非常庞大的数字，要想全部覆盖，往往不太现实。如图 2-13 所示，A B A 是一个 20 次的循环，该循环嵌套了几个条件语句，要执行所有可能的路径，其路径数为 $5^{20} + 5^{19} + 5^{18} + \dots + 5^1$ 。特别是在以赢利为目的的项目中，项目进度、成本与质量之间需要进行权衡。

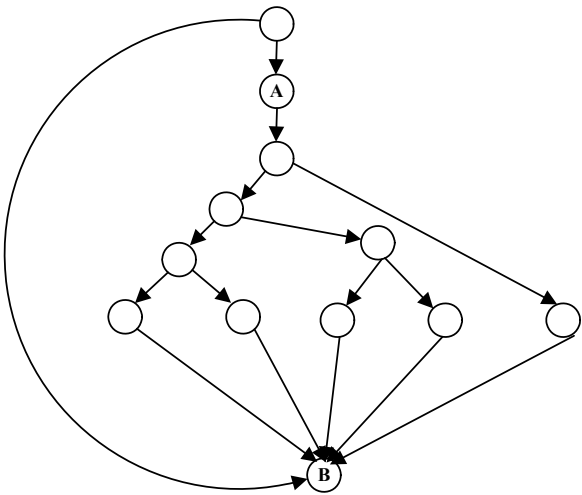


图 2-13 路径覆盖示意图

2.3.3 等价类分法

等价类是相对于穷举法的，它来源于采样的原理。由于不可能用所有可以输入的数据来测试程序，而只能从全部可供输入的数据中选择有代表性的数据进行测试。等价类分法是一种典型的黑盒测试方法，使用该方法时完全不考虑程序的内部结构，只依据程序的规格说明来设计测试用例。它把数目极多的输入数据划分为若干个等价类，在该等价类中各个输入数据对于发现程序中的缺陷都是等效的，当测试某等价类的代表值时就等于对这一类中的其

他值进行了测试。因此可以把全部输入数据合理划分为若干个等价类，在每个等价类中取一个数据作为测试的输入条件，就可以取得较好的测试结果，而且降低了测试的成本。

有效等价类对于用户需求规格说明书来说是合理的、有意义的输入数据的集合。无效等价类与之正好相反。等价类划分有以下几个规则：

① 在输入条件规定了取值范围或取值个数的情况下，可以确定一个有效等价类和两个无效等价类。如图 2-14 所示，有效等价类 x 是 $20 \leq x \leq 30$ ，两个无效等价类是 $x \leq 20$ 和 $x \geq 30$ 。

② 在输入条件规定了输入值的集合或者类似于“必须如何”的条件下，可以确定一个有效等价类和一个无效等价类。如图 2-15 所示，一个有效等价类和一个无效等价类分别填写正确的邮件地址和错误格式的邮件地址。

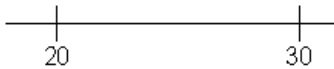


图 2-14 等价类划分原则 1

E-mail:

图 2-15 等价类划分原则 2

③ 在输入条件是一个布尔值的情况下，可确定一个有效等价类和一个无效等价类。

④ 在确定了输入数据的一组值（假定 n 个），并且程序要对每一个输入值分别处理的情况下，可确立 n 个有效等价类和一个无效等价类。如图 2-16 所示， n 个有效等价类是每个输入的内容都填写正确的信息，一个无效等价类是所有输入的内容都填写错误的信息。

部门:

Date:

E-mail:

Notes:

图 2-16 等价类划分原则 4

2.3.4 压力测试

压力测试是检测在巨大的工作压力负荷下应用程序的运行情况。在项目开发过程中有时会遇到软件测试人员按照《需求规格说明书》进行测试的情况，程序可以正常运行并且毫无缺陷。然而当产品进入用户试运行，由于用户数量的增多有时会出现产品崩溃或系统瘫痪的情况，有时只能重新启动服务器或者数据库程序才可以继续正常使用，这说明软件产品可能存在性能问题，需要进行性能测试。

在通常情况下，软件测试人员只需要模拟不同数量的用户对软件产品进行测试，并对各性能指标进行分析就可以找到软件产品的瓶颈，或者得到该产品能够支持的最大负载。在一般情况下软件测试人员使用排除法来查找哪里出现了瓶颈。首先应该从测试服务器的硬件配置入手分析，然后再来排除网络的问题，接下来是排查服务器上操作系统的各种配置是否存在瓶颈。如果以上三项都经过逐一检查，那接下来就要对软件产品所使用的中间件进行检查，例如 SQL Server 数据库、某个“Com+”或某个 Web Service，最后才要对具

体的代码、存储过程等进行分析。

Mercury LoadRunner 是一款非常全面的性能测试工具，它可以对软件产品进行负载和压力测试，以防止产品因硬件或架构设计等因素带来的各种性能瓶颈。它通过模拟成千上万的虚拟用户以及对性能指标的实时监测，使软件测试人员可以通过对测试报告的分析 and 诊断，找到优化系统解决瓶颈的最佳方案。如图 2-17 所示，软件测试人员可以按照以下步骤来运用 LoadRunner 进行性能测试。

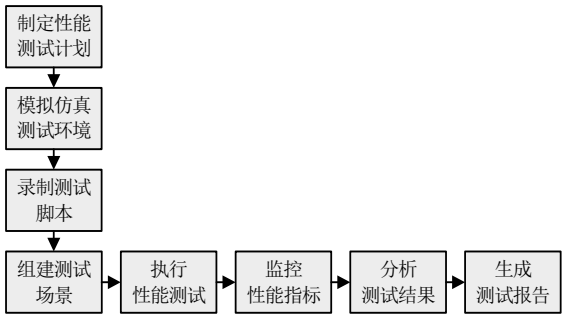


图 2-17 LoadRunner 性能测试步骤

在进行压力测试前要先了解以下概念。

- ① 响应时间是对页面请求做出响应所需要的时间，在对 Web 系统进行测试中经常用到。响应时间还可以分为呈现时间和系统响应时间两个部分：
 - 呈现时间是指客户端数据在页面上显示出来的时间。
 - 系统响应时间是指服务器从接收到请求开始到做出响应的的时间。
- ② 并发用户数与同时在线数不是同一个概念，例如一个软件系统平时最多有 5000 人同时在使用，但这 5000 人并不是并发用户，因为他们分别在使用不同的模块，在操作不同的功能，也许其中一部分人只是登录了系统而什么操作都没有进行。
- ③ 吞吐量是单位时间内系统处理客户请求的数量，直接体现出软件性能的承载能力。
- ④ 页交换是指以页面为单位，将固定大小的数据从内存移到硬盘的过程，其目的是为了释放内存空间。

在进行压力测试时首先应该对服务器的内存进行监控，因为软件系统的性能瓶颈通常来自于内存的不足。如果发生系统“页交换”频繁，则说明内存出现瓶颈。软件测试人员可以通过以下参数进行判断：

- **Available Mbytes:** 可用物理内存数是能立刻分配给一个进程或系统使用的内存数量。在通常情况下至少要有 10%的可用物理内存，否则说明计算机上总的内存数可能不足，或某程序没有释放内存。如图 2-18 所示，对该参数进行监控时可以使用 Windows

的任务管理器。

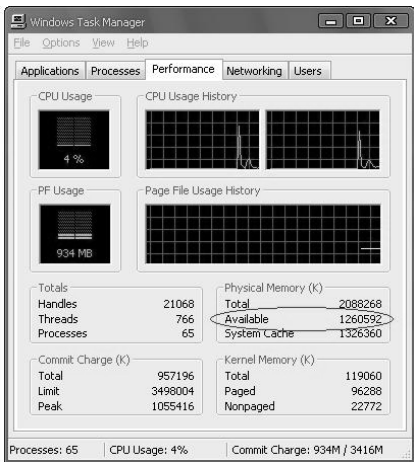


图 2-18 Windows 任务管理器

在 Loadrunner 的“Windows Resource”报表中可以添加对“Available Mbytes”进行监控的参数，如图 2-19 所示，在 512MB 内存的系统中，平均可用内存为 42MB，即小于 10%，系统内存出现瓶颈。

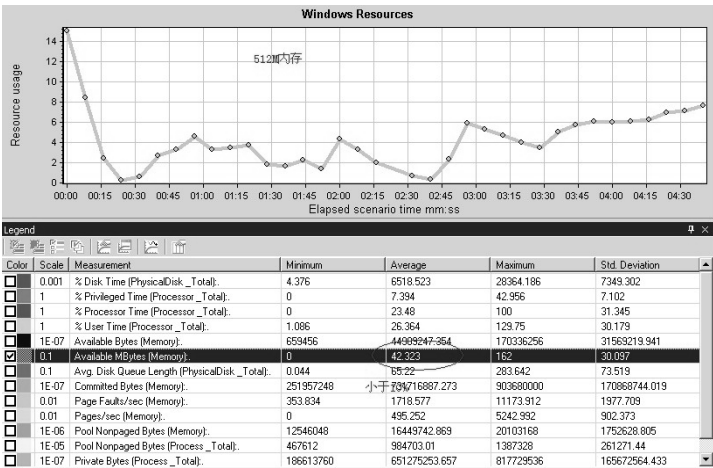


图 2-19 Windows Resource 资源报表

在压力测试中，硬盘的性能往往也会成为系统的瓶颈，一般可以通过以下参数进行综合判断：

- %Disk Time: 该参数是指磁盘驱动器忙于为处理读或写的请求所用时间的百分比。

- Avg. Disk Queue Length: 指读取和写入请求（队列）的平均数。

在当今硬件设备迅速发展而硬件价格迅速降低的时代，硬盘出现瓶颈的概率越来越小。但系统在“页交换”的过程中是将数据在内存与硬盘之间进行迁移，为了准确判断系统的瓶颈是出现在内存上还是硬盘上，就需要将内存的参数与硬盘的参数进行综合判断。如果“页面”读取操作速率 Page Reads/sec 很低，同时%Disk Time 和 Avg.Disk Queue Length 的值很高，则可能是磁盘瓶颈，反之则是内存出现了瓶颈。

对内存和硬盘的性能进行分析后，还需要对 CPU 的性能进行判断。

- %Total Processor Time: 系统上所有 CPU 忙于处理非空闲线程的平均时间百分比，反映了 CPU 的平均繁忙程度。
- %Processor Time: 在以上参数基础上，如果%Processor Time 的值持续超过 90%，则表明 CPU 出现瓶颈，此时就可以考虑增加或更换处理器。

使用 Loadrunner 对软件进行性能分析时可以按照以下步骤进行：

STEP 01 确认负载发生器的 Windows Resource 没有出现异常。如果负载发生器的性能都出现问题，那么本次性能测试的基础就不正确。重点检查负载发生器的内存和 CPU 是否出现瓶颈。

STEP 02 如图 2-20 所示，分析 Loadrunner 生成的 Summary Report。确认本次性能测试的用户是否全部运行，如果要求的用户数都没有执行完毕，那么本次测试的结果将不符合设计的要求。如果发现问题，则要打开 Running Vusers 图查找原因。

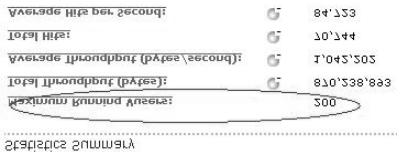


图 2-20 分析 Summary Report 中虚拟用户的执行情况

STEP 03 分析 Summary Report 中事物的执行情况。如图 2-21 所示，检查每个具体业务逻辑的 Transaction，看看是否有失败的情况。如果有失败的，则要对它进行重点分析。

Transaction Summary

Transactions: Total Passed: 1,984 Total Failed: 0 Total Stopped: 8 Average Response Time

Transaction Name	SLA Status	Minimum	Average	Maximum	Std. Deviation	90 Percent	Pass	Fail	Stop
Action Transaction		328.729	376.03	485.668	52.63	469.738	392	0	8
Edit		82.165	94.571	106.088	10	104.593	392	0	0
Login		25.651	31.098	36.388	5.145	36.299	400	0	0
Search		66.881	77.212	87.536	10.23	87.492	400	0	0
vuser_end Transaction		0	0	0	0	0	200	0	0
vuser_init Transaction		1.551	3.282	4.244	0.744	4.017	200	0	0

Service Level Agreement Legend: Pass Fail No Data

图 2-21 分析 Summary Report 中每个事物的执行情况

STEP 04 如图 2-22 所示，分析 Summary Report 中每个具体事物的“平均响应时间”和“90%的最大响应时间”，目的是与需求文档中客户要求的响应时间进行对比，以判断是否满足客户的要求。如果事物响应时间过长，那么就需要通过事物类型的图表做进一步分析。



图 2-22 分析 Summary Report 中每个事物的响应时间

STEP 05 分析“Time to First Buffer Breakdown”报表，如图 2-23 所示，可以知道软件系统的性能瓶颈处在哪里，这样就可以缩小分析的范围。如果“Network Time”的值过大，则说明网络传输速度出现了瓶颈；如果“Server Time”的值过大，则说明服务器端出现了瓶颈。

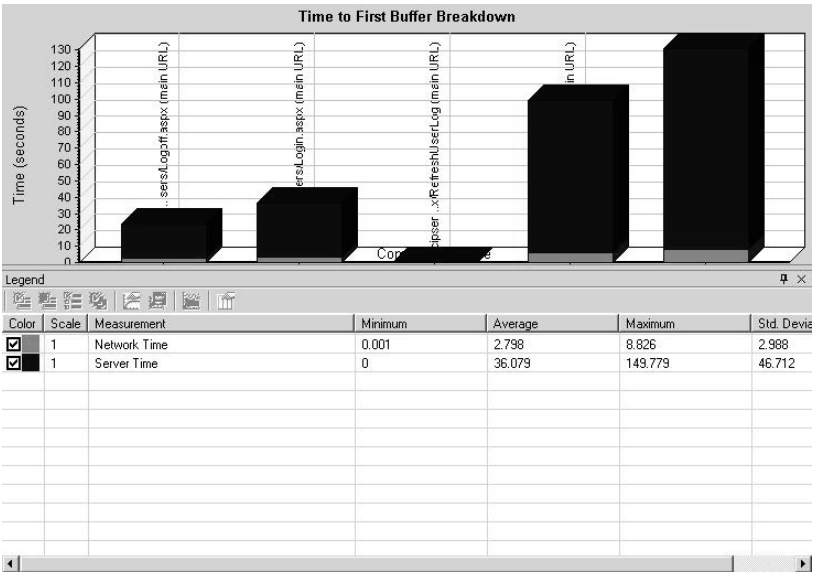


图 2-23 Time to First Buffer Breakdown 报表

STEP 06 将“Average Transaction Response Time”报表和“Running Vusers”报表合并进行分析效果会更好，通过判断事物的平均响应时间是否随虚拟用户的增加而线性变短来判断系统是否出现了性能瓶颈。如图 2-24 所示，在正常情况下当一定程度的用户并发时，事物的平均响应时间不会出现太大波动，否则如图 2-25 所示，事物响应时间会出现大幅波动。

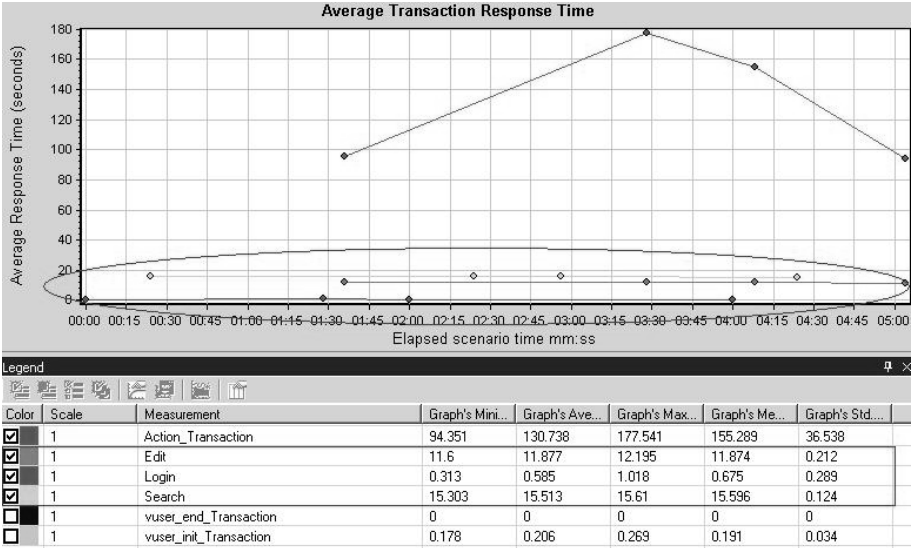


图 2-24 事物平均响应时间正常状态图

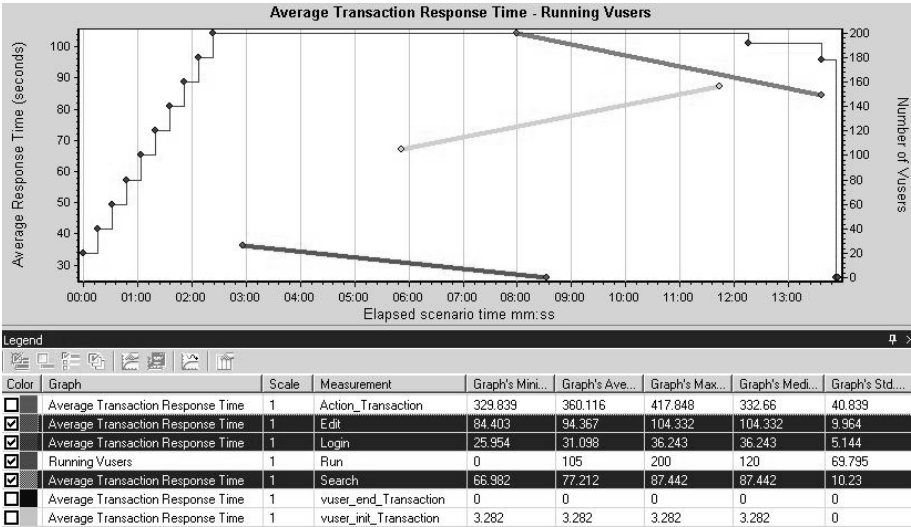


图 2-25 事物平均响应时间异常状态图

STEP 07 在查找软件程序瓶颈的时候，应该重点关注如表 2-15 所示的参数。

表 2-15 对网页进行性能分析时的重要参数

名 称	描 述
First Buffer	从客户端发出 HTTP 请求开始计算，到客户端成功收到来自服务器发出的响应所经过的时间。是判断服务器瓶颈还是网络瓶颈的指示器
Receive	标示从服务器接收到最后一个字节并完成下载之前经过的时间。是衡量网络速度的指示器
Client Time	因浏览器思考时间或其他客户端软件有延迟而导致客户端的请求发生延迟的平均时间。是判断客户端性能的指示器

如图 2-26 所示，在对“Page Download Time Breakdown”报表进行分析时，可以通过“Breakdown Tree”的逐级分解找到软件程序瓶颈的根源，即消耗时间最多的链接或组件。

如图 2-27 所示，在“Page Download Time Breakdown”报表中“First Buffer”和“Receive Time”花费的时间最多，也就是说代码的算法有待提高。

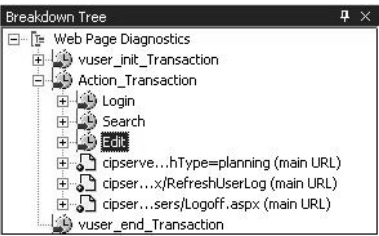


图 2-26 Breakdown Tree

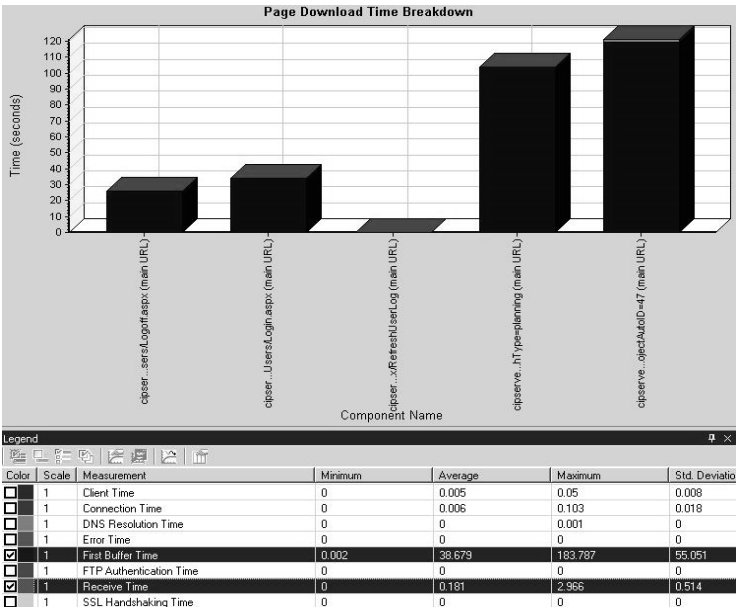


图 2-27 Page Download Time Breakdown 报表

如图 2-28 所示，在“Page Component Breakdown”报表中某个页面组件占用了 24.35%

统用户每次在线的时间长度为 4 小时。

- T 是考查时间的长度,也就是系统在线时间的长度。该系统在线的时间长度为 8 小时。根据以上公式进行计算,该 MIS 系统的最大并发数是 $C=400 \times 4/8=200$ 个。

2.5 小结

本章的内容可以分为两个部分:第 1 个部分是将各种测试手段进行抽象,归纳总结为“验证的准备工作”、“执行验证”和“纠正措施”3 个步骤;第 2 个部分是对各种测试方法的介绍,希望读者可以理解软件测试的原理。

2.6 思考题

1. 软件测试抽象后的 3 个步骤是什么?
2. 戴明环的 4 个要素是什么?
3. 单元测试、集成测试、系统测试、验收测试的依据和准则是什么?

3

第3章

软件质量管理的信任机制——确认

人们的日常生活往往离不开对各种各样的事情进行确认，例如：当使用信用卡的时候，服务员会要求顾客确认银联回执单上的金额，然后在上面签字；当顾客在银联回执单上签字后，服务员还要确认签字笔迹是否与信用卡上的相符；当一对恋人打算结婚的时候，他们都会去民政局进行婚姻登记，以在法律上确认他们的合法关系，当然在婚姻登记时也需要男女双方签字确认。

在软件研发过程中也离不开各种确认的工作，例如：甲乙双方签订合同时，要对合同上的金额、完工时间、项目范围等内容进行确认，确认后要双方签字、盖章；当需求人员在完成《软件需求说明书》后，为了减少需求的变更，往往也会给客户进行确认。

由此可见，确认是一种行为，该行为的方式有很多，既可以通过口头方式进行确认，也可以通过书面形式进行确认。确认的深层含义是承诺，换句话说一个人的承诺是通过确认的方式来体现的。例如：顾客不在银联回执单上签字，那么就代表顾客否定了本次交易，这是一种相反的承诺，那么银行就会按照顾客的这种承诺拒绝付款给商家；当一对恋人没有进行婚姻登记，那么在法律上也就没有给彼此一个共同生活的承诺，因此他们还有权力选择他人；在软件研发过程中如果客户没有对《软件需求说明书》的内容进行确认，也就是他没有给出承诺，那么再发生需求变更时他也不会感到愧疚。

3.1 软件确认管理的概述

确认（Validation）简称 VAL，确认管理是软件工程体系中一名新成员，它与配置管理、风险管理、度量管理等分支同等重要，是软件质量体系中不可或缺的环节。

确认是指对软件研发生命周期中某个过程所产出的工作产品进行的审查，这些工作产品可以是《软件需求说明书》、合同等文档，也可以是开发出来的组件或最终产品，甚至可以是某个生命周期阶段进行的整体审查。

确认的目的就是确保某个过程或阶段“做对的工作产品”，并使它符合使用者的期望，并且只有通过审查后的工作产品才能交付给“使用者”使用。

在软件研发过程中有两个重要的确认过程是众所周知的，一个是“客户”对《软件需求说明书》的确认，另一个是项目组开发出来的最终产品要在客户现场进行验收测试，以确认该产品是否符合“客户”的需要。这两个确认都是针对客户方的，但是在确认管理过程中却是不使用“客户”两个字的，而用“使用者”来代替“客户”，这是为了避免广大软件从业人员对确认过程的误解。《软件需求说明书》是软件项目范围的依据，它用来描述软件产品的功能，软件产品的最终“使用者”就是“客户”；验收测试的目的就是确保产品达到“客户”也就是最终“使用者”的要求。但在软件确认管理中并不是只有“客户”才需要对项目的工作产品进行确认，项目组或公司内部同样需要对某些工作产品进行确认，而这种确认往往非常关键，但进行确认的人却不是合同的甲方，因此在软件确认管理中要用“使用者”这个名称来对它进行代替。

那么什么时候才会出现项目组内部的确认呢？很多人对这个事情都有疑问，这是可以理解的，因为在早期的软件工程中谈及确认管理的内容是非常少的。但项目组内的确认工作是天天都在进行的，例如：对《概要设计》文档进行评审并且合格通过后，与会人员都会在评审记录上签字。这个过程中就“包含”了确认的内容。但有人又会说同行评审是“验证”的过程，怎么会包含确认的内容呢？大家可以想想，首先确认的目的是承诺，那么签字就代表了与会人员对《概要设计》文档的正确性进行了承诺。其次参加本次评审的人员中一定会有软件开发人员，软件开发人员将是这份《概要设计》文档的“使用者”，只有“使用者”对该工作产品的质量进行确认后才能被使用。因此，在对《概要设计》文档进行评审时，这个过程除了对《概要设计》文档的内容进行验证，与会人员中的“使用者”还要对其内容是否符合要求并且是否可以指导软件开发人员的工作进行确认。

由此可见，在软件生命周期内凡是一个环节“输出”的工作成果都将成为后续环节的“输入”，那么上一个环节的生产者要承诺该工作产品是符合质量要求的，后续环节的“使

用者”也要对其工作产品进行确认。这就好比“亲兄弟明算账”，通过这样的方式来建立相互间的信任关系。

3.2 软件确认流程及最佳实践

为了确保对工作产品确认的效果，通常建议该工作产品在仿真环境下进行审查，因此建立确认的环境是确认管理中的一个部分。一个软件项目所产出的工作产品非常多，仅配置项列表中的内容就有几十项，项目组需要在项目计划阶段识别所需进行确认的工作产品。确认是以使用者的视角来对工作产品进行审查，因此要在制订项目计划时就确定哪些项目关系人要对哪些工作产品进行确认。接下来我们对确认管理的流程和最佳实践进行举例讲解。

3.2.1 确认的准备工作

确认工作在准备阶段包括以下3个方面的内容，这些内容都应该在项目计划阶段完成：

- ① 选择需要确认的工作产品与产品组件
- ② 建立和维护确认环境
- ③ 建立确认的流程及准则

1. 选择需要确认的工作产品与产品组件

在选择需要确认的工作产品和产品组件时，可以根据项目的生命周期模型，并配合项目配置项列表来进行识别。配置项列表中的内容都是项目关键的工作产品，因为配置项是项目基线的组成部分，虽然并不是所有配置项都需要进行确认，但是确认管理的工作还需要很多资源、时间和成本的投入，这要根据项目的实际情况进行确定。

在识别完待确认的对象后就应该为它制订相应的确认方法，并确定参与确认的角色。软件项目中确认的方法有以下两大类，软件生命周期中常见的确认内容及方法如表3-1所示。

- ① 对文档类型的工作产品进行确认，通常可以与其文档的评审合并进行。
- ② 对产品或产品组件进行确认时，通常可以与单元测试、集成测试、系统测试和验收测试合并进行。

表 3-1 软件项目中参加的确认内容及确认方法

项目生命周期	确认内容	确认方法	确认目的	确 认 人
需求阶段	需求调研计划	评审	确保需求调研计划时间安排合理	需求调研人员
			承诺可以按计划的时间参加需求调研的活动	客户

续表

项目生命周期	确认内容	确认方法	确认目的	确 认 人
需求阶段	软件需求说明书	评审或原型展示	承诺需求尽量不发生变更	客户
			确保软件功能可以实现	项目组成员
	系统规格说明书	评审或原型展示	承诺需求尽量不发生变更	客户
			确保软件功能可以实现	项目组成员
计划阶段	项目过程定义书	评审	确保所定义的过程是合理的	项目组成员
	项目估算表	评审	确保项目估算的过程是合理的	项目组成员
	项目计划及其从属计划	评审	承诺可以提高所需的资源	公司高层
			确保项目计划是合理的	项目组成员
设计阶段	概要设计说明书	评审	承诺设计的内容合理有效	软件设计人员
			确保概要设计的内容可以实现	软件开发人员
	详细设计说明书	评审	承诺设计的内容合理有效	软件设计人员
			确保概要设计的内容可以实现	软件开发人员
	产品集成方案	评审	承诺产品基础的方案是合理有效的	软件设计人员
			确保产品集成顺序是合理的	软件开发人员
编码阶段	产品组件	单元测试	承诺代码的质量是合格的	软件开发人员
			确保代码的功能是正确的	软件测试人员
	集成后的产品或组件	集成测试	承诺产品或组件的质量是合格的	软件开发人员
			确保产品或组件的功能是正确的	软件测试人员
系统测试阶段	产品或组件	系统测试	承诺产品的质量已经符合要求	软件测试人员
			确认产品是否可以发布	项目经理
用户验收阶段	产品	验收测试	承诺软件产品已经完成并且达到质量标准	项目经理
			确认产品是否可以验收,项目是否可以结束	客户

在项目计划阶段通过对配置项列表中的配置项进行识别，挑选适当的工作产品在项目过程中进行确认，并将挑选出来的内容记录在确认清单或项目计划中，其流程如图 3-1 所示。

2. 建立和维护确认环境

确认工作的开展最好是在“使用者”的环境下进行，只有这样才能证明该工作产品的质量和功能是否符合“使用者”的要求。但在软件研发过程中这个前提条件并不一定完全可行，在建立确认环境时往往也要考虑确认的方法。例如：要对《软件需求说明书》进行

确认，确认的方法是“评审”，开评审会所需要的环境通常是一间会议室，最好有白板、各种颜色的水笔、投影等设备，不管是甲方还是乙方召开《软件需求说明书》的评审，这些配备都是相同的。再例如对开发阶段集成后的产品或组件进行确认，往往是通过执行集成测试用例来完成的，由于确认的对象是代码，所以集成测试用例通常是由白盒测试技术实现的。在进行此种确认时，软件测试人员是该工作产品的“使用者”，但该确认的方法却是一种开发的技术，所以在软件测试人员的系统测试环境中是无法进行的。

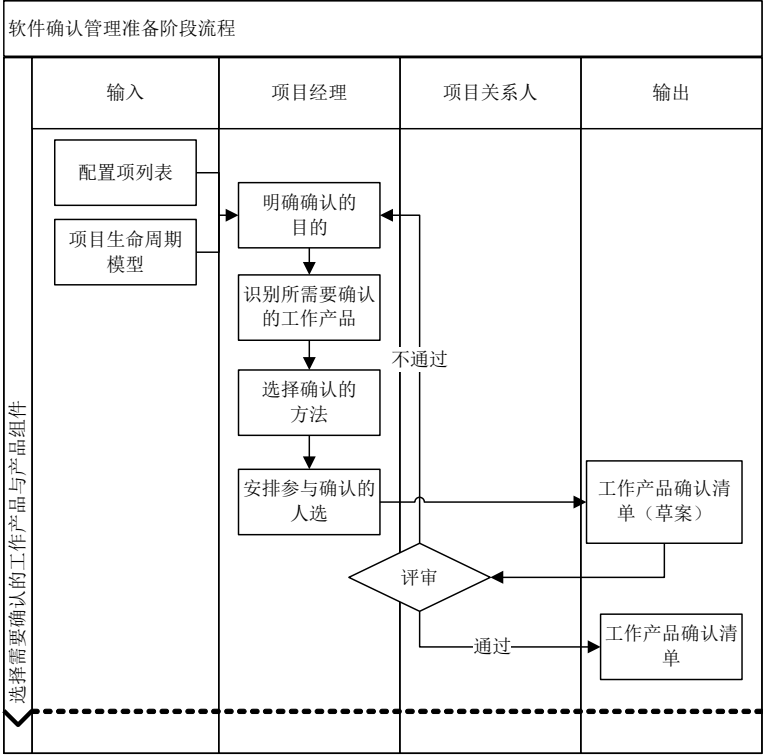


图 3-1 选择确认的产品

“环境”在软件工程中包含了两方面的内容：一个是以硬件设备为主的“硬环境”；另一方面是确认流程和准则的“软环境”。当项目组要对某一个工作产品开展确认活动时，制订配套的流程和准则是必不可少的。如果通过评审的方式进行确认，那么评审的议程应该提前制订，评审过程中的评判标准需要提前制订，否则就会出现无休止的争论。如果通过技术手段对工作产品进行确认，那么部署该工作产品的步骤要提前制订，否则产品部署出现问题，那么确认也就无法进行。软件研发过程中常用的确认环境如表 3-2 所示。

表 3-2 软件研发过程中常用的集成环境

项目生命周期	确认内容	确认方法	确 认	
			硬环境	软 环 境
需求阶段	需求调研计划	评审	白板、各种颜色的水笔、投影等设备	同行评审的具体议程； 同行评审的标准流程； 同行评审的方法； 确认的准则
	软件需求说明书	评审或原型展示	白板、各种颜色的水笔、投影等设备； 电脑、原型展示的环境	同行评审的具体议程； 同行评审的标准流程； 同行评审的方法； 确认的准则； 原型展示的流程； 确认原型的标准
	系统规格说明书	评审或原型展示		
计划阶段	项目过程定义书	评审	白板、各种颜色的水笔、投影等设备	同行评审的具体议程； 同行评审的标准流程； 同行评审的方法； 确认的准则
	项目估算表	评审		
	项目计划及其从属计划	评审		
设计阶段	概要设计说明书	评审		
	详细设计说明书	评审		
	产品集成方案	评审		
编码阶段	产品组件	单元测试	(集成测试环境) 独立的硬件设备； 配套的操作系统、开发环境、单元测试环境	代码及单元测试用例或集成测试用例部署的流程和验证方法； 设计文档和接口说明书； 确认的准则
	集成后的产品或组件	集成测试		
系统测试阶段	产品或组件	系统测试	(系统测试环境) 独立的硬件设备； 配套的操作系统、补丁； 产品运行所需的其他应用程序	系统测试的用例； 自动化测试的脚步； 系统测试的流程； 确认的准则
用户验收阶段	产品	验收测试		验收测试用例； 验收测试的流程； 确认的准则

在项目计划阶段制订确认环境时有可能会引发“Make or Buy”的决策或其他方面的变更。例如某项目要对代码进行确认，但是没有独立的编译服务器或日构建服务器，此时就会导致采购的发生，因此也会造成项目预算的变更。

确认工作所使用的环境是确认的约束条件，同样也是项目约束条件之一，因此项目经理要在项目进度计划中增加相关的活动并分派相应的资源。当发现由确认导致的采购时，就需要对此约束按照项目监控的流程进行管理，否则确认环境不能按时到位，会影响项目的进度和产品的质量。建立确认环境的流程如图 3-2 所示。

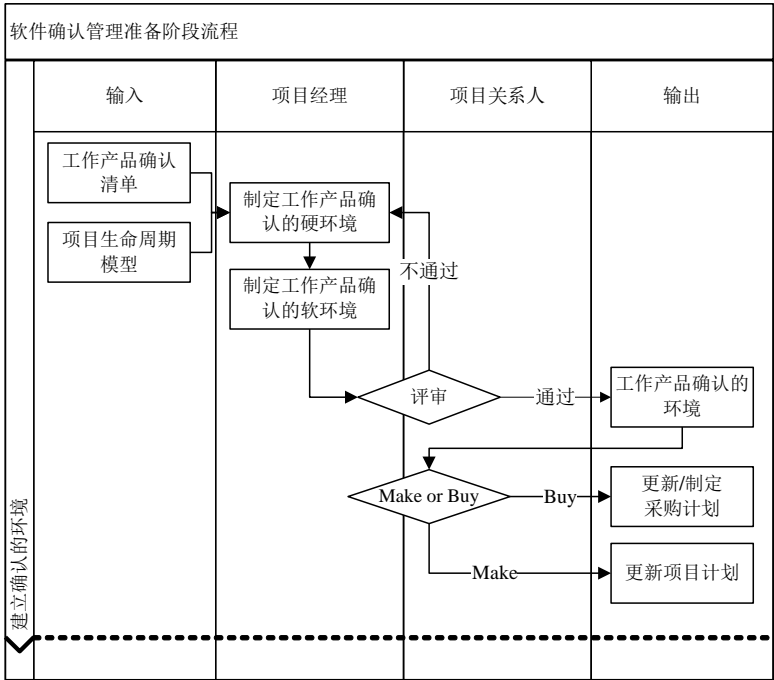


图 3-2 建立确认的环境

3. 建立确认的流程及准则

在讲述建立确认环境时特别提到了“配套的流程和准则”，除了部署工作产品和搭建确认环境的流程外，还包含了判断本次确认是否通过的准则。这些判断的准则往来源于：

- 产品或产品组件的需求
- 国际或行业的标准
- 客户方验收的标准
- 项目绩效的评判标准

不同的软件公司对质量的要求是不同的，因此在制订确认准则时也不尽相同，一般的确认准则如表 3-3 所示。

表 3-3 工作产品确认准则

项目生命周期	确认内容	确认方法	确认准则
需求阶段	需求调研计划	评审	客户方同意并签字确认
	软件需求说明书	评审或原型展示	客户方同意并签字确认； 《软件需求说明书》中的每个功能都必须在评审中覆盖到； 在评审时发现的严重和较严重级别的缺陷必须修复
	系统规格说明书	评审或原型展示	与会人员一致同意并签字确认； 《系统规格说明书》中的每个功能都必须在评审中覆盖到； 在评审时发现的严重和较严重级别的缺陷必须修复
计划阶段	项目计划及其从属计划	评审	项目组成员要同意并签字确认； 公司高层领导要签字确认
设计阶段	概要设计说明书 详细设计说明书 产品集成方案	评审	软件开发人员要同意并签字确认； 设计文档中的每个方法在评审时要被覆盖到； 在评审时发现的严重和较严重级别的缺陷必须修复
编码阶段	产品组件	单元测试	软件开发人员要同意并签字确认； 单元测试用例执行率要达到 100%； 单元测试代码行覆盖率平均要达到 40%； 单元测试中所发现的所有缺陷必须被修复； 单元测试用例执行结果必须全部为通过
	集成后的产品或组件	集成测试	软件测试人员要同意并签字确认； 集成测试用例执行率要达到 100%； 集成测试代码行覆盖率平均要达到 30%； 集成测试中所发现的所有缺陷必须被修复； 集成测试用例执行结果必须全部为通过
系统测试阶段	产品或组件	系统测试	项目经理或软件测试经理要同意并签字确认； 系统测试用例执行率要达到 100%； 产品功能覆盖率要达到 100%； 系统测试中所发现的严重或较严重级别的缺陷必须修复； 系统测试中所发现的严重级别较低的缺陷必须修复 80%
用户验收阶段	产品	验收测试	客户要同意并签字确认； 验收测试用例执行率要达到 100%

确认管理是确保“你做了正确的事”，确认与验证活动经常同时进行，而且可能使用相同的环境。确认的目的是让问题在项目生命周期中尽早地被发现，避免影响后续的任务而造成返工。当问题被发现后，项目管理人员就可以参考项目监控、需求开发、风险管理等其他软件研发的流程对它进行解决。制订确认准则的流程如图 3-3 所示。

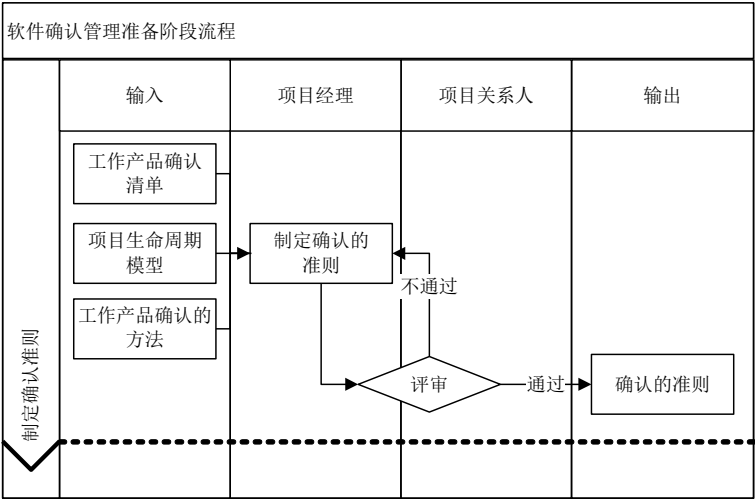


图 3-3 制订确认的准则

识别确认的对象、制订确认的方法、建立确认的环境、定义确认的准则都是确认准备阶段的工作，其目的是为了“使用者”更好地接受放置在确认环境中工作产品的表现情况。

3.2.2 执行确认

在确认管理的准备工作完成以后，就将按照既定的流程和准则在确认环境中执行并收集确认的结果。然后将确认的结果与评估的准则进行比较，当发生偏差时应该及时进行识别并制订相应的措施，最后根据偏差的程度判断确认工作是否还需要继续进行，其流程如图 3-4 所示。确认过程中的偏差往往有以下 3 种可能：

- ① 工作产品质量问题。
- ② 确认环境没有搭建好，而导致工作产品在该环境中出现偏差。
- ③ 制订的确认准则不合理。

3.3 软件确认过程中常见问题及案例分析

软件确认过程是以往软件工程中讲述不多的内容，但在软件研发过程中很多产品质量缺陷、项目进度偏差、项目成本偏差都是由于确认工作没有做好而导致的。以下通过几个案例来对它进行深入分析。

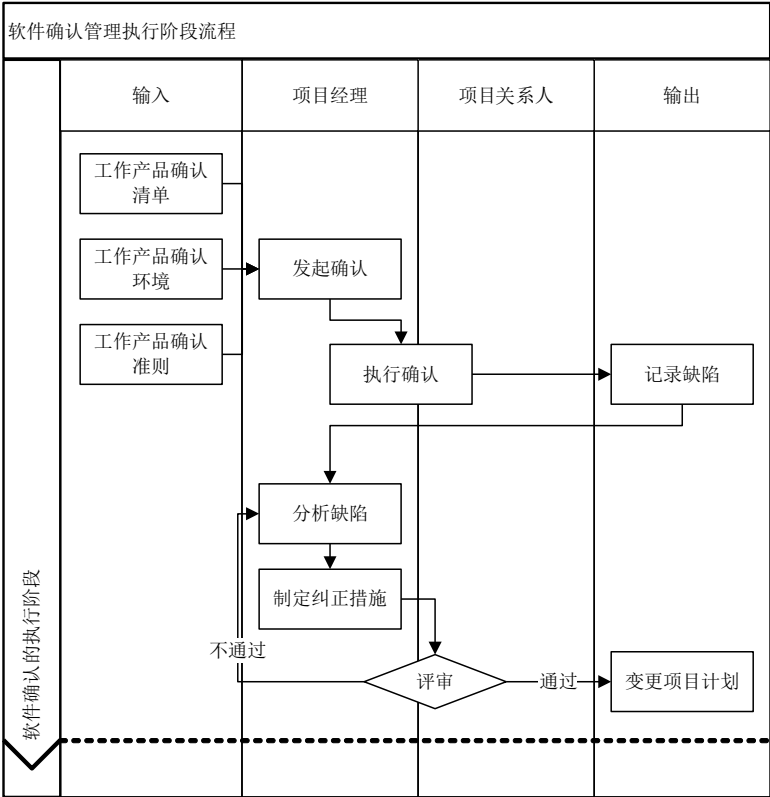


图 3-4 软件确认的执行过程

3.3.1 为什么开发和测试之间总是反复

【案例】

最近某软件公司 GIS 项目组负责人小黎头痛不已，项目已经进展到系统测试阶段，软件开发人员提交给测试组的产品总是无法通过系统测试，甚至一天出现两三次产品内部的发布，软件开发和测试人员都被加班压得透不过气来。软件开发和测试人员之间的埋怨也越来越多，开发人员认为软件测试人员在挑他们的毛病。软件测试人员总是觉得产品质量实在太差，还没有怎么测试系统就不能使用，这样的产品就不应该发布。可是软件开发人员却认为他们做过了单元测试和集成测试，所以提交的产品质量是合格的。

项目负责人小黎觉得这个问题必须尽快解决，开发与测试之间的反复已经导致项目延期了一周的时间，以这样的情况发展下去还有可能恶化。更重要的一点是项目团队成员之间出现了矛盾，软件开发和测试人员之间已经越来越缺乏信任，这样下去将导致项目彻底崩溃。

【分析】

项目负责人小黎找到项目总监张经理，希望得到他的帮助。项目总监经过对项目组成员的访谈以及实际查看了项目的代码后，发现项目组发生的问题都是因为没有对工作产品进行确认而导致的。

本着“亲兄弟明算账”的原则，软件开发人员必须证明他们提交的工作产品已经符合了质量的要求，软件测试人员也要确认开发人员所讲的是否是真实的。公司已经规定了单元测试代码平均覆盖率至少为 60%，集成测试用例代码平均覆盖率至少为 45%，那么确认的准则已经存在，就应该按照此准则进行确认。

小黎按照张总的要求对项目组下达了对每次发布必须进行确认的任务，这下产品内部的发布戛然而止，原因是开发人员注意到他们的单元测试用例和系统测试用例很多都是重叠的，代码覆盖的比例达不到公司的要求，因此他们需要增加新的测试用例。在随后的日子中通过增加单元测试用例和集成测试用例又发现了很多产品质量的缺陷。

通过采取对工作产品进行确认后，产品只经过了两轮的反复就通过了系统测试，而且开发和测试人员之间的矛盾也大大降低了。

3.3.2 确认是对需求变更的约束

【案例】

某软件公司人力资源管理系统的项目经理小白非常开心，因为他的项目已经顺利通过验收。虽然项目过程中发生了一大一小两次需求的变更，但是项目的进度、成本并没有发生什么偏差。公司领导让小白对他的项目进行总结，并且将心得体会与其他项目组成员进行分享。小白在他的项目经验总结中就提到了一点：“需求的确认工作是对需求变更的约束”。

【分析】

小白在项目需求调研阶段就把《软件需求说明书》拿出来让客户进行了确认，客户通过对需求说明书的评审了解到他们所提出来的内容已经完全包含在《软件需求说明书》中，由于《软件需求说明书》的内容非常细致，因此客户对本次合作充满信心，所以很痛快地就进行了确认。但在设计和编码开发阶段分别发生了一次需求的变更，主要原因是客户方忽略了一些需要实现的功能而没有告诉项目组，但是客户又对《软件需求说明书》进行了确认。由于第一次变更的影响不大，小白决定免费为其进行修改，并且将变更后的《软件需求说明书》给客户进行再次确认，客户对小白的工作态度非常满意。

当在编码阶段发生再次需求变更时，小白将变更的影响告知了客户，并且提示客户方已经先后两次对需求进行了确认，希望可以在此次变更增加相应费用。客户方的代表也觉得自已有一定责任，所以就相应延长了项目的时间并增加了项目的费用，直到最后项目顺利交付。

因此小白在本次项目中最大的收获就是体会到软件确认管理给项目需求变更带来的好处。

3.4 小结

软件确认管理可以使软件研发“流水线”前、后环节中的操作人员增加彼此之间的信任感，而减少互相间推卸责任的理由。软件确认管理可以确保软件生命周期中的每个阶段都按部就班地开展工作，避免了外界或人为的因素而导致项目出现混乱，它是让项目“冷静”下来的重要手段。

3.5 思考题

1. 软件确认的准备工作有哪些？
2. 常用的确认方法有哪两大类？
3. 确认是如何来提高团队合作信任感的？

4

第 4 章

软件质量管理的预防手段——同行评审

提高软件产品质量的手段，如果只依靠软件测试人员（SQC）在开发流程的末端进行一次努力是绝对不够的，就算有些软件项目采用迭代式开发模型，在整个软件生命周期内尽早地将部分模块或功能进行测试，仍然是不够的。

软件测试人员常用的测试手段，例如：功能性测试、界面友好性测试、冒烟测试、回归测试等方式都属于被动型的测试，因为此时项目的工作产品已经完成，软件产品质量的好坏已经定型，不管软件测试人员是否测试，该软件的缺陷都已经驻留在产品中。此时软件测试人员所做的工作只是尽可能多地发现产品的缺陷，以防止软件的缺陷落入客户的手中。因此，这种被动型的软件测试方式会消耗大量的人力和物力，而且可能会因为覆盖率的不足导致漏测情况的出现。

那么如何提高软件测试的效率和效果，尽早发现产品的缺陷，采用更加主动的方式来提高软件的质量，本章就将为大家介绍软件质量管理中的一个最佳实践——“同行评审”。

4.1 软件同行评审的概述

同行评审的英文是 Peer Review。Review 的意思是检查、审阅。Peer 在英文中有两个含义，一个是“同行、同辈”的意思，同行一般泛指从事同样工作的人，例如同样从事软

件开发的同事，或者同样从事软件测试的同行；Peer 的另外一个意思是“凝视、盯着看”，也就是说 Review 的工作要盯着每个环节，要认真真地去做。从字面意思可见，同行评审是一群从事相同或相关工作的人在一起认真真地对工作产品进行检查或审阅。

其实同行评审对广大软件质量管理人员，特别是软件测试人员来说并不陌生。软件测试可以粗略分为两大类：动态测试和静态测试。动态测试指的是通过运行代码或程序为基础进行的测试，其目的往往在于检查。静态测试就正好相反，它是静态执行程序代码而寻找程序中可能存在的错误或评估程序的过程，所以同行评审其实也就是静态测试，其目的在于预防。

为什么同行评审是静态测试呢？例如：在开发过程中的代码走查；或者在编码结束后，通过反向工程将代码转化为 UML 的静态类图，然后将其与设计文档进行对比。这些在大家日常工作中的行为都是同行评审，同样也是静态测试，因为这些都不需要运行程序，而且也能发现产品中各种各样的缺陷，从而提高产品的质量，更重要的一点就是该项工作是项目组相关人员共同合作完成，这也就体现了同行的含义。

另外，相对于动态测试而言，静态测试成本更低，效率更高，它可以在软件开发生命周期的早期就发现缺陷和问题，从而减少返工的成本。

同行评审在 CMMI 中是 VER（VERIFICATION）验证的一个 SG（特殊目标）。在 CMMI 中对同行的定义就不仅仅局限于从事相同工作的人，而是与该工作相关的所有人员，例如：软件开发人员的工作就与软件设计人员、软件测试人员、软件需求人员、项目管理人员的工作息息相关，凡是从事软件相关工作的人，都可以称为同行。

为什么 CMMI 将同行的范围扩大了？这样做是有意的，还是对于英文理解上的偏差？上面已经讲述同行评审其实就是静态测试，也就是说同行评审不是一种单一的测试方法，而是一类软件测试方法的统称。另外软件行业也有其自身的特点，部分岗位的技能往往会覆盖另一种岗位，例如对代码进行走查时，参与走查的人员不一定是软件开发人员，软件设计人员的参与往往会更有效果，效率也会更高。对需求文档进行审查时，如果都是需求人员参与评审，效果不一定理想，因为需求文档的最终使用者不是需求人员。由此可见，CMMI 将同行的范围扩大是针对软件行业的自身特点进行定义的，是符合大家日常工作需要的。

另外，熟悉 CMMI 的朋友都会发现 CMMI 中有 VER 验证和 VAL 确认两个 PA，初学者都会有疑问，为什么同行评审（Peer Review）是属于 VER 而不是 VAL 呢？

VER 验证主要介绍的是具体软件测试的方法和流程，正如上面所述，同行评审是静态测试的统称，而不是一种具体的测试技术。另外 VER 和 VAL 往往是密不可分的，不管是哪一种形式的同行评审，其结果都是需要得到所有参与评审人员对其的共同承诺，这就是 VAL 确认的深层含义。

4.2 软件同行评审流程及最佳实践

按照被评审的对象进行划分,可以分为对代码的走查和对各种工作产品(Work Product)的评审。这里工作产品的意思是指在软件开发生命周期中所产生的各种对象,包括各种文档、组件等。

代码走查依据的是每个公司颁布的编码规范等技术标准,可以通过事先制订好的检查表(Check List)进行检查。

评审从形式上可以分为正式评审和非正式评审。非正式评审更加简单,其过程可以针对正式评审的流程进行裁剪。

4.2.1 同行评审计划阶段

在标准的软件开发流程中,项目启动时项目经理就需要根据本项目的规模、周期、项目团队成员的技能等因素,确定本项目有哪些工作产品需要被同行评审,还要指明要在项目的哪个阶段进行评审。项目经理可以将这些信息记录在项目进度计划的 Gantt 图中,以便安排每次评审的具体日期和时间。换句话说,软件的同行评审是被计划的。在软件开发过程中,常见的同行评审如表 4-1 所示。

表 4-1 软件开发生命周期中常见的同行评审

软件生命周期	待评审的工作产品	评审方式
项目立项阶段	项目立项书 项目过程定义书	评审
需求调研阶段	软件需求说明书 系统规格说明书	评审
项目计划阶段	项目计划 项目进度计划 项目质量管理计划 项目度量管理计划 项目配置管理计划 项目测试计划 项目估算表 项目采购计划 项目风险管理计划	评审

续表

软件生命周期	待评审的工作产品	评审方式
项目设计阶段	概要设计文档 详细设计文档 产品集成计划	评审
编码开发阶段	代码	走查
	集成测试用例 单元测试用例	评审
系统测试阶段	系统测试用例 验收测试用例	评审
验收交接阶段	产品验收结果 项目结项报告	评审

当项目经理选定本项目所需要开展哪些同行评审后，就需要为这些评审进行必要的准备。首先要根据项目组的情况安排每次评审的参与人员，但要尽量做到参与评审的人员在工作角色或掌握技能等方面进行互补。表 4-2 给出了同行评审中核心人员的参与列表。

表 4-2 同行评审的参与者

软件生命周期	待评审的工作产品	同行评审参与者
项目立项阶段	项目立项书	高层经理 项目经理 软件质量保证人员（SQA）
	项目过程定义书	项目经理 EPG 经理 软件质量保证人员
需求调研阶段	软件需求说明书	客户 需求人员 项目经理 软件设计人员 软件开发人员 软件测试人员（SQC） 软件质量保证人员
	系统规格说明书	客户 需求人员 项目经理 软件设计人员

续表

软件生命周期	待评审的工作产品	同行评审参与者
需求调研阶段	系统规格说明书	软件开发人员 软件测试人员 软件质量保证人员
项目计划阶段	项目计划	客户 高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
	项目进度计划	客户 高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
	项目质量管理计划	高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
	项目度量管理计划	高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员

续表

软件生命周期	待评审的工作产品	同行评审参与者
项目计划阶段	项目配置管理计划	高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
	项目测试计划	高层经理 项目经理 需求人员 软件测试人员 软件质量保证人员
	项目估算表	高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
	项目采购计划	高层经理 项目经理 需求人员
	项目风险管理计划	高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
项目设计阶段	概要设计文档	项目经理 软件设计人员 软件开发人员 软件质量保证人员

续表

软件生命周期	待评审的工作产品	同行评审参与者
项目设计阶段	详细设计文档	项目经理 软件设计人员 软件开发人员 软件质量保证人员
	产品集成计划	项目经理 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员
编码开发阶段	代码	软件设计人员 软件开发人员 软件质量保证人员
	集成测试用例	软件设计人员 软件开发人员 软件质量保证人员
	单元测试用例	软件设计人员 软件开发人员 软件质量保证人员
系统测试阶段	系统测试用例	项目经理 需求人员 软件测试人员 软件质量保证人员
	验收测试用例	客户 项目经理 需求人员 软件测试人员 软件质量保证人员
验收交接阶段	产品验收结果	客户 高级经理 项目经理 需求人员 软件测试人员 软件质量保证人员

续表

软件生命周期	待评审的工作产品	同行评审参与者
验收交接阶段	项目结项报告	高层经理 项目经理 需求人员 软件设计人员 软件开发人员 软件测试人员 软件质量保证人员

由此可见，同行评审在软件开发过程中是经常出现的，要想把同行评审做好，首先必须有计划。但很多项目管理人员在制订项目计划时，往往只关注软件产品本身，而忽略了例如同行评审、周会等小的、周期性的事件和任务，这是 WBS 分解不完整的典型错误，项目经理应该意识到如果这些小的任务累加起来，所花费的工作量也是十分惊人的。

4.2.2 同行评审启动阶段

1. 制订准入与准出条款

同行评审计划完毕后，就要针对每个待评审的工作产品制订评审的准入和准出条款。为了避免在评审会上才发现待评审的工作产品是不符合要求的，需要制订相关的检查条件，由作者或第三方对它进行事先检查。例如：需求文档中是否存在遗漏的功能、需求文档的格式是否符合要求、需求文档中的 UseCase 与文字描述是否存在二义性等。一般情况下会推荐项目经理来检查待评审的工作产品，因为大多数的评审会是由项目经理主持的。

同样要制订同行评审的准出条款，也就是说什么样的工作产品是可以得到与会人员的认可并通过本次同行评审，或有条件地通过本次评审。例如：待评审的工作产品不能存在严重级别为 1~3 的缺陷、待评审的产品必须覆盖所有的业务功能；最终选择的方案必须高于 8 分等。只有提前定义了准出的条款，在评审会上才能尽快确定最终的评审结果。

2. 确定评审的方法

在定义完评审的准入和准出条款后，项目经理就需要为每次评审选择适当的评审方法。同行评审多是以讨论的形式出现，在讨论中仁者见仁智者见智，为了避免无休止的争论，必须事前明确本次评审的方法，简单来讲就是在游戏开始前要先定义一个输赢的标准，总不能出现争论时依靠抓阄来产生结果。

常见的方法有一票否决法、投票优胜法、加权打分法、筛选淘汰法。

一票否决法、投票优胜法常在对某个事物进行决策的评审中使用。加权打分法、筛选

淘汰法常在对某些技术方案进行选择时使用。在日常评审或决策中，加权打分法、筛选淘汰法往往是一起使用的。在评审的准则中有些条款是必须遵循的，有些条款是可选的，因此在评判结果时要先使用筛选淘汰法进行过滤，剔除不符合条件的方案，缩小评审或决策的范围，最后才对入围的方案进行加权打分。

3. 对规则达成一致

同行评审是一项团队活动，评审的准入和准出条款以及评审的方法就是团队活动的游戏规则，如果参加活动的人员对本次游戏的规则不了解，或者不认可某些规则，那么在游戏的过程中就必定会发生不和谐的状况。为了避免此类事情的发生，以上制订的游戏规则必须由与会人员先进行讨论，并得到大家的共同认可。

4. 发出评审通知

以上评审的各项规则确立后，同行评审的主持人就将按照评审的进度计划，发通知给所有与会人员，让其做好评审的准备。如图4-1所示，评审的通知往往包含以下内容：

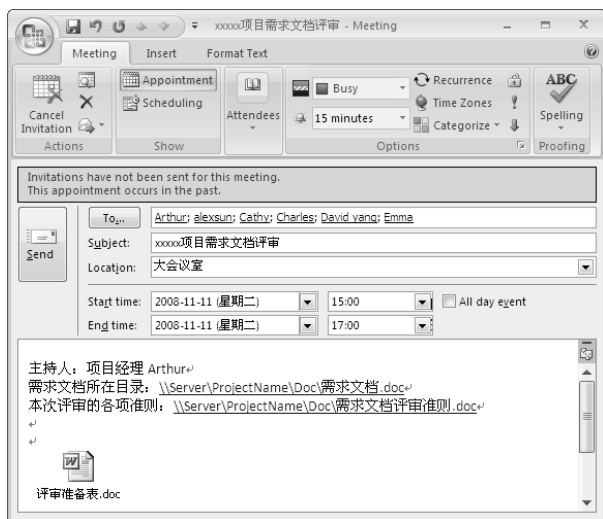


图 4-1 需求文档评审通知

- ① 本次评审的标题
- ② 同行评审的时间
- ③ 同行评审的地点
- ④ 同行评审的主持人
- ⑤ 服务器上待评审的工作产品所在目录

- ⑥ 同行评审的各项规则
- ⑦ 与会人员个人的评审准备表

5. 完成个人准备

与会人员必须事先熟读文档。在公司范围内要逐渐营造一种氛围，评审工作也是和开发、设计、测试同等重要的。同行评审是一个集体活动，谁忽略了同行评审就等于浪费自己和大家的时间，是对集体的不尊重。

另外，评审是被计划的，因此项目会预留时间给大家做好准备，个人在准备评审时一定要多思考。如表 4-3 所示，与会人员要将自己有疑问的地方或者是好的建议填写到个人的评审准备表中，并将该表回复给会议的主持人和待评审工作产品的作者。

表 4-3 评审准备表模板

项目名称						
计划评审日期						
角 色	项目经理		作者		主持人	
	与会人员					
	QA 人员					
	评审准备所花费时间 (小时)					
<input type="checkbox"/> 已阅读 <input type="checkbox"/> 未阅读						
序 号	问题描述				备 注	
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

如果某些与会人员的准备表中注明“未阅读”，除非十分必要，否则该与会人员不能参加本次同行评审。

如果某些与会人员的准备表中问题描述较少，甚至没有，或者提出的问题十分表面，

那么主持人就需要特别注意，通常是因为该与会人员没有认真阅读文档或进行思考。

6. 同行评审准备流程图

至此，同行评审的准备工作就完成了。将以上步骤贯串起来，其流程图如图 4-2 所示。虽然步骤较多，但要做一次有效的、正式的同行评审，这些都是必要的。

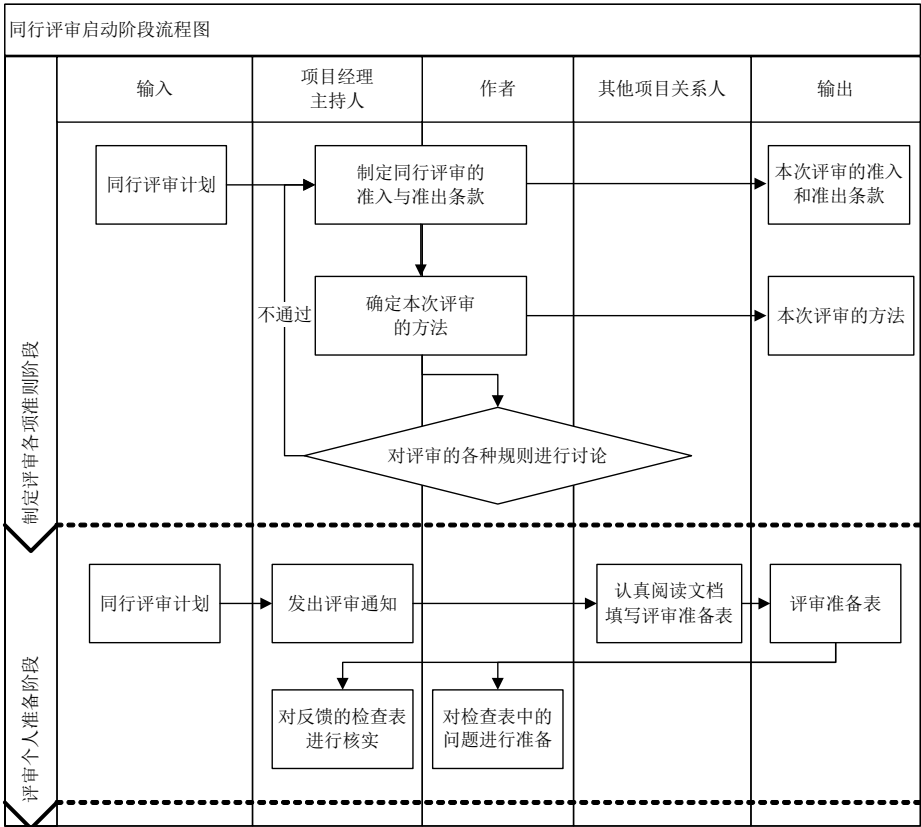


图 4-2 同行评审启动阶段流程图

4.2.3 同行评审执行阶段

在执行同行评审时，与会人员要清楚地知道评审的对象是工作产品，而不是该工作产品的作者。很多时候同行评审都像批斗会那样给作者带来很大的压力，因此需要在公司范围内营造一种良好的气氛。

评审会的主持人要注意控制会议的内容，同行评审会的焦点是所有与会人员提交上来

评审准备表中的内容。同行评审会是针对问题进行讨论的，而不是业务或技术的培训会，更不是头脑风暴会，这是提高同行评审效率的关键。

在评审过程中，作者只需要针对每人提交上来的准备表中的问题逐一进行解答，并且将所发现的问题或缺陷进行记录即可。

由此可见，一个准备充分的同行评审，其执行的过程是非常短暂的，但效率是非常高的。同行评审执行的流程如图 4-3 所示。

4.2.4 同行评审收尾阶段

同行评审产生的最终结果一般分为三种情况：通过、有条件通过和不通过。

依据之前定义的评审准出条款和评审的方法，针对评审记录中各个问题的答复进行判断。如果不能满足本次评审的准出条款，那么本次评审的结果就是不通过。

如果绝大部分符合本次评审的准出条款，但又存在一些不是很重要的缺陷，那么经过与会人员的讨论和打分，如果大家认可这些缺陷是可以被修正的，则本次评审可以有条件通过。但是在这种情况下，主持人需要指派相应人员对发现的缺陷进行跟进，确保作者在会后对遗留的缺陷进行修改，而且修改的结果符合要求。

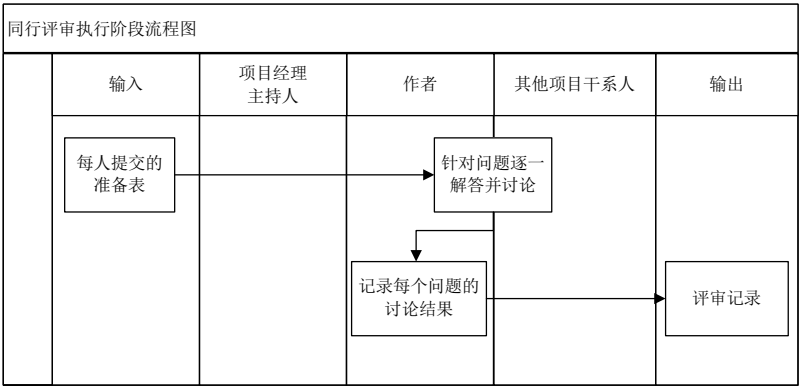


图 4-3 同行评审执行阶段流程图

同行评审的主持人在会后要将本次评审的度量数据进行收集，其中包括准备阶段和评审执行阶段的各项度量数据，并将这些数据提交给相关度量管理人员。其流程如图 4-4 所示。

4.2.5 同行评审流程裁剪指南

以上介绍了正式的同行评审流程，但对于很多小型的项目或时间周期很短的项目来说，这种正式的评审流程有些过于庞大，在项目中无法真正贯彻执行，这就需要管理人员对正式的评审流程进行裁剪，制订符合本项目特点的个性化流程。

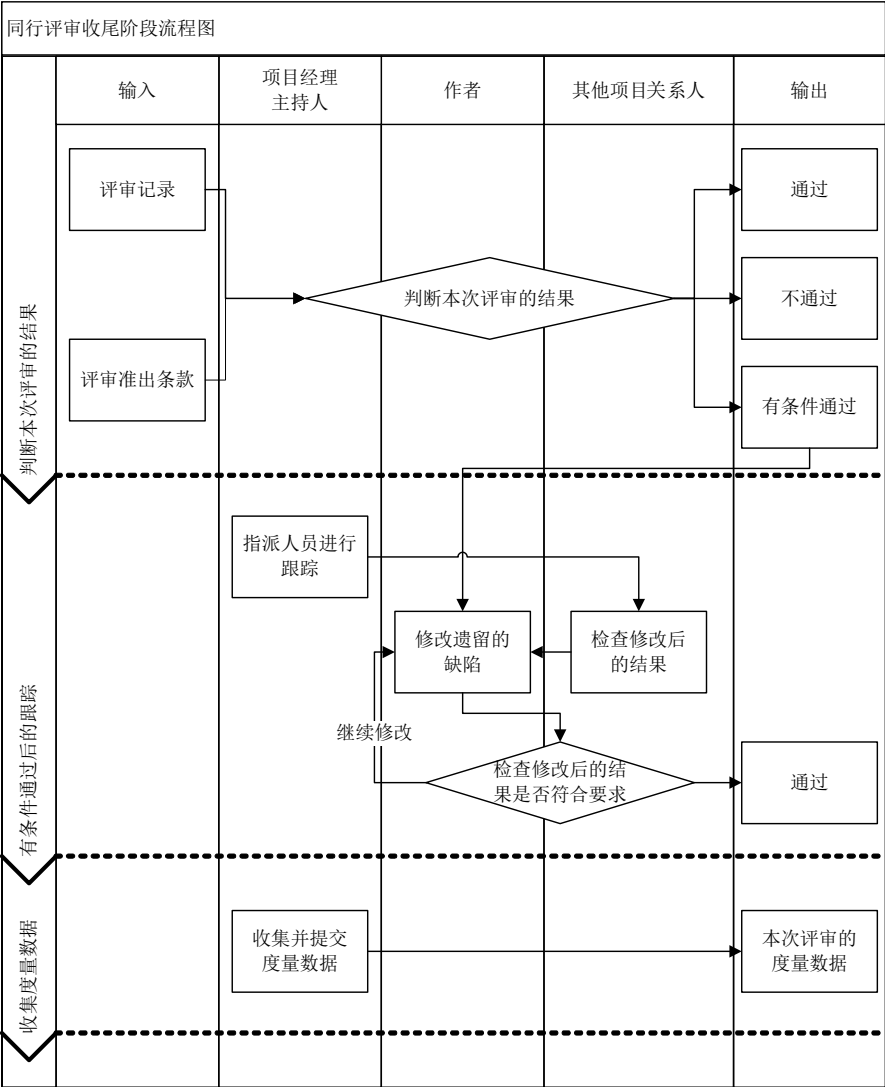


图 4-4 同行评审收尾阶段流程图

在同行评审计划阶段，根据项目的类型、规模等因素确定所需评审的工作产品。

在同行评审启动阶段，主要的工作有以下几点：

- ① 准入/准出条款、评审的方法通常在公司范围内都有统一的标准，每个项目组一般不需要再自行定义。如果公司范围内已经有相关标准，那么这几个步骤可以进行裁剪。
- ② 对于准入/准出条款和评审方法的讨论，如果项目较小，团队成员之间的沟通又比

较顺畅，那么这个环境可以被简化或被裁剪。

③ 评审的正式通知在一些小规模团队中也可以被简化或被裁剪，前提是与会人员都应该在同一个地点办公，但不管是那种通知方式，都必须做到提前通知。

同行评审执行阶段和收尾阶段已经非常简单，因此不建议对其流程进行裁剪。

对于代码走查类型的同行评审来说，工作的流程可以进行一下定义和裁剪。

代码走查应该被软件开发人员视为与编码、单元测试同等重要的日常工作，只是代码走查的频率不像单元测试那样天天都要进行。在制订项目计划时可以根据实际情况定义代码走查的间隔时间，例如三天检查一次，或一周检查一次。

在代码走查的启动阶段，不一定要等某个模块的功能完全实现再进行代码走查，因为代码走查的重点不是模块的功能是否正确，而是代码是否规范、算法是否合理，因此代码走查的准入条款可以被裁剪。代码走查的准出条款相对比较严格，通常发现的所有缺陷都必须被修正。代码走查的方法可以是项目组内开发人员互相检查，也可以安排设计人员来检查开发人员的代码，只要设计人员的时间允许，这种检查方式的效率和效果都是最好的。由于代码走查中准入/准出条款及走查的方法都比较简单，因此不需要对这些内容进行再次讨论，另外代码走查已经成为开发环节中的日常工作，而且流程相对简单，因此也不需要每次走查前都发出正式的通知。代码走查也不需要填写评审准备表，公司的编码规范就已经起到评审准备表的作用。

在代码走查的执行阶段中，只需要把发现的缺陷以及不符合编码规范的地方记录下来即可。代码走查的结果只有通过和不通过两种，软件项目不存在代码级别上有条件的通过，这会给软件产品带来较大的风险。

裁剪指南的原则就是不管增加还是减少了某些流程或工作产品，都不能给项目带来风险。对于项目组裁剪后的流程都必须经过过程改进（EPG）人员的确认。

4.2.6 同行评审最佳实践

要想进一步提高同行评审的效果和效率，单纯按照以上步骤执行还不够。接下来为大家讲解一些同行评审的技巧。

① 为了对同行评审的效果和效率进行客观的评价，主持人依据同行评审度量指标对同行评审的数据进行收集，并将这些度量的基础数据提交给专人进行统计和分析。这个步骤是同行评审必不可少的环节，也是今后提高同行评审效果和效率的基础和依据。

② 同行评审不是批斗会。与会人员准备表中的反馈一定要提前给作者，让作者针对这些问题做好准备，这样作者才会对本次评审有十足的把握和信心。这样做的好处是彻底避免批斗现象的发生，而且可以提高评审的效率。

③ 在组织范围内应该建立一个缺陷管理平台。保证评审环节中发现的缺陷可以被跟踪，直到缺陷被彻底解决。

④ 为了使同行评审更加有效，应该让不同角色的相关人员都参与进来。参与评审的人员不是越多越好，而是越有经验越好。要挑选有丰富工作经验的人，而不是有充足时间的人。

⑤ 同行评审的工作量很大，特别是准备阶段的工作量占了很大的比例。而准备阶段的工作量往往会被项目管理人员忽略，这就造成与会人员没有时间熟悉待评审的工作产品，这也是同行评审被形式化的主要原因。因此，我们才强调同行评审是要被计划的。

⑥ 针对不同的工作产品，准入/准出条款在公司范围内可以预先定义一些通用的内容，这样可以降低评审准备阶段的工作量。

⑦ 同行评审要想做出效果，必须先争取高层的支持和理解。

4.3 软件同行评审常见问题及案例分析

虽然同行评审的流程已经定义，但如何提高评审的效率和效果，让同行评审更加有效地发挥其软件质量管理预防的功效，可以通过以下案例进行分析。

4.3.1 案例 1——如何提高同行评审的效果

【案例】

在某软件公司的 OA 项目组内，项目经理小张在项目启动前与 EPG 组的同事共同定义了本项目同行评审的全部流程，而且该流程通过了评审。但在项目结束后，QA 人员经过质量回溯，发现有些关键的产品缺陷其实应该在评审会上就被发现，也就是说评审的效果有待提高。项目经理小张感到非常疑惑，在他的项目中同行评审的流程都有被很好地执行，为什么还没有起到应有的预防作用呢？

【分析】

首先要防止落入检查表的误区。同行评审的执行阶段其实就是解决评审准备表中提出的各种问题，评审准备表也就相当于检查表。

当人们使用检查表的时候，往往注意力就只集中于该表上所罗列的检查项，而这些检查项也往往集中于工作产品本身，而忽略了其他各种边界的情况。例如：升级某个软件系统，就要考虑新系统与原有系统之间的接口是否匹配；如果是新开发的项目，也要考虑客户原有数据如何导入的问题。总之可以将软件测试中边界值的理论从微观的函数或方法的级别扩展到更大的范畴进行使用。

其次，工作产品的漏测率与抽样率成反比。如何提高同行评审的抽样率呢？大家要知

道同行评审中的抽样就是同行评审准备表中记录的每个检查项。只要检查项覆盖面足够，那么同行评审自然就会有效果。如果评审主持人收集回来的评审准备表中反馈的内容太少，那就没有必要继续进行评审，因为这样的评审只能是走过场。

评审的主持人和作者可以先将工作产品的内容进行分解作为纵坐标，其分解方法与分解 WBS 相同，然后将本次评审的关注点逐一罗列作为横坐标，最后将与会人员的反馈填写到这个矩阵中，以此来判断检查项的覆盖率是否足够。检查项覆盖表如表 4-4 所示。

表 4-4 同行评审检查项覆盖表

待评审的功能点	功能完整性	技术复杂度	可维护性
功能点 A1	×			×
功能点 A2	×	×		
功能点 A3	×		×	
功能点 A4	×			×
功能点 B1		×		×
功能点 B2	×			
功能点 B3	×		×	×
.....				

通过该表的填写，接下来就要统计该表中有多少检查项被覆盖，有多少检查项被遗留。

同行评审覆盖率 = $\frac{\text{被覆盖的检查项CD}}{\text{总检查项CI}} \times 100\%$

通过对以上方法进行分析，如图 4-5 所示，项目经理小张发现项目中的同行评审覆盖率都较低，与会人员评审准备表中问题的内容都相对重复，也就是大家都集中在某些内容上进行思考，而忽略了一些其他环节。最终出现的质量问题恰好是同行评审中没有涉及的地方，软件质量预防工作也因此出现了漏洞。

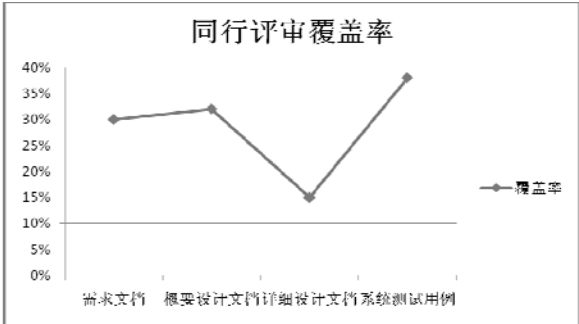


图 4-5 某公司 OA 项目同行评审覆盖率

4.3.2 案例2——如何计算同行评审的投资回报率

【案例】

某软件公司近期完成了一个 CRM 的项目，高级经理老王在项目结束后对项目经理小李说：“你们的项目经常开评审会，好像挺浪费大家的时间，应该将大家的时间更好地安排到软件设计、开发和测试中，同行评审好像对项目没有太多的必要。”项目经理小李将同行评审在软件质量中的预防作用告诉了老王，但是老王还不太相信，小李该怎么办呢？

【分析】

衡量一个工作是否应该去做，可以用投资回报率 ROI 来判断。在同行评审中就是所有与会人员花费在本次评审中的成本与本次评审发现缺陷具有的价值之间的比例。也就是每投入 1 块钱在同行评审上可以得到多少钱的回报。公式如下：

每个与会人员在本次评审所花费的时间 UT = 本次评审准备工作的时间 + 开会的时间 + 开会后某些人员还需要对缺陷进行跟进的时间

每个与会人员在本次评审所花费的成本 UA = 该人员的单位小时成本 $\times UT$

所有与会人员在本次评审总的花费 PC （元）= $UA_1 + UA_2 + UA_3 + \dots$

公司可以通过一段时间的积累或讨论，按照不同工作产品、不同的严重程度来统计如果某个缺陷没有被及时发现，那么其返工的成本为多少。例如：在对需求文档和设计文档进行评审时，发现同等级别严重程度的缺陷，其返工的成本是不同的，需求评审所发现的缺陷价值更高。

总返工成本 RC （元）= 缺陷 1 的成本 D_1 + 缺陷 2 的成本 D_2 + 缺陷 3 的成本 $D_3 + \dots$

$$\text{同行评审 ROI} = \frac{\text{总返工成本 } RC}{\text{所有与会人员花费在本次评审总的成本 } PC}$$

项目经理小李对本项目同行评审 ROI 进行统计分析，其结果如图 4-6 所示。其中详细设计的投资回报率是负数，经过更加深入的分析，本项目选用了两名高水平的设计人员，该设计文档在评审中没有发现任何缺陷。

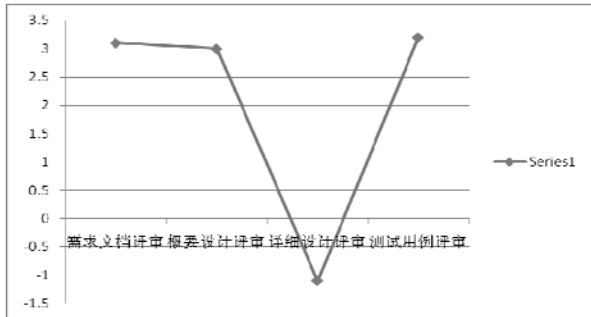


图 4-6 某公司 CRM 项目同行评审 ROI 统计图

项目经理小李将该图表交给老王，并通过客观的度量数据反映了同行评审在 CRM 项目中起到了积极的作用，最后得到老王在今后项目中的支持和理解。

4.3.3 案例 3——如何更好地执行同行评审

【案例】

某公司已经通过 CMMI L3 级的认证，公司要求所有项目中关键的工作产品要经过同行评审。项目经理小杨是该公司 GIS 项目的负责人，该项目工期较紧。在项目进行中小杨发现大家对同行评审都比较抵触，而且多数项目成员都觉得同行评审只是走过场。

【分析】

项目经理小杨请来公司的 EPG Leader 老张来和他一起对这个现象进行分析，他们通过与项目团队成员的交流发现，项目计划中只计划了开同行评审会的具体时间，而没有给与会人员预留阅读文档、填写准备表的时间。另外项目奖中有一条评判的标准是“个人缺陷数大于 30 个且小于 50 个的，项目奖将减少 10%；个人缺陷数大于 50 个的，项目奖将减少 30%”。

经过以上分析，找到了大家抵触同行评审的最终原因。项目经理小杨与公司领导和客户进行了沟通，大家一致认为保证产品质量具有最高优先级，因此小杨修改了项目计划，为之后的每次评审预留了准备时间。

通过项目经理小杨的反馈，公司领导也认识到了绩效考核与软件度量之间的关系，及时修改了那些影响团队积极性的绩效指标，这给提高产品质量创造了良好的环境。

4.4 小结

同行评审的方法有很多，但其共同的目的是为了预防缺陷的发生。依靠软件测试人员进行大量的、重复性的测试永远无法在真正意义上提高产品的品质，只有采用积极的预防措施，才能真正提高软件产品的质量。

要想将同行评审真正贯彻执行，同行评审的启动阶段是关键，只有将准备工作做到位，才能提高同行评审的效率和效果。

4.5 思考题

1. 同行评审分为几个阶段？
2. 同行评审的结果有几种？

5

第 5 章

软件质量管理的审计体系——质量保证

软件质量保证（Software Quality Assurance）简称 SQA，它的由来是为了弥补软件质量控制（QC），也就是业内称为软件测试工作的不足。因为很多软件公司的项目都会因为需求的大量变化或其他原因造成返工，浪费了大量的项目时间，从而导致软件测试特别是系统测试的时间严重不足。软件测试人员经常会听到项目经理说：“你们只有 1 天的测试时间，因为明天系统就要发布给客户了”，这样的软件产品漏测率高，给软件产品的质量也带来严重的隐患。

软件质量保证是软件质量体系中非常重要而又特殊的组成部分。之所以重要是因为软件质量保证的工作涉及软件研发流程的各个环节，以及每名参与研发的人员；之所以称其特殊是因为质量保证的工作不涉及具体的软件研发细节，并且软件质量保证人员的职位在公司组织架构中又独立于研发体系之外，它辅助高层管理人员对项目进行审计，它的工作成果会给整个研发团队提供信心。软件质量保证的工作在软件质量体系中起到了以下两个作用：

① 第三方审计。例如某软件项目中发生了系统测试时间被之前工作的返工所大量占用，这样必然会导致软件测试人员的漏测率升高。当客户在使用该软件产品时发现了质量问题并向公司进行投诉，那么公司领导第一个批评的就是软件测试人员。这样的一个结果对软件测试人员是非常不公平的，因为原本 1 周的系统测试时间到头来变成了 1 天的系统测试时间，这是一个不可能完成的任务。如果有了 SQA 的参与，那么 SQA 人员会审计软

件项目的方方面面，这样就可以客观发现问题的根源，透过现象看本质，避免管理层出现错误的判断。

② 预防的作用。这个思想符合了当今先进的软件质量理论“质量行为前行，越早发现缺陷就可以挽回更多的损失”。

软件质量保证的叫法其实并不够准确，更加准确的叫法应该为 PPQA，其英文全称是 Process and Product Quality Assurance，这是从软件质量保证的工作内容进行定义的。因为软件质量保证人员的具体工作就是要确保项目团队的工作内容符合公司既定的研发流程，并且要确保项目的产品符合质量的要求，简单来讲软件质量保证人员所审计的内容就是两个方面：过程和产品质量。

软件质量保证的工作是一个复杂的系统工程，之所以将它上升到系统过程的层面，是因为软件质量的保证不是某一个人、某一个团队或某一个项目的工作，而是整个企业都要参与的。虽然软件工程中的各个流程都是围绕软件开发流程进行工作的，但软件质量保证的工作不只是对软件开发和测试的流程进行审计，而是对整个软件工程中的各个组成部分都要进行审计，例如配置管理的流程、软件度量的流程、风险管理的流程等，这样做的目的是为了软件研发的过程更为透明、更为可控，这样才会让项目管理人员，特别是高层管理人员或客户感觉到安全感，也只有这样他们才会相信项目能够提供一个高质量的产品。软件质量保证要起到第三方的审计作用就必须独立于项目组之外，就像国家在进行行政改革时采用“政企分离”的目的一样。在当今软件企业中对软件测试人员的重视度越来越高，但是对软件质量保证的关注度还只是刚刚开始，很多软件质量保证人员都没有技术、项目管理、质量管理的背景，工作经验也都不足，在这种情况下虽然可以确保审计工作的客观性，但是在审计过程中对发现的各种不一致项很难让人信服，那么公司的高层领导也就会对他缺乏信任和支持，软件质量保证人员在公司的地位也就非常尴尬，工作起来的难度也相当大。因此，一名优秀的软件质量保证人员应该具有多年软件开发、软件项目管理、质量管理工作经验，并且精通公司行业背景的知识，最好还能掌握数据统计和概率分析。

5.1 软件质量保证概述

如果一个软件项目所开发出来的软件产品具有较高的可用性和易用性，这是否就满足了客户的需要，也达到了产品的质量呢？假如一个软件产品有较高的产品质量，但是客户要求半年完成，项目组却用了一年半才交付，那客户就完全满意吗？假如一个软件项目的预算成本是 10 万元人民币，但是项目组却花费了 20 万元的成本开发出非常高质量的产品，那么客户和公司就真的完全满意吗？

在软件开发过程中质量被视为软件产品的生命，而始终被所有人员高度关注，然而在现实生活中总是不尽如人意。归根到底是没有把握软件产品质量的内涵，很多时候项目组仅停留在减少软件运行错误、加强软件测试、尽量避免软件缺陷落入客户手中的一般性做法，而缺乏对整个软件开发生命周期的全过程质量管理。因此软件质量保证的目标如下：

- ① 通过对过程的审计确保交付给客户的产品具有高质量。
- ② 保证软件项目的过程恰当、合理，并遵循公司的标准和规范。
- ③ 对软件项目组各个方面的评价客观准确。
- ④ 对发现的不符合项及时与当事人进行沟通，在必要时将信息客观反馈给高层领导并争取高层的支持。
- ⑤ 让软件研发的过程更加透明，以便项目组成员、各级管理人员，甚至客户都可以清楚了解项目的情况和问题，这样可以获得更多的安全感和信心。
- ⑥ 为持续的软件过程改进提供必要的建议。

如图 5-1 所示，软件质量保证的工作是一个闭合的环形，包括以下内容：

- ① 所有执行的过程、产品和服务依据适当的过程标准和流程进行客观评估。
- ② 识别并记录不符合项。
- ③ 确保不符合项得以解决。
- ④ 向项目相关关系人反馈软件质量保证获得的结果。

为了确保软件质量保证人员可以顺利开展第三方审计的工作，更加客观地对软件研发体系进行评价，通常软件质量保证人员要求独立于项目组，并且直接接受高层领导的管理。这是开展软件质量保证工作的基础，其组织架构如图 5-2 所示。

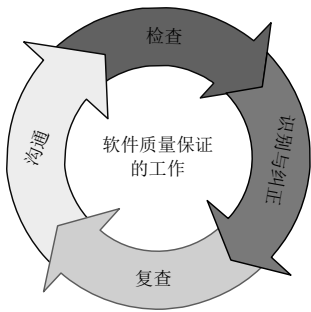


图 5-1 软件质量保证的工作

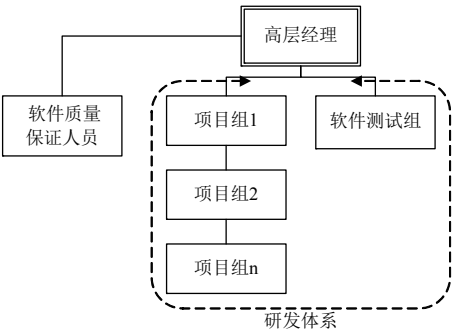


图 5-2 软件质量保证人员在组织架构中的位置

5.1.1 PPQA 与 SQC 的区别

PPQA 与软件质量控制 SQC 共同构成了一个完整的软件质量体系，二者在工作中有交

叉，但更重要的是它们的工作可以进行互补。软件质量控制是通过各种软件测试的手段，例如：系统测试、回归测试、压力测试等，确保软件产品和服务具有高的质量，那么 SQC 与 PPQA 之间的异同点是什么呢？

举一个简单的例子：当你租了一辆汽车并带着你的女朋友驰骋在高速公路上时，汽车中的各项安全措施，（例如：ABS、安全气囊等）就相对于软件质量控制的手段。但这些安全措施只是被动地为你提供安全保护，它不能真正避免危险的到来。当你开车前详细查看了车辆的使用手册，严格遵守了交通法规，关注高速公路上的各种标志并及时确认车距，以及担心测速雷达而不敢开快车等。这些都是车辆本身以外的，由第三方提供的安全保证，特别是当测速雷达发现你高速驾驶时，它会及时通知警察来对你进行检查。以上这些才是避免车祸发生的有效手段，它起着预防和监督的作用，这也就是 PPQA 的工作方式。PPQA 与 SQC 的具体区别如表 5-1 所示。

表 5-1 PPQA 与 SQC 的区别

	PPQA	SQC
角色	软件质量保证工程师	软件测试工程师
职责	过程、产品质量审计者	产品质量检查者
工作定义	为了确保软件研发过程、产品和服务符合预期的结果，依照质量保证的过程和计划采取的一系列活动	为了发现软件产品已经存在缺陷而进行各种抽样检查的一系列活动
工作重点	侧重的是管理方面	侧重的是技术方面
工作范围	软件研发全过程的管控，它包括了对 SQC 所做工作的审计和监控	软件研发过程中的一个环节
工作方式	通过管理手段进行各种检查，以综合提高产品的质量	通过抽样的方法进行测试，以避免缺陷落入客户的手中
使用工具	检查表	各种测试工具

软件质量保证与软件质量控制工作的内容虽然有着更多的互补关系，但也有重叠的内容。在软件生命周期的各个阶段中它们关注的内容也是不同的，具体内容如表 5-2 所示。

表 5-2 在软件生命周期各个阶段 PPQA 与 SQC 的工作对比

类别阶段	PPQA		SQC	
	主要的工作内容	提交的工作产品	主要的工作内容	提交的工作产品
项目启动阶段	检查项目的启动过程、项目过程的定义	项目启动过程检查表； 项目定义过程检查表； 不符合项报告	无	无
需求阶段	对需求开发、需求管理过程进行检查； 对《软件需求说明书》、《系统规	需求开发过程检查表； 需求管理过程检查表； 需求评审过程检查表；	参加《软件需求说明书》、《系统规格说明书》评审	对需求文档评审结果的承诺和签字

续表

类别阶段	PPQA		SQC	
	主要的工作内容	提交的工作产品	主要的工作内容	提交的工作产品
需求阶段	格说明书》的评审过程进行检查; 对《软件需求说明书》、《系统规格说明书》的内容进行检查; 对需求确认书的内容进行检查	需求文档内容的检查表; 需求确认记录检查表; 不符合项报告	参加《软件需求说明书》、《系统规格说明书》评审	对需求文档评审结果的承诺和签字
项目计划阶段	对项目计划的内容进行检查; 对项目进度计划的内容进行检查; 对项目各种从属计划的内容进行检查; 对项目计划的评审流程进行检查; 对项目计划的过程进行检查	项目计划过程检查表; 项目计划评审过程检查表; 项目计划及其从属计划内容检查表; 不符合项报告	参与项目计划的评审,为制订测试计划提供支持	对项目计划评审的结果承诺和签字
系统设计阶段	对概要设计文档进行检查; 对详细设计文档进行检查; 对数据库设计文档进行检查; 对系统设计过程进行检查; 对设计阶段评审进行检查	设计过程检查表; 设计评审过程检查表; 设计阶段各种文档内容检查表; 不符合项报告	无	无
编码开发阶段	对代码走查过程进行检查; 对代码走查表的内容进行检查; 对单元测试的内容进行检查; 对软件开发的过程进行检查; 对产品集成的过程进行检查; 对集成测试过程进行检查; 对集成测试的结果进行检查	软件开发过程检查表; 代码走查过程检查表; 单元测试执行情况检查表; 产品集成过程检查表; 集成测试过程检查表; 集成测试结果检查表; 不符合项报告	无	无
系统测试阶段	对系统测试流程进行检查; 对系统测试用例的评审过程进行检查; 对系统测试的内容进行检查; 对缺陷管理的流程进行检查; 对缺陷填写的内容进行检查	系统测试流程检查表; 系统测试用例评审流程检查表; 系统测试用例内容检查表; 缺陷管理流程检查表; 缺陷内容检查表; 不符合项报告	编写系统测试用例; 评审系统测试用例; 进行系统测试; 管理、跟踪产品缺陷	系统测试用例; 对系统测试用例评审结果的承诺和签字; 提交的产品缺陷
用户验收阶段	对用户验收的流程进行检查; 对用户验收测试用例评审流程进行检查; 对缺陷管理的流程进行检查; 对缺陷填写的内容进行检查; 对用户验收报告的内容进行检查	用户验收流程检查表; 用户验收测试用例评审过程检查表; 缺陷管理流程检查表; 缺陷内容检查表; 用户验收报告内容检查表; 不符合项报告	编写用户验收测试用例; 评审用户验收测试用例; 配合进行用户验收测试; 管理、跟踪产品缺陷	用户验收测试用例; 对用户验收测试用例评审结果的承诺和签字; 提交的产品缺陷

续表

类别阶段	PPQA		SQC	
	主要的工作内容	提交的工作产品	主要的工作内容	提交的工作产品
项目结项阶段	对项目结项流程进行检查； 对项目结项报告内容进行检查； 提交个人对过程改进的建议	项目结项流程检查表； 项目结项报告内容检查表； 不符合项报告； 过程改进建议书	总结、提供项目组所需要的信息和数据； 提交个人对过程改进的建议	过程改进建议书

5.1.2 软件质量保证人员的素质和责任

软件质量保证的工作涉及软件工程的各个方面，软件质量保证人员要与不同角色的人进行沟通，因此软件质量保证人员除了要具有较高的智商和情商外，还要具备以下素质和能力：

① 首先要有控制软件质量的能力。也就是说要熟练掌握公司的各种流程、标准和规范，做好第三方独立审计的工作并及时发现、纠正问题，在必要时可以利用向高层经理直接汇报的权力来“威慑”相关人员，以确保软件质量向好的方向发展。在控制软件质量发展方向的同时要学会控制自己的情绪，因为软件质量保证人员是专业的质量专家，但公司内其他人员却不一定都很了解软件质量保证的工作以及如何从根本上去提高软件的质量，软件质量保证人员在工作中往往有种“秀才遇到兵，有理讲不清”的感觉，这时就更加需要控制自己的情绪和言语，找到合适的方式进行沟通，使问题最终得以解决。

② 软件研发流程和软件产品中很多出现的问题和不符合项通常会有相似的地方，这也就需要软件质量保证人员具有对问题根源识别和归纳的能力，也就是透过现象看本质的能力。例如：某软件公司的很多项目计划都不够准确，通常项目计划在项目刚刚开始前两个月还具有指导性的意义，两个月以后就算没有其他原因，项目计划本身也需要进行变更。那么软件质量保证人员就会对多份项目计划进行检查，结果发现所有项目计划中都没有开周会的任务，这种周期性的多人参与的任务工作量其实是惊人的，这就是导致项目计划在两个月以后不准确的原因之一。对问题的归纳能力有点像软件测试技术中的“等价类比法”，要将所关注的内容进行归类，这样才能减少软件质量保证人员的工作量。

③ 软件质量保证人员要有举一反三的能力，很多问题和风险的起因都是相同或相近的，如果可以举一反三，那么对于未发生的风险和问题来说将起到预防的作用，对于已经发生的风险和问题来说就可以尽早对它进行识别，从而降低它的负面影响。

④ 软件质量保证人员要有很强的沟通能力。

⑤ 很多软件质量保证人员由于工作年限短，而且又缺乏软件开发的技术实力，在与项目经理沟通时往往处于弱势，因此软件质量保证人员应该对自己有信心，要保持适当的强势。

⑥ 要熟悉软件工程的理论和知识，这样才能更好理解公司制订的各种流程、标准和规范。也只有掌握了这些知识，软件质量保证人员才能向过程改进小组（EPG）提出过程改进的建议。

⑦ 一个软件质量保证人员通常要审计多个软件项目，因此软件质量保证人员本身要有很强的计划性和条理性，否则自己的工作都无法做好，就更谈不上审计、监督他人的工作了。

⑧ 软件质量保证人员要有一个客观的、对事不对人的职业素养。因为软件质量保证人员有权力直接向公司高层领导反映情况，这个权力是把双刃剑，它既可以帮助 QA 人员解决项目组发现的问题，也可能让人感觉有打小报告的嫌疑。因此具有一颗客观、公正的心是必不可少的。

软件质量保证人员的工作经历中最好有从事过软件开发、测试、项目管理和质量管理的经验，显然这个要求实在很高，符合条件的人确实很少，原因是软件质量保证是一个新兴的工种。软件质量保证的工作同样可以像软件开发和测试那样划分为不同的等级，不同等级所从事的工作内容和责任也是不同的。通常将软件质量保证人员分为以下三类：

① 交警

初级的软件质量保证人员可以作为软件研发过程中的交警，在很多公司中，这样角色的软件质量保证人员通常是有 1~2 年工作经验的软件开发或测试人员，甚至由熟悉软件工程的应届生来担任。

他们工作的内容和方法都很简单，只要通过一段时间的软件研发管理培训，熟悉公司的软件研发流程和规范，知道什么样的检查使用什么样的检查表即可。初级的软件质量保证人员对软件过程或产品进行检查时，只要按照检查表上的内容逐一进行核对，将发现的不符合项提交到缺陷管理系统中，并及时与相关人员进行沟通即可。当发现的不符合项不能得到解决或项目组不认可时，应该及时向高级的质量保证人员寻求帮助，或者将此情况直接反馈给公司高级领导。

总的来说，初级的软件质量保证人员就像交警查处交通违章那样，直接向所发现的不符合项贴“罚单”即可。

② 医生

中级的软件质量保证人员可以作为软件研发过程中的医生，此类软件质量保证人员最好有 3~5 年的软件开发或测试经验。因为在多年的软件研发过程中，中级的软件质量保证人员对项目中的问题已有切身的体会。

他们的工作内容和方法要更深入一些，除了发现不符合项以外，还要告诉项目经理或者相关项目关系人错在哪里，这样的错误会对项目和产品质量有哪些影响。

总的来说，中级的软件质量保证人员要像医生那样对项目进行检查和诊断，发现问题并可以开出“药方”。

③ 老师

高级的软件质量保证人员可以作为软件研发过程中的老师，此类软件质量保证人员要有多年的软件项目管理或质量管理经验，并参加过多年的软件开发或测试工作，最好高级的软件质量保证人员参加过 CMMI 的评估或拥有 PMP 等项目管理类的认证。

高级的软件质量保证人员不但要给项目开出“药方”，还要对项目相关人员进行培训，教会他们以后如何避免此类问题的再次发生。

总的来说，高级软件质量保证人员要像老师那样发现学生的弱项，并找到如何提高学生能力的方案，然后对学生进行辅导和培训。

5.1.3 软件质量保证人员与其他岗位的关系

软件质量保证人员与软件工程中其他角色之间有着种种分工与合作的关系：

① 软件质量保证与软件质量控制人员之间的关系

软件质量控制通常是指软件测试，这是软件研发流程中的一个环节，因此软件质量保证人员要对 SQC 的工作和工作过程中产生的工作产品进行审计。

软件测试人员在工作中会发现大量的产品问题，并且会产生大量有关质量方面的度量数据。因此，软件测试人员的工作结果是软件质量保证人员工作的一项重要输入数据。对这些数据的分析也是软件质量保证人员作为“医生”和“老师”的一种能力。

② 软件质量保证人员与项目经理之间的关系

当一个公司没有开展质量保证工作时项目经理有着很大的权力，项目经理做起事情来往比较随意，因为他们常以“只要给客户交付一个满意的产品”为借口来搪塞对他的任何质疑。当企业开展软件质量保证活动后，项目经理就会觉得有些不习惯，再也不能天马行空地随意而行。而且凡事必有人对其进行检查，有些项目经理会觉得软件质量保证人员就是他们的敌人，甚至有些心态不成熟的项目经理会觉得这种做法是公司高层领导对他的不信任。软件质量保证人员与项目经理之间的关系非常微妙，质量保证人员虽然有向高层领导直接汇报的权力，但是也不要以“高层领导代言人”的形象出现。反而质量保证人员要以一个朋友的姿态来辅助项目经理进行工作，因为二者之间有着共同的目的，就是让项目有一个好的质量。有了质量保证人员的辅助，对项目经理来说也可以降低一些管理上的工作量。只有这样软件质量保证人员才能与项目经理和睦相处、同心协力。

③ 软件质量保证人员与过程改进人员之间的关系

软件过程改进小组（EPG）的工作职责就是起草公司各种管理流程、标准和规范，把握一切可以改进的机会，持续不断地进行过程的改进。甚至有人把 EPG 比作改革开放进程中的主导者“国家发改委”。软件质量保证人员作为第三方的审计者往往与项目组是对立

的关系，就像拳击场上的两个运动员，当他们对某个不符合项发生争论时应该有一个裁判来进行最终的裁定。软件过程改进小组的成员就是起到裁判的作用，这样可以保证对争议解决的公正性。另外 EPG 的成员还可以帮助软件质量保证人员对项目组进行某些专题的培训，辅助软件质量保证人员做好“老师”的工作。

5.2 软件质量保证流程及最佳实践

本章将软件质量保证的内涵进行了深化，将软件质量保证定位于 PPQA，这两个“P”分别代表“过程”和“产品”，那么软件质量保证的工作流程和方法是如何实现及对“过程”进行审计，又是如何对“产品”进行审计呢？本节将通过几个软件质量保证的最佳实践来对它进行讲述。

5.2.1 对软件研发过程的审计

在当今软件质量体系里，质量的含义已经不再局限于产品，对质量的享有者也不再局限于客户。因此只有好的过程才能提供高的质量，而且可以让所有项目关系人都享受到高质量所带来的收益。传统行业中的过程常常体现在工厂中的一条条生产线，一个工序完成后另一个工序立刻开始。而软件行业中的过程不能按照软件开发的流程将项目经理、需求人员、设计人员等不同角色的人排成一队来安排他们的座位。软件行业中的过程是公司制订的各种规范、标准和流程，这些都是无形的，是需要在每个项目团队成员脑海中记忆的，这与传统行业的生产线有着本质的区别。

对软件过程的审计是基于这样一个前提：“软件项目所遵循的流程是符合项目实际需要的，是恰当并充分的；该流程是经过公司审批的”。如果这个假设都不存在，那么这种审计就缺乏存在的基础，而且审计的结果也缺乏合理性和公正性。制订一个合理的项目流程，这是过程改进小组的责任，而不是质量保证人员的责任，质量保证人员只要将过程改进建议反馈给 EPG 即可。

软件工程是个复杂的系统工程，它是由软件开发过程、软件测试过程、配置管理过程、度量分析过程、风险管理过程等十多个大的流程所构成，每个管理过程又可以分为多个子流程。到底哪些过程是需要软件质量保证人员关注的，这需要根据项目的规模和复杂度进行判断。因此，制订软件质量保证计划是软件质量保证人员所需要做的第一件事情。

软件质量保证人员对过程进行审计时按照过程的特点可以分为对触发性过程的审计和对周期性过程的审计两种。

触发性的过程就如对需求文档的评审过程，该评审过程何时开展，需要等待项目经理发出评审通知，因此该审计活动是被评审通知触发的。对触发性过程的审计流程如图 5-3 所示。触发性的审计过程可以分为以下 3 个大的步骤：

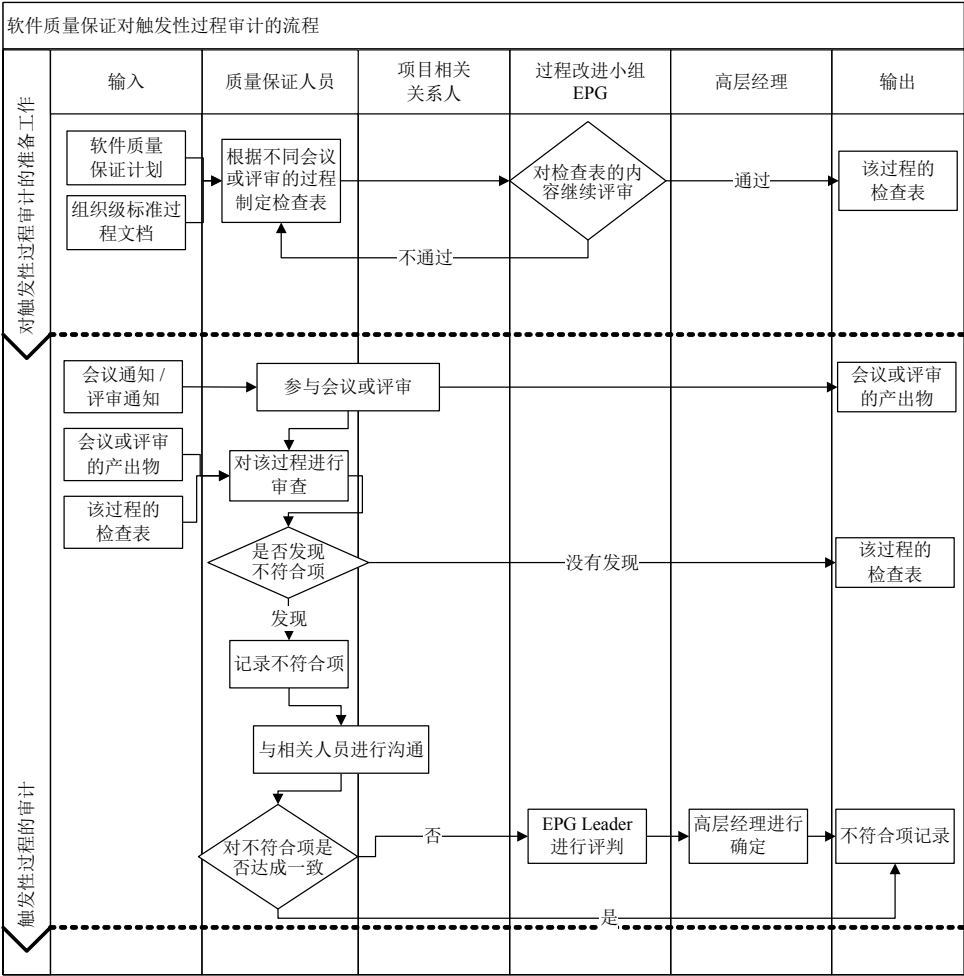


图 5-3 触发性过程的审计流程

STEP 01 QA 对触发性评审活动的准备工作。该准备工作的主要内容是针对该评审的过程制订检查表，由于过程改进小组的主要职责是制订公司内的各种规范和流程，检查表也属于其中的一个部分，因此 QA 制订的检查表需要经过 EPG 成员的评审后方能使用。例如以《软件需求说明书》评审流程为例，其检查表的内容如表 5-3 所示。

STEP 02 触发性评审的审计过程主要是通过软件质量保证人员的参与，以及对评审相关产出物进行审查。

STEP 03 尽可能与适当的人员为解决每一个不符合项进行沟通，并记录不符合项。当不符合项在项目中无法解决时，应该及时向高层领导汇报以寻求支持。度量、分析不符合项，以了解其发展趋势。跟踪不符合项的状态直到关闭为止，除非该不符合项得到高层领导的赦免。

表 5-3 《软件需求说明书》评审流程检查表

项目名称:		北极光		项目经理:	莫小贝
QA 名称:		李大嘴		审计时间:	2009-1-6
序 号	检 查 项	审计结果	本次审计是否适用	备 注	
1	评审通知是否提前 1.5 天发出	√	√		
2	《软件需求说明书》是否作为附件一起发出	√	√		
3	需求文档的评审中是否有软件测试人员的参与	√	√		
4	需求文档的评审中是否有软件开发和设计人员的参与	√	√		
5	需求文档的评审中是否有客户方代表的参与	√	√		
6	参与需求评审的人员是否都提交了本次评审的反馈表。 如果有人未提交，请在备注中指出具体人员	√	√		
7	本次评审是否有会议记录	√	√		
8	本次的与会人员是否都有签名。 如果有人未签名，请在备注中指出具体人员以及未签名的原因	×	√	开发人员白展堂因 请假未参加本次评审	
9	本次评审中的发现的问题是否都被记录	√	√		

周期性过程的审计主要是针对各种软件工程中流程的审计，例如：对需求阶段过程审计、系统测试阶段过程审计、风险管理的审计、度量管理的审计等。它针对的是软件工程中的各个流程，因此周期性审计与触发性审计最大的不同就是它审计的对象主要集中在管理类的文档和支持类的文档。对周期性过程的审计流程如图 5-4 所示。

5.2.2 对软件工作产品的审计

软件开发过程中所产生的任何工作成果都称为“工作产品”，该工作产品可以是代码、组件、文档、会议记录等。但并不是所有工作产品都要交付给客户，例如在普通项目中的概要设计文档、需求确认书等文档就是不交付给客户的。因此“产品”是特指交付给客户的部分工作产品，由此可见“产品”是“工作产品”的一个子集。

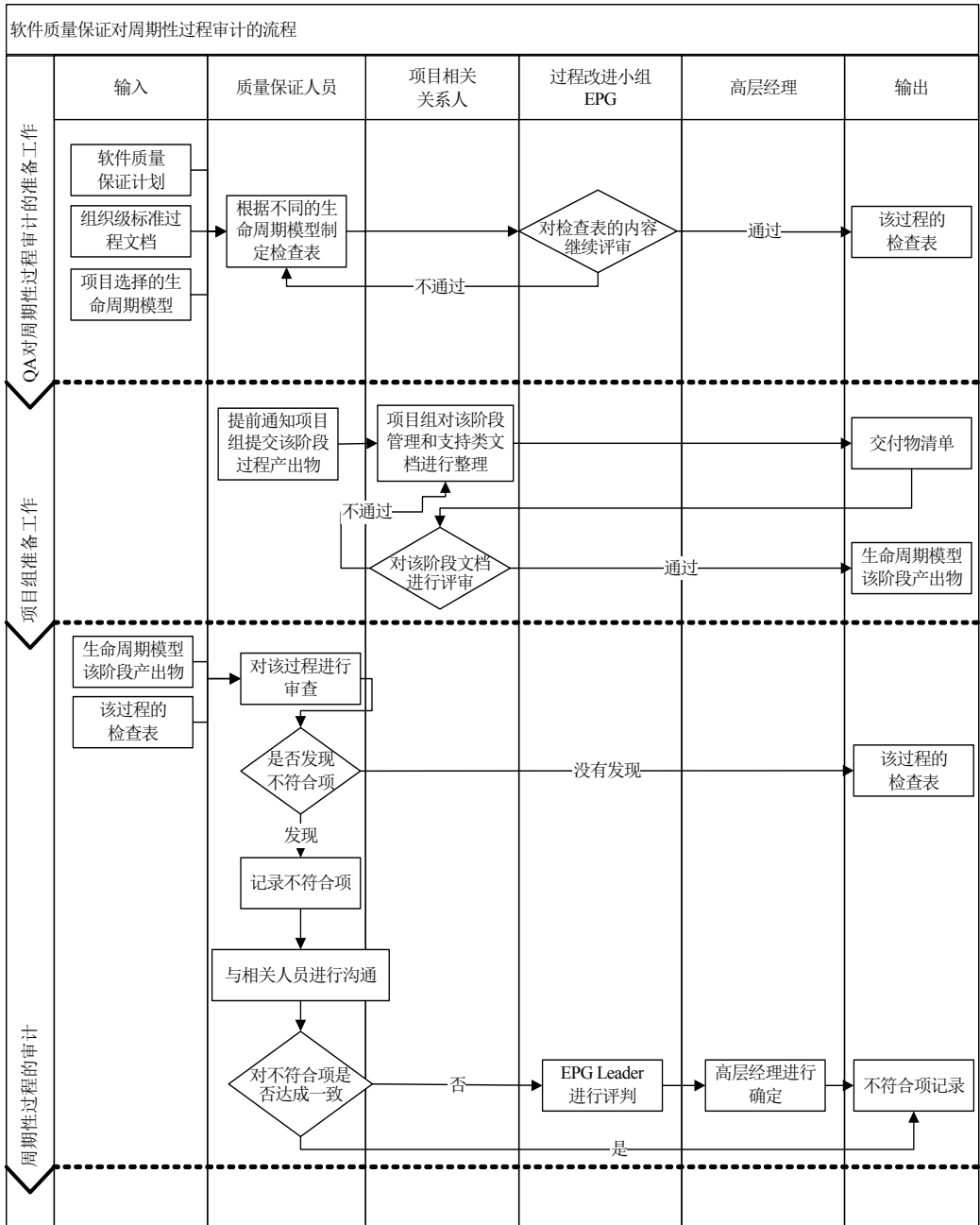


图 5-4 周期性过程的评审流程

如果要想保证交付给客户的是一个高质量的产品，那么首先就要确保工作产品是高质量的。例如常见的分布式开发最终交付客户的产品通常是由多个组件集成而来的，每个组件都是一个工作产品，如果某个工作产品出现质量隐患，那么最终交付的产品质量也一定不会太好。

软件质量保证人员（PPQA）的其中一个“P”指的就是“产品”，那么质量保证人员如何确保“产品”的质量呢？软件质量保证人员确保产品质量的方法是建立在每个工作产品高质量基础上的。首先他不能像软件测试人员（SQC）那样进行测试，因为软件测试人员进行的系统测试大多数都是针对最终的“产品”。软件质量保证人员只能在软件研发的过程中通过对每个工作产品的审计来确保工作产品的质量，例如对《软件需求说明书》内容的审查、对系统测试用例内容的审查、对产品集成后“半成品”的质量进行审查等。

可是软件质量保证人员是否有足够的时间和精力对整个项目团队所产生的每个工作产品都进行审计呢？软件质量保证人员是否对软件开发和测试技术样样精通呢？答案很简单，如果一个人能有那么多精力而且具有那么多知识和技能，那这个人就只能是超人。软件质量保证人员可以通过以下两种方式来对工作产品的质量进行审计：

① 对于文档类的工作产品，例如《概要设计文档》、《详细设计文档》等文档可以通过审查这些文档的评审报告来确保其质量是否符合要求。如果评审报告中有记录缺陷，那么质量保证人员就要对缺陷进行跟踪，确保缺陷得以解决。

② 对代码、组件类工作产品进行审查时，可以通过对单元测试报告、集成测试报告、代码走查报告的内容进行审计，如果这些报告中记录了缺陷的信息，那么软件质量保证人员也要确保这些缺陷最终得以解决。

由此可见，在对软件产品进行审计时，是通过审查相关技术人员对该工作产品质量的评审或检查记录来间接进行审计的。它关注的是这些报告的内容、评审或检查的结果，以及所发现缺陷是否得以解决。对过程进行审计关注的是该过程的流程是否符合标准以及该过程是否按要求使用相应管理类或支持类的模板，而不关注该过程输入或输出工作产品的自身质量。例如：对概要设计进行评审，过程审计关注的是本次评审的流程是否符合要求，而产品审计关注的是本次评审的结果以及评审中所发现的缺陷。例如以对《软件需求说明书》进行质量审计为例，其检查表的内容如表 5-4 所示。

表 5-4 《软件需求说明书》质量的审计

项目名称：		北极光		项目经理：		莫小贝	
QA 名称：		李大嘴		审计时间：		2009-1-6	
序 号	检 查 项			审计结果	本次审计是否适用	备 注	
1	软件需求说明书的结构是否与组织级模板相一致			√	√		
2	软件需求说明书中的每一个功能是否能被单独识别			√	√		

续表

序 号	检 查 项	审 计 结 果	本次审计是否适用	备 注
3	软件需求说明书中的每一个功能是否都有唯一的编号	√	√	
4	软件需求说明书中的每一个功能是否都有场景描述	√	√	
5	软件需求说明书中的每一个功能是否都有功能描述	√	√	
6	软件需求说明书中是否对该项目业务背景进行描述	√	√	
7	软件需求说明书中的图片或界面原型与文字描述之间不能出现二义性	√	√	
8	软件需求说明书中对某个需求项的描述不能出现前后矛盾	√	√	
9	软件需求说明书中是否有对内部接口和外部接口的描述	√	√	
10	是否有对需求进行优先级的划分	√	√	
11	软件需求说明书中对功能的描述是否可以为软件测试找到相应的证据	√	√	
12	《软件需求说明书》是否经过了评审	√	√	
13	《软件需求说明书》评审是否通过，相关人员是否都签字确认	×	√	开发人员白展堂因请假未参加本次评审
14	《软件需求说明书》评审时发现的缺陷是否都得以解决	√	√	

在软件质量保证计划中要明确以下对产品审计的内容，对产品质量进行审计的流程如图 5-5 所示：

- ① 该项目配置项列表中有哪些配置项要进行产品审计。
- ② 软件质量保证人员何时对选择的配置项进行审计。
- ③ 对配置项的审计频率如何，多久检查一次。
- ④ 对配置项进行审计时应该使用哪些检查表。
- ⑤ 这些检查表中都会包含哪些内容。
- ⑥ 对发现的不符合项如何进行跟踪和管理。
- ⑦ 软件质量保证人员与各项目关系人如何进行沟通，以及在什么情况下如何向高层领导进行反馈。

5.3 软件质量保证常见问题及案例分析

软件质量保证的工作十分重要而又特殊，但由于该项工作在业内都属于刚刚起步的阶段，因此在实际工作中会遇到各种各样的情况，以下通过几个案例对软件质量保证工作中遇到的问题进行分析。

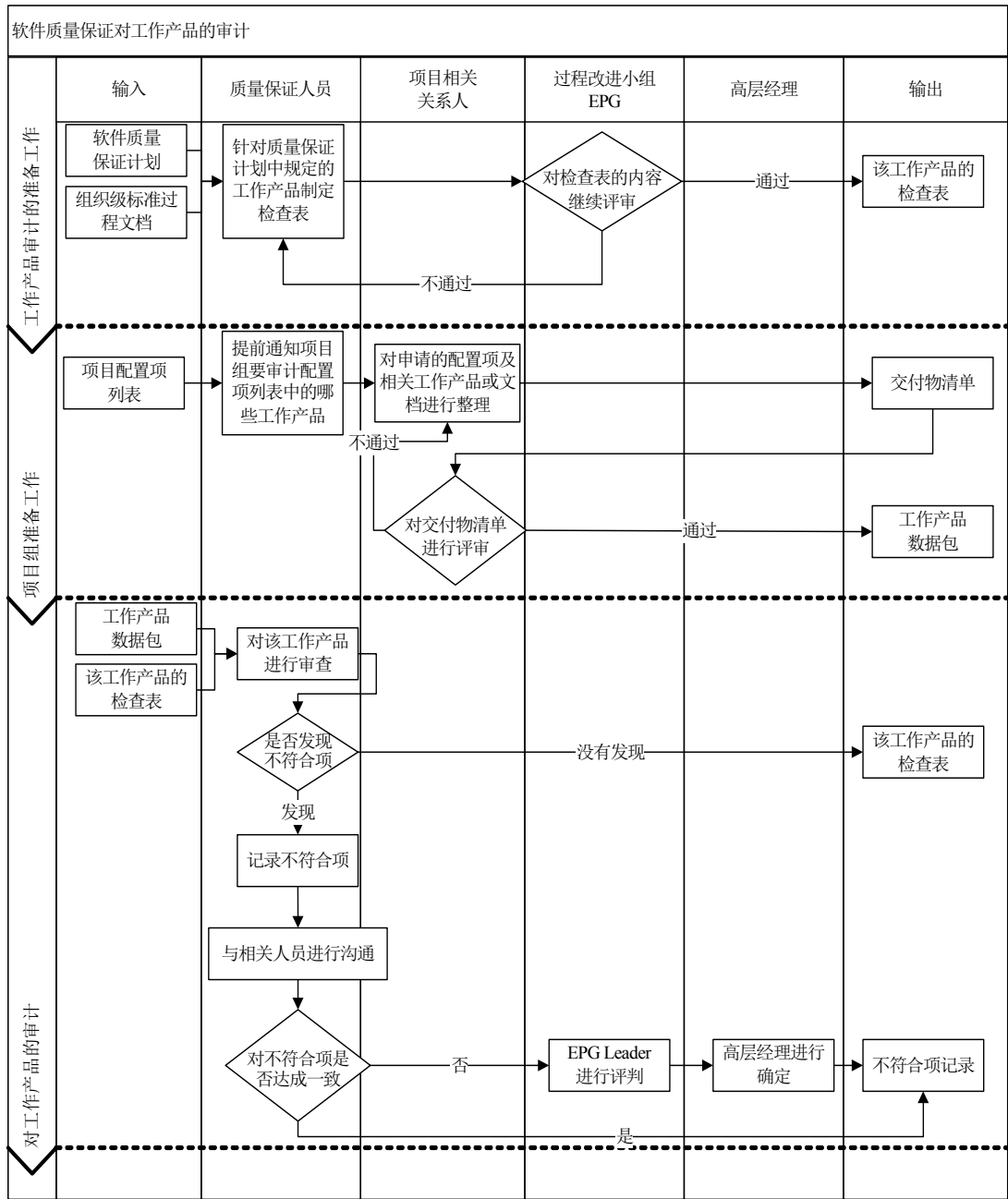


图 5-5 对工作产品的审计流程

【案例】

某软件公司即将成立软件质量保证小组，该小组将作为软件品质部门的一个组成部分，这样就可以对项目组进行独立的审计了。对于这样一个规划项目，经理小杜觉得有些不能理解，公司为什么要设立一个这样的岗位来对项目组进行监督，那么品质部门又应该由谁来进行第三方的审计呢？软件质量保证人员的工作又该由谁进行审计呢？

【分析】

项目经理小杜提出的建议非常好，既然是第三方独立审计，那么就要对软件研发的各个环节都进行审计，当然不能只针对项目组。因此公司管理层经过讨论后重新对组织架构进行了调整。软件质量保证小组仍然作为品质部门的一个部分，但是质量保证小组的工作由公司高级经理进行直接领导，这样软件质量保证人员相对于开发和测试来说就足够独立，而且在必要时他们还可以直接向高层领导汇报。高级经理将作为软件质量保证人员的 QA 来对其工作进行审计。

对软件质量保证过程审查时所使用检查表的大致内容例如表 5-5 所示，对软件质量保证过程产出的工作产品进行审查时所使用的检查表内容如表 5-6 所示。

表 5-5 针对 QA 过程所使用的检查表

项目名称：		北极光		高层经理：	秦琼
QA 人员：		关羽		审计时间：	2009-1-6
序号	检 查 项	审计结果	本次审计是否适用	备 注	
1	是否编写软件质量保证计划	√	√		
2	软件质量保证计划是否经过评审	√	√		
3	软件质量保证人员是否有统计每次对项目组审计所花费的时间	√	√		
4	是否有将审计中不符合项提交缺陷管理系统进行跟踪和管理	√	√		
5	软件质量保证人员是否有按期提交质量保证周报	√	√		
6	是否有针对项目的每个过程制订检查表	√	√		
7	是否有针对项目配置项列表中选定的配置项制订检查表	√	√		
8	质量保证人员所使用的检查表是否都经过评审	√	√		
9	对检查表评审时，EPG 人员是否有参与	√	√		

表 5-6 针对 QA 工作产品使用的检查表

项目名称:		北极光		项目经理:		莫小贝	
QA 名称:		李大嘴		审计时间:		2009-1-6	
序号	检 查 项			审计结果	本次审计是否适用	备 注	
1	软件质量保证计划的结构是否符合组织级模板的要求			√	√		
2	检查表的格式是否符合组织级模板的要求			√	√		
3	在检查表中是否有根据不同的审计重点定义有针对性的审计项			√	√		
4	提交不符合项到缺陷管理系统时是否注明发生的阶段及严重的等级			√	√		

5.4 小结

本章将软件质量保证（SQA）的含义进行了升华，并将它重新定位于 PPQA。软件质量保证人员的工作属于第三方独立的审计，他们通过对产品研发过程和过程中产生的工作产品进行审查，以实现高质量的产品目标。

5.5 思考题

- 1. PPQA 的英文全称是什么，分别代表对哪两个方面的审计？
- 2. 过程审计关注的重点是什么？
- 3. 产品审计关注的重点是什么？

6

第 6 章

软件质量管理的基石——配置管理

“配置”这个名词最早用于了解计算机的硬件组成，例如一台计算机是由 2.1 的双核 CPU、4GB 的内存和 200GB 的硬盘构成。但当今在软件工程中“配置”这个名词也被越来越多地使用，这是由代码“高内聚、低耦合”的特点造成的，软件产品也会由不同的组件拼装而成，例如某移动增值类产品现在使用的是 3.0 的网关和 2.0 的业务系统。

随着软件技术的发展，软件开发团队的规模越来越大，一个软件项目少则四五人，多则几十人，而且更复杂的软件项目会由处于不同地点的多个团队共同完成。在一些当前比较流行的软件开发模型中，例如极限式开发模型会不断推出新的产品给客户试用，软件代码、文档的版本也不断地变化。甚至有些软件项目中还出现跨越多个平台的开发，有些项目组是在 Windows 平台上进行研发，有些却是在 Linux 或 UNIX 平台中进行的，因此会带来软件研发过程中各种各样的合作问题，例如如何对当前众多版本的产品进行开发和维护、如何保证团队中所有人员使用的都是最新的程序和文档、如何能够找到某个特定版本的缺陷、如何将发现的缺陷在对应的产品中进行修复而不会遗漏其他的版本。这些都不是项目质量、进度、成本和资源的问题，而是需要依赖一套流程、制度和工具来解决的。

既然软件配置管理工作那么重要，如果一个软件项目没有开展配置管理的工作，那会带来什么样的后果呢？

虽然现在软件专业的应届毕业生就业率不高，但是对于中、高层次的技术和管理人员的需求量还是非常大的，因此软件行业的人员流动率始终居高不下。软件开发是个知识高度密集的行业，如果缺乏配置管理，那么一旦他们辞职，这些年所积累下来的行业知识、技术经验、项目代码都有可能随之消失或出现混乱。

“变化是永恒的，不变是短暂的”，软件行业更是要勇于面对变更、拥抱变更。如果没有配置管理的规范和流程，那么软件的变更将缺乏指导性，软件的研发过程也将出现混乱，那就永远也不可能达到客户的要求。

如果不开展配置管理的工作，那么项目的很多历史数据将会丢失。例如当项目计划变更的时候可能会覆盖掉原有的计划；当某个代码修改时又会覆盖掉其他人刚刚改过的内容。

有些时候项目并行开发是非常重要的，特别是在追赶项目进度时往往会采取快速跟进这样的手段。如果没有配置管理工作的支持，那么快速跟进也将只是空谈。

提高软件开发效率的手段之一就是提高代码的复用率。没有开展配置管理的公司，软件复用率通常也是很低的，因为没有人可以保证他们所复用的代码是否安全。

值得高兴的是现在的软件项目不论规模大小，不管采用的是 CMMI 还是 XP 极限开发作为指导，均使用了各种各样的配置管理工具，例如：VSS、Rational、CVS 或 SVN。但软件配置管理和版本控制并不是一回事，版本控制简单来讲就是为不同阶段开发出来的软件产品或组件通过分配不同的版本号来对其进行可回溯性控制，它是软件配置管理的一个子集，软件配置管理除了版本控制以外还有识别配置项、建立基线、建立配置库、跟踪与控制变更等众多内容。

6.1 软件配置管理概述

软件配置管理（Software Configuration Management, SCM）早在 20 世纪 70 年代美国加州大学的 Leon Presser 教授就提出了软件版本控制和变更的理论。当时在一些管理不规范的项目中，经常会出现一些被修复的缺陷又再次发生，一些新开发的功能又不见了，之前测试通过的项目又出现问题，这些情况都有可能是因为开发人员把代码互相覆盖造成的，也就是说软件产品的版本出现混乱。自从有了软件配置管理的理论，在很多大型的软件公司都设立了专职的配置管理员，并成立了软件变更控制委员会（Change Control Board, CCB）。

软件配置管理是整个软件工程的基础，它贯穿整个软件生命周期。软件配置管理的目的就是为了让整个软件研发的过程条理有序，而且可以保证软件研发过程中的各个工作产品条理清晰，并且工作产品的内容不会出错，这也就是软件配置管理所要实现的可回溯性、完整性，通过以下例子可以对配置管理的意义进行更深入的理解。

所有人都去过大大小小的超市，超市里的物品非常丰富，就算同一类的产品也有不同的供应商，例如同样的液体奶就有伊利、蒙牛、光明等不同品牌，这就好比同一个软件产品要对应不同的客户，要有不同的版本；就算是同一个供应商的同一类产品也有着不同的生产批号，类似于软件的基线名称；超市中的每个商品都是该超市的一个配置项，并且超市为每个商品都打印了唯一的条码作为其编号，这就是软件配置管理中对配置项的识别和命名。只有做到这样的细致和条理性，超市的运营才不会出现混乱。当某个厂商的某个产品要进行促销时，超市管理人员就可以很快对其价格进行变更；当超市管理人员需要查找某个商品时，只要根据唯一的条码就可以找到其存放的位置；当工作人员扫描了一个条码时就可以知道该商品的价格、生产日期等信息，而且这些信息永远都是最新的；当超市要下架某些厂商某个批次的液态奶时，就可以根据该产品的“基线”进行统一处理，而不会产生遗漏，这就是配置管理的强大作用。

为了将软件配置管理工作落到实处，应该将其工作内容和责任进行详细分工，其中高层经理要为软件配置管理工作提供必要的资源和支持；配置管理员通常要协助组织和项目组制订相关的配置管理计划，并且按照配置计划对配置项进行识别、标示及管理，保证配置库中的内容是可回溯的、完整的；过程改进小组的成员要负责开发和维护配置管理工作的相关流程及模板，并对配置管理员进行培训和过程指导；变更控制委员会要负责对项目的变更进行决策；项目经理要协助配置管理员制订并执行配置管理计划，控制处于受控状态下配置的变更和申请；软件质量保证人员要对配置管理的日常工作进行审计；软件项目组成员要严格遵循配置管理的各种规范。

可见配置管理工作是项目全体相关人员共同参与的工作，不能只依靠配置管理员来完成，而且各级管理人员应该提供相应的资源和支持。

6.2 软件配置管理流程及最佳实践

软件配置管理是一套从识别配置项到变更管理的完整体系，接下来会通过几个软件配置管理的最佳实践来详细介绍以下内容：

- 识别项目配置项
- 在指定时间将其汇总形成基线，并确保基线的完整性
- 控制配置项和基线的变更
- 对配置项的审计
- 建立变更控制委员会

6.2.1 建立基线

配置项与基线就好比“一条绳上的蚂蚱”，每个“蚂蚱”就是一个配置项，用绳子穿起来就成了基线，因此识别“蚂蚱”就成为了软件配置管理的第一步。

1. 识别配置项

配置项应该具有以下特点之一：

- 交付或非交付的产品、工作产品或工具
- 可能被两个或更多小组/人共享的工作产品
- 随时间改变的工作产品
- 具有相互依赖性的工作产品，其中一个的改变时将会影响其他工作产品
- 重要性高的工作产品

因此，软件研发过程中所产生的工作产品可以分为配置项和支持性记录两大类。支持性记录是在软件项目中不重要的，而且一旦生成就不会再修改的，例如：会议记录、周报等。配置项就是符合以上特点之一的工作产品，例如：需求文档的修改会影响到项目计划、设计文档等很多文件；项目的代码是项目团队共享的；项目计划会随着项目的进展而逐渐细化等。总的来说配置项就是纳入配置管理的工作产品，它包含交付给客户的产品、内部指定的工作产品、从第三方获取的产品或工具，以及其他用于产生或描述这些工作产品的对象。软件项目常见的纳入配置管理的工作产品举例如下：

- 各种计划
- 项目过程描述
- 客户需求
- 接口文件
- 技术设计文档
- 程序代码
- 工具、组件
- 产品数据文件
- 产品技术支持文件

如图 6-1 所示，在识别配置项时首先要根据项目的生命周期模型来定义，有些项目是没有需求阶段的，那么配置项列表中就没有诸如《软件需求说明书》之类的内容，另外参考公司其他项目的历史数据对配置项的识别也有帮助。当配置项识别出来后，要像超市中的商品那样给每个配置项一个唯一的标示，然后记录配置项的重要属性，例如：作者、创建或修改的时间等。接下来就要识别出这些工作产品纳入配置管理的时间点，例如：该工

作产品处于生命周期的哪个阶段，或者该工作产品是否应该通过测试后才能纳入配置管理。最后就是为该配置项指派相应的负责人。

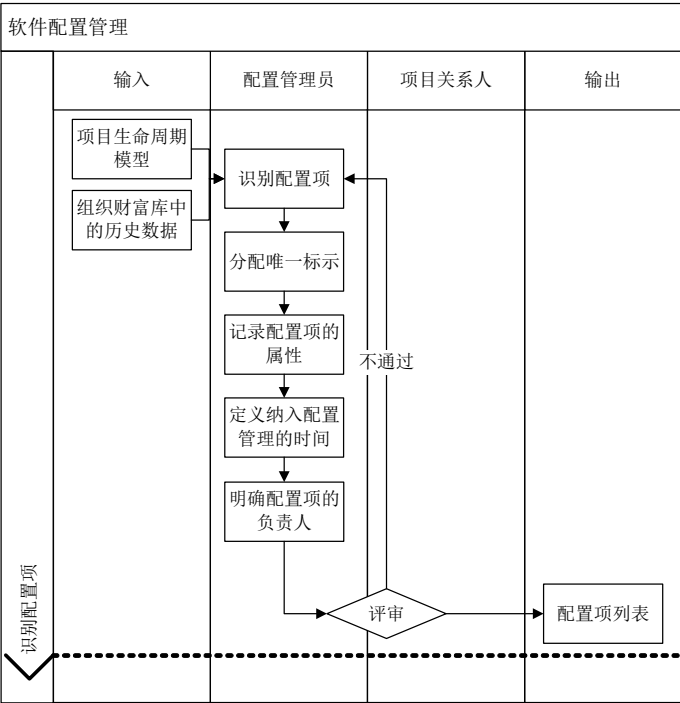


图 6-1 配置项识别的流程

很多公司在定义配置项唯一标示时通常会有自己的原则，例如：配置项前 3 位代表项目缩写，接下来的两位代表配置项产生的阶段，再后面的几位代表配置项的名称，例如：

CRM_需求_软件需求说明书

有些公司配置项的唯一标示要求全部用英文，有些公司则要求使用中文。其实标示配置项的原则首先是唯一性，其次是易懂性。如果在没有外籍员工参与的项目组中建议使用中文进行命名的效果会更好。配置项识别后将会记录在该项目的配置项列表中，其内容如表 6-1 所示。

表 6-1 软件项目配置项列表

生命周期阶段	配置名称	唯一标示	负 责 人	纳入配置管理时间点
立项阶段	项目立项书	xxx_立项_项目立项书	项目经理	需求评审通过后
需求调研阶段	需求调查报告	xxx_需求_需求调查报告	需求分析人员	需求基线发布时
	需求调研计划	xxx_需求_需求调研计划	项目经理	需求评审通过后

续表

生命周期阶段	配置名称	唯一标示	负 责 人	纳入配置管理时间点
需求调研阶段	需求分析报告	xxx_需求_需求分析报告	需求分析人员	需求基线发布时
	原始需求	xxx_需求_原始需求	需求分析人员	需求基线发布时
	软件需求说明书	xxx_需求_软件需求说明书	需求分析人员	需求评审通过后
	需求确认书	xxx_需求_需求确认书	需求分析人员	需求基线发布时
	系统需求规格说明书	xxx_需求_系统需求规格说明书	需求分析人员	需求评审通过后
项目计划阶段	项目过程定义书	xxx_计划_项目过程定义书	项目经理	计划基线发布时
	项目估算表	xxx_计划_项目估算表	项目经理	计划基线发布时
	项目计划	xxx_计划_项目计划	项目经理	项目计划评审通过后
	配置项列表	xxx_计划_配置项列表	配置管理员	项目计划评审通过后
	项目进度计划	xxx_计划_项目进度计划	项目经理	项目计划评审通过后
	配置管理计划	xxx_计划_配置管理计划	配置管理员	项目计划评审通过后
	质量保证计划	xxx_计划_质量保证计划	软件质量保证人员	项目计划评审通过后
	风险管理计划	xxx_计划_风险管理计划	项目经理	项目计划评审通过后
	度量计划	xxx_计划_度量计划	度量工程师	项目计划评审通过后
	配置库结构和权限表	xxx_计划_配置库结构和权限表	配置管理员	项目计划评审通过后
	系统测试计划	xxx_计划_系统测试计划	软件测试经理	项目计划评审通过后
	功能点估算记录	xxx_计划_功能点估算记录	项目经理	计划基线发布时
	采购计划	xxx_计划_采购计划	采购专员	项目计划评审通过后
	项目 WBS	xxx_计划_WBS	项目经理	项目计划评审通过后
	产品集成计划	xxx_计划_产品集成计划	设计人员	项目计划评审通过后
设计阶段	概要设计说明书	xxx_设计_概要设计说明书	设计人员	设计评审通过后
	详细设计说明书	xxx_设计_详细设计说明书	设计人员	设计评审通过后
	候选技术解决方案	xxx_设计_候选技术解决方案	设计人员	设计基线发布时
	技术解决方案决策分析报告	xxx_设计_技术解决方案决策分析报告	设计人员	设计基线发布时
	接口与组件对照表	xxx_设计_接口与组件对照表	设计人员	设计评审通过后
	产品集成方案	xxx_设计_产品集成方案	设计人员	设计评审通过后
编码阶段	产品代码	xxx_编码_产品代码	开发人员	单元测试通过后
	代码交付确认单	xxx_编码_代码交付确认单	开发人员	编码基线发布时
系统测试阶段	系统测试用例	xxx_测试_系统测试用例	测试人员	系统测试用例评审通过 后
	系统测试总结报告	xxx_测试_系统测试总结报告	软件测试经理	系统测试基线发布时

续表

生命周期阶段	配置名称	唯一标示	负 责 人	纳入配置管理时间点
用户验收阶段	用户验收测试用例	xxx_验收_用户验收测试用例	测试人员	用户验收测试用例评审通过后
	验收测试报告	xxx_验收_验收测试报告	测试人员	用户验收基线发布时
项目结项阶段	度量分析总结报告	xxx_结项_度量分析总结报告	项目经理	项目结项基线发布时
	过程改进解决方案	xxx_结项_过程改进解决方案	过程改进小组成员	项目结项基线发布时
	项目总结报告	xxx_结项_项目总结报告	项目经理	项目结项基线发布时
	过程改进建议	xxx_结项_过程改进建议	项目成员	项目结项基线发布时

2. 建立或发布基线

配置项在变更时会分配给它不同的版本号以对历史上的不同版本进行区分，当软件项目进展到一定阶段后经常会出现这样的情况，例如项目当前使用的是 2.0 版本的需求文档，3.0 版本的设计文档、项目组件 A 的版本是 1.0、组件 B 的版本是 3.5。那么为了保证项目团队所有成员都可以看到并使用工作产品，正确的版本将是项目成败的关键。

基线是一组经过正式评审的工作产品，是项目未来发展的基础，是项目运作的根本。一旦需要对基线中的某些配置项进行修改，就必须通过变更管理来操作。建立基线的目的就是为了让不同版本的工作产品按照项目当前所需要的信息进行组合，或者可以说是将不同版本的工作产品进行打包，这样就方便项目团队成员对所需信息和数据进行查找和使用。基线同样要有唯一的标示，这也符合软件配置管理条理清晰的目的。

基线可以对内或对外进行发布，但基线是否可以发布通常不是项目经理可以决定的，基线发布必须经过变更控制委员会的授权。变更控制委员会的工作内容通常包括以下两个方面：

- ① 对重大变更进行决策和审批
- ② 对基线的发布进行审批

变更控制委员会通常由项目经理、高层领导、软件质量保证人员等组成，根据项目规模的大小通常由 3~5 人构成。建立 CCB 的目的是可以让公司高层领导有更多途径对项目的进展情况进行了解，以便在必要时对项目的关键问题进行决策，并对项目所需要的资源提供支持。由于基线是项目阶段性的工作成果，如果这个工作成果中包含了大量缺陷，那么后果将是非常严重的，因此 CCB 也给了软件质量保证人员一个对项目进行审计、确保项目质量、给项目关系人增强信心的场合。

何时发布基线，这是在项目启动之初项目过程定义时就已经计划好的，通常基线的发布次数和时间会参考项目所选择的生命周期模型，这些信息会记录在《项目过程定义书》中。当基线建立并发布以后，配置管理员就需要记录基线中配置项的状态，这样便于项目管理人员对配置项的变更进行跟踪和管理，某项目基线状态列表如表 6-2 所示。

表 6-2 基线状态列表

生命周期阶段	基线标示	基线包含的配置项	基线发布计划时间
项目立项阶段	1.0_立项基线	xxx_立项_项目立项书_V1.0	2009-1-8
需求调研阶段	2.0_需求基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V1.0、 xxx_需求_系统需求规格说明书_V1.0	2009-1-15
项目计划阶段	3.0_计划基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V1.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V1.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V1.0	2009-2-9
系统设计阶段	4.0_设计基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V2.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V2.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V2.0、 xxx_设计_概要设计说明书_V1.0、 xxx_设计_详细设计说明书_V1.0、 xxx_设计_接口与组件对照表_V1.0、 xxx_设计_产品集成方案_V1.0	2009-3-6
编码开发阶段	5.0_编码基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V2.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、	2009-5-28

续表

生命周期阶段	基线标示	基线包含的配置项	基线发布计划时间
编码开发阶段	5.0_编码基线	xxx_计划_项目计划及从属计划_V2.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V2.0、 xxx_设计_概要设计说明书_V1.0、 xxx_设计_详细设计说明书_V1.0、 xxx_设计_接口与组件对照表_V1.0、 xxx_设计_产品集成方案_V1.0、 xxx_编码_产品代码_V1.0、 xxx_编码_代码交付确认单_V1.0	2009-5-28
系统测试阶段	6.0_系统测试基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V2.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V2.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V2.0、 xxx_设计_概要设计说明书_V1.0、 xxx_设计_详细设计说明书_V2.0、 xxx_设计_接口与组件对照表_V2.0、 xxx_设计_产品集成方案_V1.0、 xxx_编码_产品代码_V3.0、 xxx_编码_代码交付确认单_V1.0、 xxx_测试_系统测试用例_V1.0、 xxx_测试_系统测试总结报告_V1.0	2009-7-2
用户验收阶段	7.0_用户验收基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V2.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V2.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V2.0、 xxx_设计_概要设计说明书_V1.0、	2009-8-5

续表

生命周期阶段	基线标示	基线包含的配置项	基线发布计划时间
用户验收阶段	7.0_用户验收基线	xxx_设计_详细设计说明书_V2.0、 xxx_设计_接口与组件对照表_V2.0、 xxx_设计_产品集成方案_V1.0、 xxx_编码_产品代码_V3.0、 xxx_编码_代码交付确认单_V1.0、 xxx_测试_系统测试用例_V1.0、 xxx_测试_系统测试总结报告_V1.0、 xxx_验收_用户验收测试用例_V1.0、 xxx_验收_验收测试报告_V1.0	2009-8-5
项目结项阶段	8.0_项目结项基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V2.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V2.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V2.0、 xxx_设计_概要设计说明书_V1.0、 xxx_设计_详细设计说明书_V2.0、 xxx_设计_接口与组件对照表_V2.0、 xxx_设计_产品集成方案_V1.0、 xxx_编码_产品代码_V3.0、 xxx_编码_代码交付确认单_V1.0、 xxx_测试_系统测试用例_V1.0、 xxx_测试_系统测试总结报告_V1.0、 xxx_验收_用户验收测试用例_V1.0、 xxx_验收_验收测试报告_V1.0、 xxx_结项_总结及度量分析总结报告_V1.0、 xxx_结项_过程改进解决方案_V1.0、 xxx_结项_过程改进建议_V1.0	2009-8-15

从基线状态列表中不难发现每条基线都包含了上一条基线的内容，这是基线发布的一个原则，也是为了符合软件配置管理条理清晰的目的。通过这样的方式，项目组成员在查找所需要的文件或数据时就不用从多个文件夹中进行检索，直接在一个目录或数据包中查找即可。等到项目结束时，项目经理只要将最后一条基线中的内容提交到组织财富库中即可，这样就避免遗漏任何工作成果。

另外，在表 6-2 中有些基线所包含的个别配置项的版本也是不同的，这就是建立基线的目的，通过这样的方式对不同版本的配置项进行管理，方便项目管理人员了解配置项的变更情况。例如在表 6-2 中设计阶段需求发生了一次变更，从而导致项目计划出现了变更；在系统测试阶段由于某些缺陷导致系统设计的变更，从而再次引发了项目计划的变更。

基线的建立和发布通常是由项目组的配置管理员来完成的，基线发布的申请是由项目经理提出的，基线的发布的审批是由 CCB 决定的，基线建立和发布的流程如图 6-2 所示。

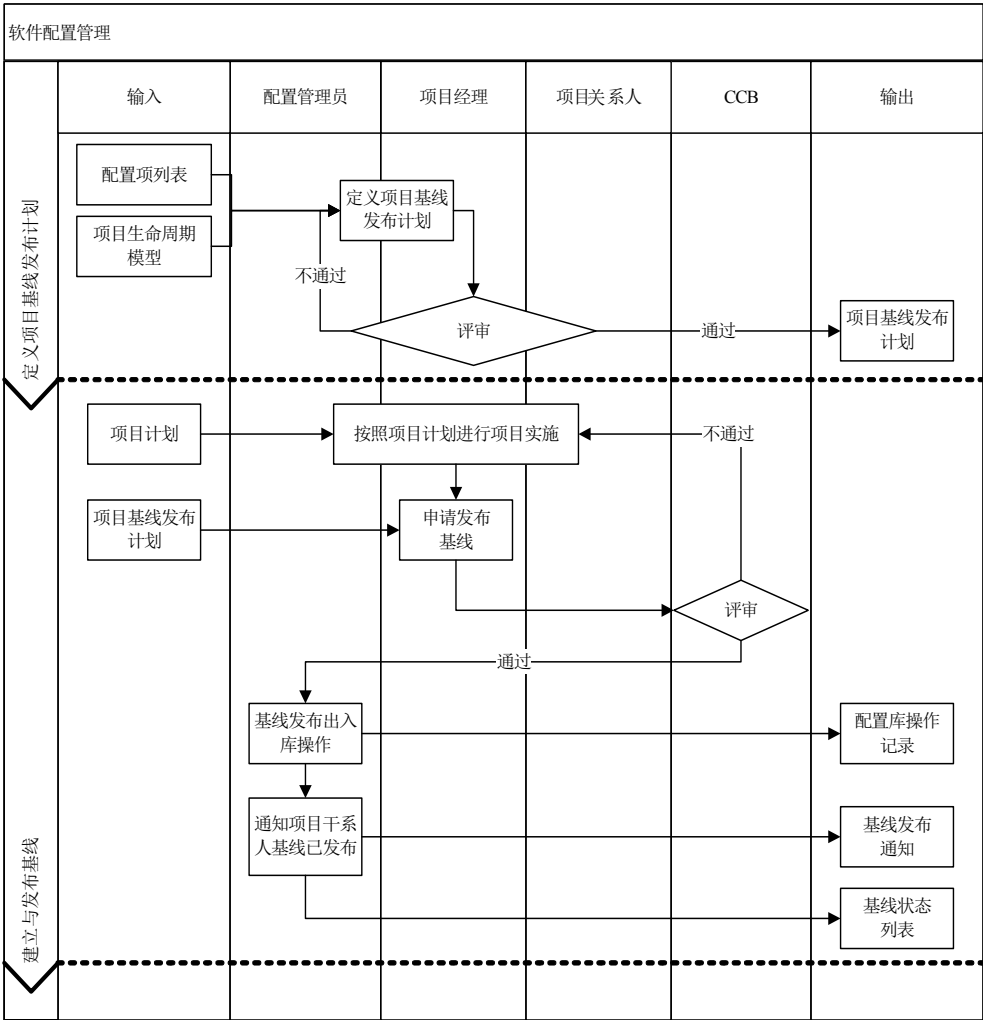


图 6-2 基线建立与发布流程

6.2.2 配置库及工具

1. 建立配置库

软件配置管理系统是配置库的统称，在项目开发过程中对工作成果起到“保护性”的作用。当今的很多配置管理工具都可以用于建立配置管理系统，但多数软件公司配置管理系统的功能划分都比较单一。配置库应该以配置项的状态进行划分，并且考虑到配置项变更管理的流程。

如图 6-3 所示，配置项的状态一般可以分为起草、待检查和已发布 3 种状态，以技术文档为例，当设计人员起草文档时可以在“开发库”中进行，此时文档处于“起草”状态；当该文档起草完毕后就需送交评审，此时文档就会被提交到“受控库”中处于“待检查”的状态，参与评审的人员就可以从受控库中获取文档并做评审前的准备工作。“受控库”中的文档是不允许任何人再修改的，如果有人修改了“受控库”中的文档，那么评审人员所获取的文件就会出现版本问题，此次评审也就失去了意义；当评审通过后，该文档就会等到基线发布时一同进行发布。当该基线发布获得 CCB 的认可后，该文档就连同其他配置项一起以基线的形式存放于“基线库”中，此时该文档的状态就是“已发布”。整个配置项从“起草”到“已发布”的过程就是软件配置管理“入库”的流程。

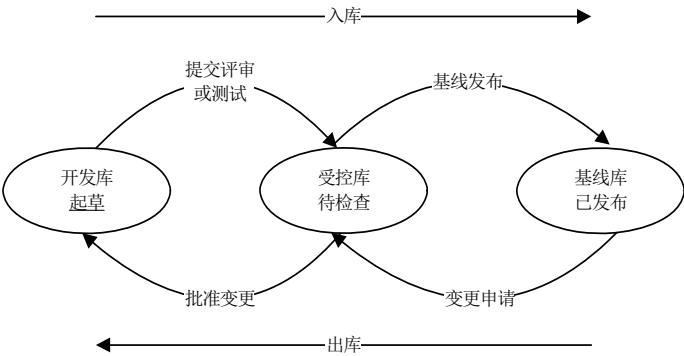


图 6-3 配置库划分原理

如果项目组需要对某些“已发布”的正式文档进行变更，那么首先要提交变更的申请，配置管理员会将“基线库”中对应的工作产品放入“受控库”中等待相关人员对本次变更进行评审，待变更评审通过后，配置管理员就会将它放入“开发库”中由相应人员对该工作产品进行修改。整个配置项从“已发布”到“起草”的过程就是软件配置管理“出库”的流程。

配置库的划分是一种逻辑上及权限上的划分，并不是物理上的区分。也就是说一个软件项目应该建立一个配置库，在该配置库中可以划分出 3 个不同的区域用于对配置项的不

同状态进行管理。如图 6-4 所示，开发库可以按照配置项的类别分为代码或文档，也可以按照项目组成员进行划分，这要根据项目的实际情况进行确定。

“受控库”主要用于存放待评审或待测试的工作产品，也就是在“开发库”和“基线库”之间进行中转。如图 6-5 所示，因此“受控库”的目录结构可以与“开发库”保持一致。



图 6-4 开发库目录结构



图 6-5 受控库与基线库目录结构

“基线库”顾名思义是用于存放已发布的基线，因此如图 6-5 所示，“基线库”中应该以各基线的名称进行划分，具体基线的名称可以参见基线发布的计划。

软件配置管理系统除了在功能上对配置库进行了划分，同时为了进一步保证各配置库中配置项的完整性，还需要对其分配不同的访问权限。

“开发库”对项目组成员几乎没有什么限制，否则会给日常工作带来不便。

“受控库”对项目组成员来说通常只有“读”的权限，只有项目的配置管理员才有“读”、“写”和“更改”的权限，因此对“受控库”的操作只能由配置管理员来完成。因为受控库中存放的是等待评审的文档或待测试的程序，假如有份文档要送去评审，会议的主持人已经将它进行了分发，可是这份文档又被作者进行了修改，那么与会人员拿到的文档可能就不是最新的，随后的评审也就没有意义了。

“基线库”对于项目组成员来说同样只有“读”的权限，也只有项目的配置管理员才有“读”、“写”和“更改”的权限，这确保了已发布基线的完整性。

2. 配置管理工具应该具备的特点

当今流行的有很多配置管理工具如 VSS、TFS、CVS、SVN 等都可以用于配置库的建立和软件配置管理的工作。那么一个优秀的配置管理工具应该具有以下特点：

• 串行和并发式的版本管理

通常小规模软件开发团队使用的是串行式的版本管理方式，也就是一个工作产品被签出或锁定后，其他人是不可以使用的，直到该工作产品被再次签入或解锁后才能被他人使用或修改。

并发式的版本管理方式通常使用在大规模的软件开发团队中，由于为了确保项目进度，往往多个项目组齐头并进，这时如果大家都需要使用某个组件或代码，可以先将其复制出

来进行使用或修改，然后多个项目组会约定一个恰当的时机对修改的内容进行合并，因此配置管理工具应该具备自动化的辅助合并的功能。

- 支持异地开发模式

很多软件项目会由处于不同地点的项目组成员共同参与，那么他们所完成的工作产品不能只依靠邮件或 FTP 的方式进行共享，这样就无法保障工作产品的版本不出现混乱，因此好的配置管理工具应该具备诸如 HTTP、HTTPS、TCP/IP、WebService 之类的功能来保障异地开发团队之间软件配置管理工作的开展。

- 灵活的权限管理和高安全性的保障

配置管理的内容都是项目至关重要的信息和数据，不同角色或级别的项目关系人应该只能看到他所需要了解的信息，这样才能确保配置项的安全性。因此对于权限分配和管理的要求非常高，为了减少配置管理员的工作量，配置管理工具应该有一套快速分配权限的机制。

- 与 IDE 集成

由于配置项的大部分内容都是软件开发、设计和测试人员所使用的，因此配置管理工具最好可以与软件开发工具或测试工具进行整合，这样可以方便项目组成员的使用。

- 自动备份与恢复功能

配置库保存内容的重要性不必再讲，因此配置管理工具需要有自动化的备份和恢复数据的功能，这是确保整个项目工作成果的关键。

- 自动记录配置项操作的功能

为了确保配置项的可回溯性，必须对配置项的每次操作进行记录。拿一个普通规模的软件项目来说，文档和代码的数量就非常大，如果依靠配置管理员进行记录，恐怕一个项目组就要配备 3~5 个配置管理员了。

- 支持日构建工具

日构建是持续集成理论的实现方式，是软件开发过程中质量保证的重要手段，配置管理工具也是日构建流程中的基础构件，因此好的配置管理工具应该支持常见的日构建工具，例如：NAnt 或 Ant 等。

3. 配置管理工具 SubVersion

SubVersion 是一个开源的配置管理工具，它具有以上配置管理工具所需要的特点，它在设计之初就确定了要弥补 CVS 功能上的不足。但 SubVersion 本身是没有图形化界面的，因此通常会将它与 TortoiseSVN 一起使用。TortoiseSVN 提供了 SubVersion 所有功能的图形化界面，接下来为大家简单介绍一下 SubVersion 和 TortoiseSVN 的功能和用法。

- 权限管理

权限管理是一个配置管理工具最基础，也是最重要的功能，SubVersion 的权限控制可

以根据需要精确到目录或文件。SubVersion 是通过以下三个文件实现权限控制的：

- `svnserve.conf` 文件：定义了 SVN 权限管理的策略。

`Svnserve.conf` 原文件如下：

```
### This file controls the configuration of the svnserve daemon, if you
### use it to allow access to this repository. (If you only allow
### access through http: and/or file: URLs, then this file is
### irrelevant.)

### Visit http://SubVersion.tigris.org/ for more information.

# [general]
### These options control access to the repository for unauthenticated
### and authenticated users. Valid values are "write", "read",
### and "none". The sample settings below are the defaults.
# anon-access = read
# auth-access = write
### The password-db option controls the location of the password
### database file. Unless you specify a path starting with a /,
### the file's location is relative to the conf directory.
### Uncomment the line below to use the default password file.
# password-db = passwd
### The authz-db option controls the location of the authorization
### rules for path-based access control. Unless you specify a path
### starting with a /, the file's location is relative to the conf
### directory. If you don't specify an authz-db, no path-based access
### control is done.
### Uncomment the line below to use the default authorization file.
# authz-db = authz
### This option specifies the authentication realm of the repository.
### If two repositories have the same authentication realm, they should
### have the same password database, and vice versa. The default realm
### is repository's uuid.
# realm = My First Repository
```

在认真阅读完该文档的内容后基本可以了解该文件的使用方法。该文件定义了用户以授权或匿名方式访问 SubVersion 时的权限，并且指定了该权限管理策略应该使用哪个“用户名及密码”列表，以及该 SubVersion 配置库中各目录、各文件的访问权限列表。它的语法及内容如下：

① **【#】** 起注释作用。

② **【anon-access】** 定义用户以匿名方式访问 SubVersion 配置库时的权限管理的策略。该选项有三个值：

a) none: 定义用户不可以使用匿名方式访问 SubVersion 配置库。

b) write: 定义用户以匿名方式访问 SubVersion 配置库时, 对 SubVersion 配置库中的文档具有“读”和“写”的权限。

c) read: 定义用户以匿名方式访问 SubVersion 配置库时, 对 SubVersion 配置库中的文档只具有“读”的权限。

③【auth-access】定义用户以授权方式访问 SubVersion 配置库的权限管理策略。该选项有三个值:

a) none: 定义授权用户不可以访问 SubVersion 配置库。

b) write: 定义授权用户访问 SubVersion 配置库时, 对 SubVersion 配置库中的文档具有“读”和“写”的权限。

c) read: 定义授权用户访问 SubVersion 配置库时, 对 SubVersion 配置库中的文档只具有“读”的权限。

④【password-db】定义 SubVersion 配置库安全策略所用的“password”文件的路径。

⑤【authz-db】定义 SubVersion 配置库安全策略所用的“authz”文件的路径。

• passwd 文件: 定义了访问 SUBVERSION 配置库的“用户名”和“密码”。

passwd 原文件如下:

```
### This file is an example password file for svnserve.  
### Its format is similar to that of svnserve.conf. As shown in the  
### example below it contains one section labelled [users].  
### The name and password for each user follow, one account per line.
```

```
# [users]  
# harry = harrysecret  
# sally = sallysecret
```

该文件记录了 SubVersion 配置库授权用户的登录名和密码, 该文件将被同目录的“Svnserve.conf”文件所引用, 其语法如下:

```
[users]
```

在该标签下定义“用户名”和“密码”。等号前面的为“用户名”, 等号后面的是“密码”, 例如: jack = 123456。

• Authz 文件: 定义了 SUBVERSION 配置库中各目录结构的权限, 以及“用户组”的信息。

Authz 原文件如下:

```
### This file is an example authorization file for svnserve.  
### Its format is identical to that of mod_authz_svn authorization  
### files.  
### As shown below each section defines authorizations for the path and  
### (optional) repository specified by the section name.
```

```

### The authorizations follow. An authorization line can refer to a
### single user, to a group of users defined in a special [groups]
### section, or to anyone using the '*' wildcard. Each definition can
### grant read ('r') access, read-write ('rw') access, or no access
### ('').

# [groups]
# harry_and_sally = harry,sally

# [/foo/bar]
# harry = rw
# * =

# [repository:/baz/fuz]
# @harry_and_sally = rw
# * = r

```

配置管理员可以在该文件中对授权的用户进行分组管理，并且记录各用户对文档或目录的访问权限，该文件被同目录的“Svnserve.conf”文件所引用，该文件的语法说明如下。

① [groups]

在该标签下定义“用户组”的名称和该“用户组”的成员列表。

② “用户组”的名称前加【@】符合与“用户名”相区分。等号后为该组成员的列表，各成员名之间使用“,”进行分隔。例如：

```
@PM = jack,susan,jim
```

③ []

定义文档目录的路径，例如：

a) [/]代表 SubVersion 配置库的“根目录”。

b) [/MyProject]代表 SubVersion 配置库“根目录”下名叫“MyProject”的子目录。

④ 定义用户对目录的访问权限。

a) 【r】：用户对该目录具有“只读”权限。

b) 【rw】：用户对该目录具有“读”、“写”、“创建”、“重命名”等权限。

c) 【'】：用户不可以访问该目录。

例如：

```

[/MyProject]
@PM = rw
Jim = r
Jack = '

```

4. TortoiseSVN 的使用介绍

TortoiseSVN 是 SubVersion 的图形化操作界面，当用户安装完 TortoiseSVN 并单击鼠标右键时会发现“右键菜单”中多了一些 TortoiseSVN 特有的内容，如图 6-6 所示。下面

将为大家分别介绍一些 TortoiseSVN 的常用功能。

● 创建 SubVersion 配置库功能

配置管理员可以通过 SubVersion 的命令行方式来创建一个配置库，也可以先在服务器上新建一个文件夹，然后右键单击该文件夹并选择【Create repository here】菜单。如图 6-7 所示，在弹出对话框中选择一种数据库类型，单击【OK】按钮完成配置库的创建工作。



图 6-6 TortoiseSVN 菜单

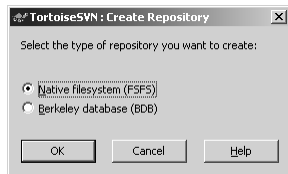


图 6-7 选择创建 SVN 的数据库类型

SubVersion 提供了两种数据库类型用于存放文档：一种是效率较高的 FSFS 类型配置库；另一种更为稳定的 BDB 类型配置库。

● 启动 SubVersion 配置库

当 SubVersion 配置库创建完毕后就需要启动该数据库，否则 TortoiseSVN 的所有操作将无法使用。由于 SubVersion 没有操作界面，因此 SubVersion 的所有操作均通过命令行方式执行，当 SubVersion 配置库建立完毕后，要想启动该配置库就可以通过以下命令行来执行。

- ① 【--listen-port】：定义 SubVersion 配置库所分配的端口号。
- ② 【-d】：指定以后台模式运行。
- ③ 【-r】：SUBVERSION 配置库的根目录路径。

范例：

启动 G 盘上名为“svnrepos”的 SubVersion 配置库，它对应的端口号是 5088。

```
svnserve --listen-port 5088 -d -r g:\svnrepos
```

● 连接并获取资料

当用户第一次连接 SubVersion 配置库并想获取资料时, 可以使用【SVN Checkout】功能。它将本地目录和 SubVersion 配置库对应目录关联起来, 并且更新 SubVersion 配置库中的文档到本地目录。

首先在客户端新建一个目录, 作为存放 SubVersion 配置库中文档副本的地方。右键单击该目录, 选择【SVN Checkout】菜单。如图 6-8 所示, 用户输入 SubVersion 配置库所在服务器的“名称”或“IP 地址”, 以及 SubVersion 配置库对应的“端口号”, 例如: SVN://localhost: 5088。单击【OK】按钮连接 SubVersion 服务器。

TortoiseSVN 弹出如图 6-9 所示的验证对话框, 用户输入“用户名”和“密码”。当用户选择了【Save authentication】选项时, 以后再对 SubVersion 配置库进行操作就无须进行验证输入。

当通过 SubVersion 的身份验证通过后, TortoiseSVN 会弹出对话框显示有哪些文件正在被更新到客户端相应的目录中。

● 添加资料到配置库

用户可以将本地的目录或文件添加到 SubVersion 配置库中。首先在本地创建一个文件或目录, 右键选择该文档或目录并单击【Add】菜单。如图 6-10 所示, TortoiseSVN 弹出对话框提示用户有哪些文件将被添加到 SubVersion 配置库中。该操作只是标识位于客户端的副本文件, 并没有正式提交配置库。

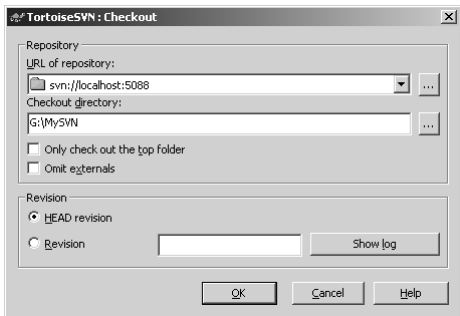


图 6-8 连接 SubVersion 配置库



图 6-9 SubVersion 身份验证对话框

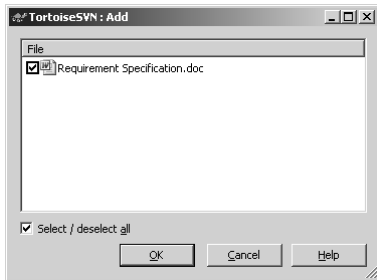


图 6-10 文件的添加列表

用户继续右键选择该文档, 并单击【SVN Commit】菜单, 此时才会更新 SubVersion 配置库中的数据。如图 6-11 所示, 用户可以在弹出的【Message】对话框中填写对本次添加操作的注释。

①【Message】填写本次变更的注释, 建议每次进行提交操作时都要填写注释。

②【Change made】显示需要提交到 SubVersion 配置库的文档列表，双击任何一个文件，都可以查看其详细信息。

③ 在【Change made】列表中的文档，用户可以通过勾选文档前面的复选框，选择哪些文件会被提交到 SubVersion 配置库。

经过提交操作后的文档或数据才会被正式提交到 SubVersion 配置库中。用户对 SubVersion 版本控制下的文档或数据进行“修改”和“删除”操作后，同样都需要执行提交操作。

● 获取最新版本

右键单击 SubVersion 客户端相应的目录或文件并选择【SVN Update】菜单。通过“用户名”和“密码”验证后，用户将获取到 SubVersion 配置库中的最新数据，如图 6-12 所示，TortoiseSVN 将弹出对话框，显示有哪些文件的最新数据被获取。

● 还原数据

对客户端副本数据所做的任何操作在没有进行提交之前，都可以取消原先的任何改动，就像许多工具都有的【Undo】功能一样。

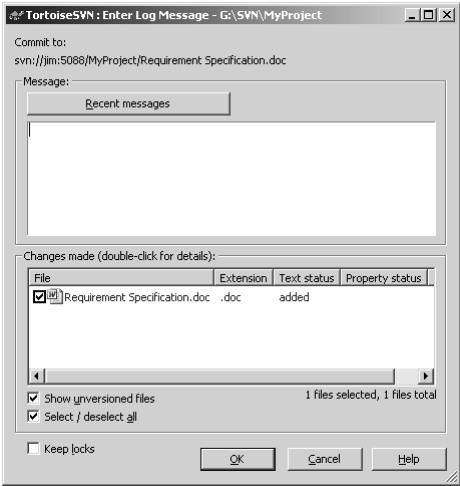


图 6-11 提交操作的列表

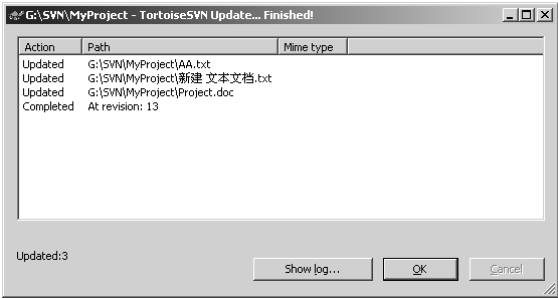


图 6-12 文件更新列表

用户右键选择某个修改后但未提交的文档或者目录，然后单击【Revert】菜单。如图 6-13 所示，TortoiseSVN 弹出需要进行还原操作的文档列表。用户可以通过勾选文档前面的复选框来选择哪些文件将被还原。

● 查询历史记录

用户可以查看 SubVersion 配置库中文档变更的历史记录。右键单击某个文档，然后选择【Show log】菜单，如图 6-14 所示，TortoiseSVN 弹出该文档历史变更信息的列表。

- ①【Revision】文档的版本号。
- ②【Actions】文档的变更行为，可以通过 TortoiseSVN 图例区分是“添加”、“修改”还是“删除”操作。
- ③【Date】文档变更的时间。
- ④【Message】文档变更时的注释信息。

• 获取历史版本数据

SubVersion 可以追溯文档版本变更的历史记录，并获取之前任意版本的数据。右键选择某一文档并单击【Update to revision】菜单。如图 6-15 所示，在【Revision】中输入所希望获取文档的版本号，然后单击【OK】按钮，该文档将被更新到指定的历史版本。

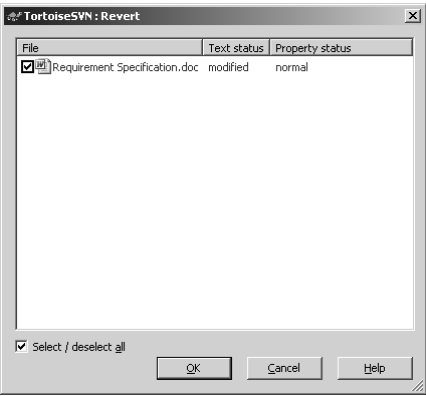


图 6-13 文档还原列表

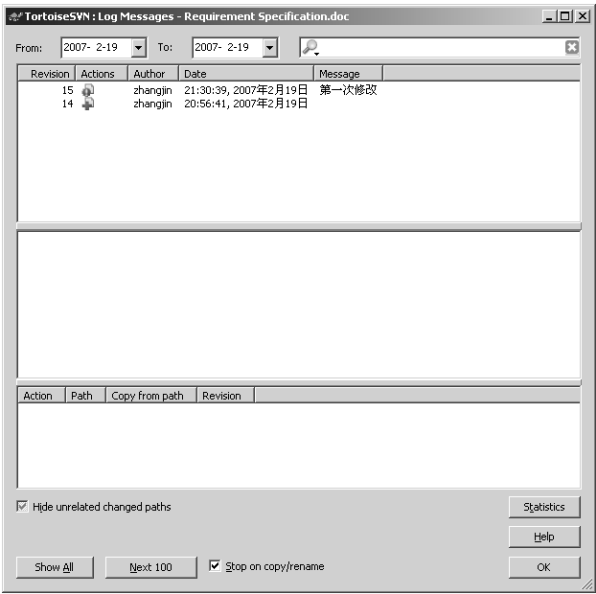


图 6-14 历史记录列表

• 重命名

用户可以对 SubVersion 配置库中的文档或目录进行重命名操作，重命名操作的原理是将原文件复制一份并用新的名称进行命名，然后将原有的文件进行删除。用户右键选择某一文档，然后单击【Rename】菜单。如图 6-16 所示，在弹出的对话框中用户输入新的文件名，然后单击【OK】按钮进行确认。重命名操作后也需要执行提交操作才能更新 SubVersion 配置库中的数据。

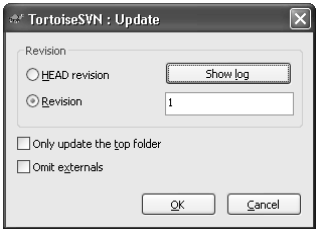


图 6-15 revision 还原版本操作

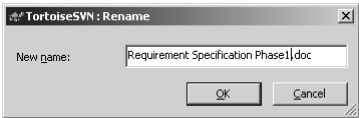


图 6-16 重命名操作

● 版本演化图

SubVersion 可以将文档之间的演变关系生成图表，使用户更清楚、更容易追溯文档的变更过程。用户右键选择某个文档，然后单击【Revision Graph】菜单，如图 6-17 所示，在 TortoiseSVN 弹出对话框中显示文档版本之间的演化关系。

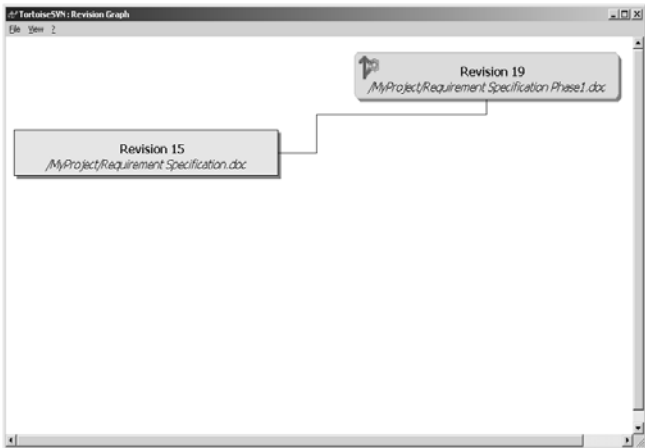


图 6-17 版本演变关系图

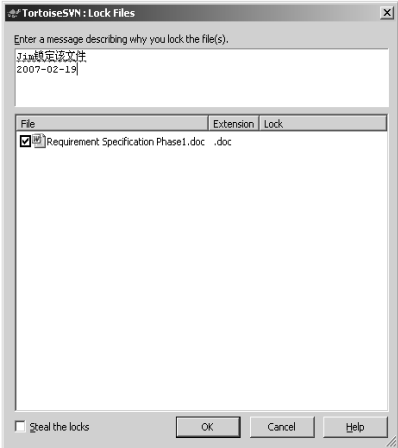


图 6-18 加锁操作

● 锁定数据

用户可以为文档加锁，保护该文档不被其他用户修改。右键选择某个文档，然后单击【Get lock】菜单。如图 6-18 所示，TortoiseSVN 弹出对话框显示有哪些文件将被锁定。用户输入锁定的原因后单击【OK】按钮进行确认。

用户可以对自己锁定的文件进行解锁。右键选择某个已经被锁定的文档，然后单击【Release lock】菜单即可解锁。

● 对比文本文件的差异

当用户修改完本地文档后，在未作提交操作之前，如果需要比对本次修改的内容与修改前有什么不同，可以使用【Diff】功能。

例如用户修改一个 Word 文档并保存内容，然后右键选择该文档并单击【Diff】菜单。TortoiseSVN 调用 Word 文件版本修订的功能进行文档修改前后的对比，如图 6-19 所示，用户能够清楚了解到本次改变的内容。

- 删除数据

用户右键选择某个文档或文件夹，然后单击【Delete】菜单可以删除 SubVersion 配置库中的文档或目录。如图 6-20 所示，在【Message】输入框中输入删除的原因，单击【OK】按钮对 SubVersion 配置库客户端副本文件进行删除。只有执行提交操作后才会更新配置库中的数据。

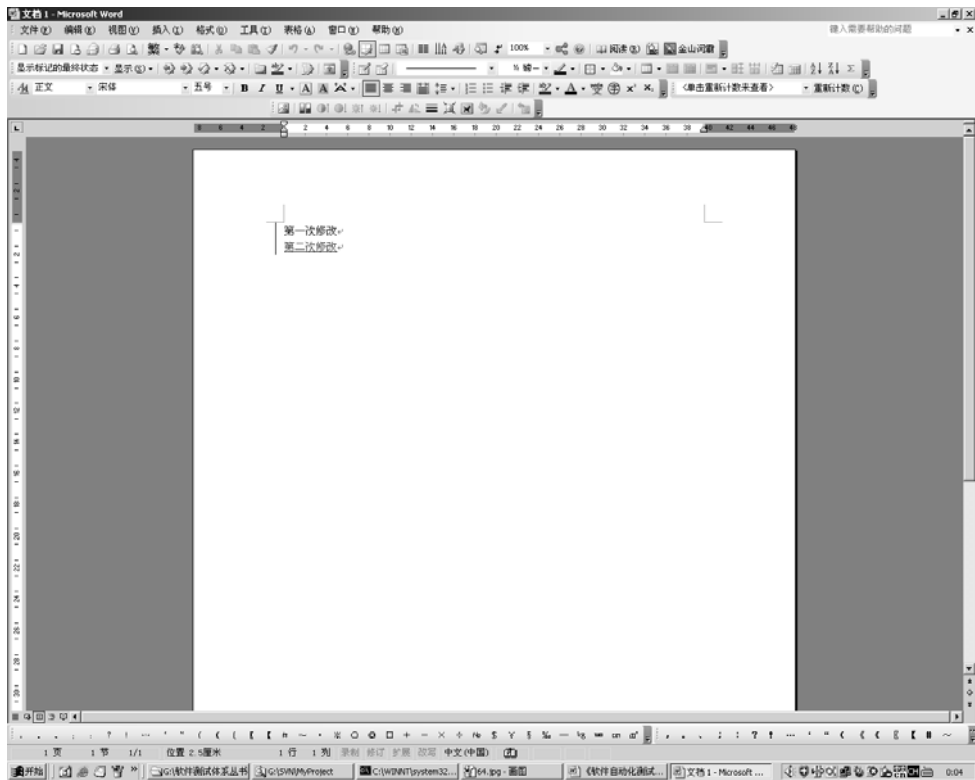


图 6-19 对比文本文件的差异

- 整理数据

在使用 TortoiseSVN 的过程中有时会出现服务器端与客户端版本不一致的情况，用户可以使用【Clean Up】功能将客户端与服务器端版本记录更新一致。如图 6-21 所示，TortoiseSVN 会提示用户更新清理工作结束。

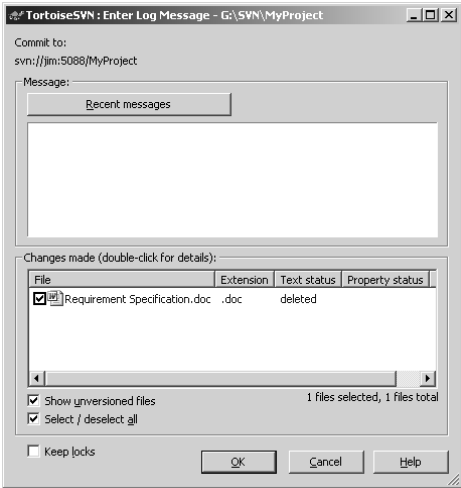


图 6-20 被删除文件的列表



图 6-21 数据整理完毕

• SubVersion 参数的基本设置

用户右键单击任意区域并选择【Settings】菜单，在这里可以配置 TortoiseSVN 工具的系统参数。如图 6-22 所示，单击界面左侧的【General】节点，可以对 TortoiseSVN 工具的常用功能进行配置。【Language】功能可以选择 TortoiseSVN 语言的种类，如果用户安装了中文补丁，则可以在这里选择中文。【Clear Now】功能可以清空用户机器中保存的登录验证信息，这样用户在下次操作时就需要输入“用户名”和“密码”。

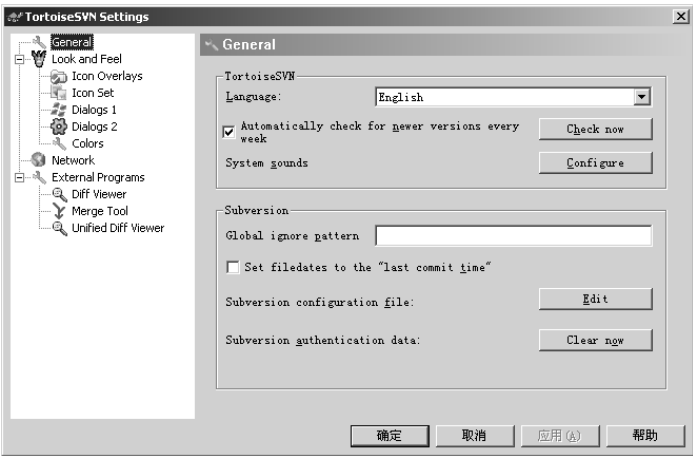


图 6-22 SubVersion 参数的基本设置

6.2.3 跟踪和控制变更

变更控制不是一个动作而是一个流程，它包括变更的申请、对变更影响的评估、对变更工作量的评估、对变更的评审、对变更的执行及效果的跟踪。为什么软件会出现变更呢？这要从软件项目的本质进行探讨。软件项目特点是“临时性”和“独特性”，就算把已经完成的项目重新再做一遍，其过程也不可能是一模一样的。其实软件开发就是将客户脑子里的主观思想转换为客观的代码行，主观的东西都是虚无缥缈的，根据人的意志会经常改变的；另外软件开发就是将企业的最佳实践转化为客观的产品，接受需求调研的客户一般都是他们行业中的佼佼者，他的思想就是企业的最佳实践，这些最佳实践也许连他的同事都不一定完全理解，何况是我们这些做软件开发的技术人员。

另外，不要以为变更给项目带来很多麻烦，有时变更也会为项目带来效益，技术的革新就是一种特殊的变更，在我国实现现代化建设的过程中不知道有多少技术革新为国家带来或挽回数以亿计的财富，在我们的软件项目中，一些新的技术或架构也一样会减少项目的开发周期和成本。

不要对变更充满恐惧，变更就像一匹充满野性的烈马，只要对它加以管理，就会收到好的效果，这样的软件项目才有挑战，项目管理人员才会有成就感。因此变更应该要有相应的流程，为了避免变更的流程对项目管理人员带来过多的束缚，给项目增加过多的工作量，在软件项目中会将变更分为两个级别：“一般变更”和“重大变更”。

凡是符合以下特点的都属于“重大变更”，这些重大变更条款的阈值是可以根据项目实际情况进行调整的，重大变更必须经过变更控制委员会的评审通过后才能实施。

- ① 项目的规模增加或减少 10%及 10%以上
- ② 里程碑延误 5 个及 5 个以上工作日
- ③ 项目组成员变动 20%及 20%以上
- ④ 项目成本累计超支 10%及 10%以上
- ⑤ 项目进度累计延迟 10 个及 10 个以上工作日
- ⑥ 在某一阶段严重级别的缺陷达到 20 个及 20 个以上

凡达不到“重大变更”条款阈值的变更都属于“一般变更”，“一般变更”不再需要 CCB 的评审，项目负责人就可以全权处理。例如：项目进度虽然有延迟，但是通过加班保证项目里程碑评审按时召开；项目成本虽有增加，但没有达到重大变更的要求。通过对变更级别的划分，给项目负责人一定的权限，这样对变更的处理就更加灵活，也不会给项目增加太多的压力，高层管理人员还可以借助“重大变更”的评审来了解项目的真实情况。

结合对配置库的划分，如果位于“开发库”内的配置项需要变更，那么相关人员可以直接进行，因为“开发库”中的配置项本来就是处于“起草”状态；如果位于“受控库”

和“基线库”内的配置项需要变更，那么项目经理就需要对变更进行评估，以判断它是否达到“重大变更”的条件，如果是“重大变更”就需要通知变更控制委员会参与评审。如果是对“基线库”中某条基线内的配置项进行变更，通常都会生成一条新的基线。

当变更发生时，首先要记录变更的来源，有些变更可能是客户方提出的，有些变更则是来自于项目组内部的，例如：软件测试人员发现的缺陷就是一种典型的变更，对变更来源的识别有助于对变更进行正确的判断。如图 6-23 所示，对变更的影响进行分析，以判断此次变更对项目成本、进度、规模、工作量等方面的影响。

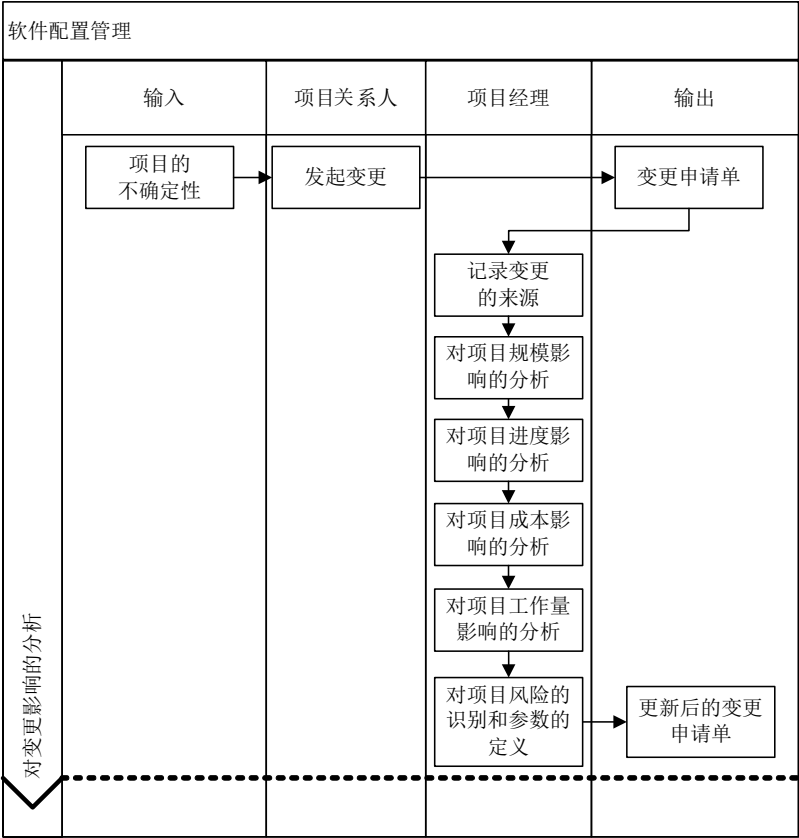


图 6-23 对变更影响的分析

变更是项目动态变化的原因之一，每当发生变更时都应该对项目的风险进行重新的识别，重新判断风险发生的可能性和严重性，制订风险的预防和应对措施，并为这些措施指派具体的负责人，最后按照风险值的大小划分对风险跟踪的优先等级。

对这些项目参数影响的估算和分析会成为变更决策的直接依据，也是判断本次变更是否为“重大变更”的主要条件。如果是“重大变更”，就需要变更控制委员会按照评审的流程对它进行评审，其流程如图 6-24 所示。最后还需要对本次变更的工作量进行统计，为日后的变更进行度量提供了依据。

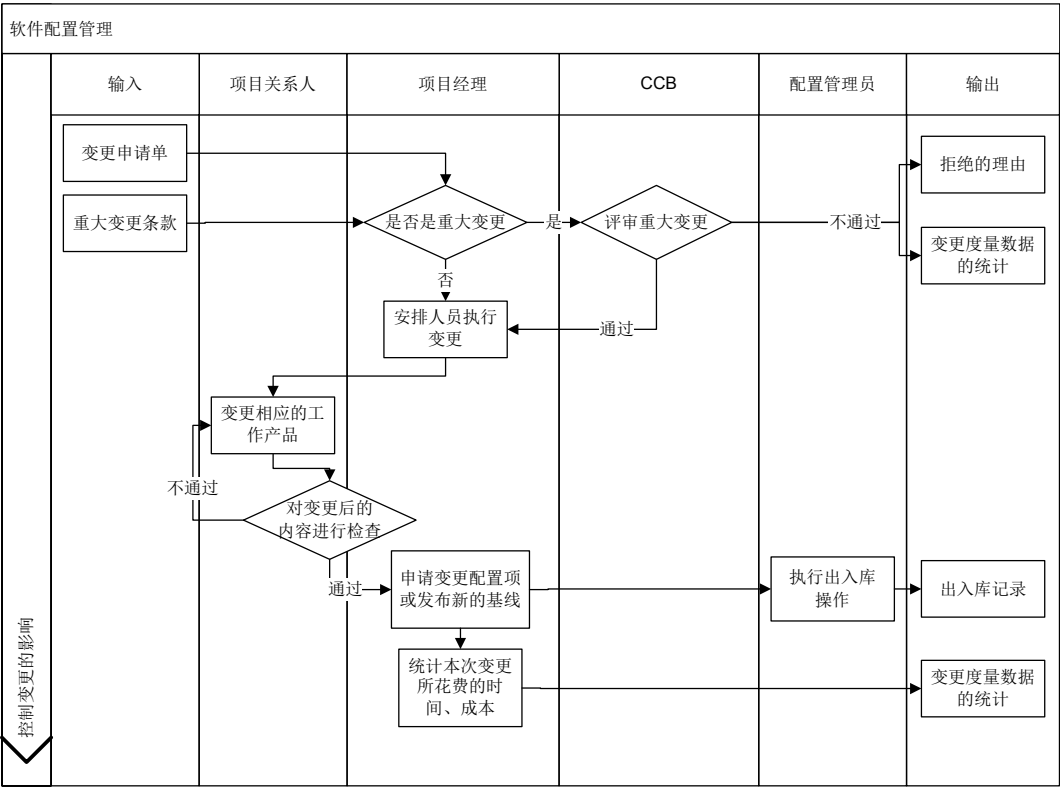


图 6-24 控制变更的影响

在对变更进行控制时，最关键的是要对相关项目关系人施加影响并达成一致。例如：争取说服客户不要对项目需求进行变更；征得公司高层对变更的认可，并申请到相应的资源；让软件开发人员理解本次变更的含义和目的等。

一旦变更被批准，那么项目管理人员就要对变更进行跟踪，确保变更得以执行，并保证变更的结果符合预期要求。项目经理可以将所有变更记录在一个列表中，通过变更的状态对其进展情况进行跟踪，常用的变更状态如图 6-25 所示。

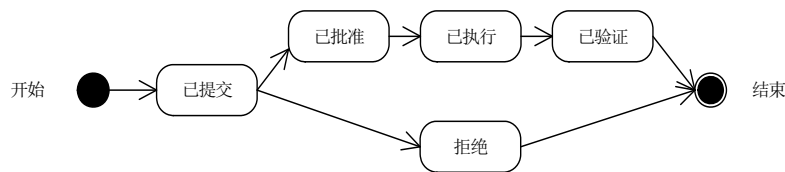


图 6-25 软件变更状态示意图

在对变更影响进行控制时，所有对变更的分析结果都可以填写到变更申请单中，变更申请单的大致内容如表 6-3 所示。

表 6-3 变更申请单

申请单编号		变更来源		提交阶段	
变更状态		变更类型		变更原因	
对象所在基线		对象目标基线		变更后版本	
提交人		提交日期		答复日期	
变更申请描述					
变更影响分析					
对项目规模的影响		对项目成本的影响		对项目进度的影响	
对项目工作量的影响		对项目风险的分析			
本次变更工作量的统计					
参与本次变更的人数		本次变更所消耗的 时长		本次变更所花费的成本	
变更执行及跟踪					
任务名称		执行人		验证人	
任务名称		执行人		验证人	
任务名称		执行人		验证人	
任务名称		执行人		验证人	
任务名称		执行人		验证人	

6.2.4 建立基线完整性

基线是用于指导软件实施的指导准则，是用于评估软件项目绩效的基准，是所有项目关系人对项目的共同理解，是指导软件测试的依据。如何确保基线的完整性对于软件项目来说是至关重要的，项目管理人员可以通过建立配置管理记录和对配置管理员的工作进行审计的方式来实现此目标。

1. 建立配置管理记录

配置管理员在日常工作中要记录以下内容：

- ① 每个配置项的出/入库记录
- ② 每个配置项的状态记录
- ③ 每个配置项的变更记录
- ④ 不同基线间的差别记录

其中配置项的出入库记录和配置项的状态通常不需要配置管理员来完成，这些可以由配置管理工具自动生成。

配置项的变更记录是为了在变更管理过程中保证配置项的完整性，以及在必要时对该配置项进行回溯。首先在每个配置项的开头都应该有一个变更记录的列表，如表 6-4 所示，用于记录单个配置项的版本变化情况，读者通过该列表可以了解到该配置项版本演变的历史，以及不同时期的作者信息。

表 6-4 计划变更记录

序 号	版 本 号	更改时间	更改内容描述	填 写 人
1	Version:0.6	2008 年 12 月 29 日	起草	张瑾
2	Version:1.0	2009 年 1 月 29 日	批准	梁满囤
3	Version:2.0	2009 年 4 月 23 日	挑战 2009 年 2 月份的工作安排	张瑾

基线是众多配置项的集合，通常使用数据包或文件目录的形式存放，虽然可以通过基线的命名规则识别其发布的先后顺序和大致内容，但是无法从基线的命名中了解不同基线间的具体差别，这就需要通过基线状态列表进行描述。如表 6-5 所示，某软件项目在系统设计阶段基线发布后进行了一次变更，重新发布了一条 4.1 版本的设计基线，项目关系人就可以很容易地通过基线状态列表找到不同基线间的差别。

表 6-5 某项目基线状态列表

生命周期阶段	基线标示	基线包含的配置项	基线发布计划时间
系统设计阶段	4.0_设计基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V1.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V1.0、	2009-3-6

续表

生命周期阶段	基线标示	基线包含的配置项	基线发布计划时间
系统设计阶段	4.0_设计基线	xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V1.0、 xxx_设计_概要设计说明书_V1.0、 xxx_设计_详细设计说明书_V1.0、 xxx_设计_接口与组件对照表_V1.0、 xxx_设计_产品集成方案_V1.0	2009-3-6
	4.1_设计基线	xxx_立项_项目立项书_V1.0、 xxx_需求_需求调研计划_V1.0、 xxx_需求_软件需求说明书_V2.0、 xxx_需求_系统需求规格说明书_V1.0、 xxx_计划_项目过程定义书_V1.0、 xxx_计划_项目估算表_V1.0、 xxx_计划_项目计划及从属计划_V2.0、 xxx_计划_配置项列表_V1.0、 xxx_计划_WBS_V2.0、 xxx_设计_概要设计说明书_V1.0、 xxx_设计_详细设计说明书_V1.0、 xxx_设计_接口与组件对照表_V1.0、 xxx_设计_产品集成方案_V1.0	2009-3-16

2. 实施配置审计

配置审计的对象是项目的配置管理员对“受控库”和“基线库”的每次操作，在日常的软件研发工作中应该对“基线库”进行重点审计。配置审计的目的是要确保配置管理员在对配置库操作时的完整性和正确性，因此配置审计的工作应该由配置管理员之外的其他人负责，这里推荐软件质量保证人员。

软件配置管理的审计分为“物理审计”和“功能审计”两类。“物理审计”是为了确保配置库的完整性；“功能审计”是为了确保配置库的正确性。

● 物理审计

物理审计的目的是为了确保软件项目配置管理员在每次出/入库操作时都没有遗漏或增加任何配置项，没有把配置项放错位置，这些对于基线的发布特别重要。“物理审计”的方式比较简单，软件质量保证人员只需根据项目组提交的“基线发布清单”制订一份检查表，再对基线中的内容进行逐一检查即可。物理审计检查表的大致内容如表 6-6 所示。

表 6-6 对项目计划基线的物理审计检查表

项目名称:		北极光		项目经理:		拉布拉多	
QA 名称:		金毛		审计时间:		2009-5-16	
序号	检 查 项			审计结果	本次审计是否适用		备 注
1	xxx_立项_项目立项书_V1.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
2	xxx_需求_需求调研计划_V1.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
3	xxx_需求_软件需求说明书_V2.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
4	xxx_需求_系统需求规格说明书_V1.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
5	xxx_计划_项目过程定义书_V1.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
6	xxx_计划_项目估算表_V1.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
7	xxx_计划_项目计划及从属计划_V2.0 是否存在与系统测试基线中，路，版本号是否正确，路径是否正确			√	√		
8	xxx_计划_配置项列表_V1.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
9	xxx_计划_WBS_V2.0 是否存在与系统测试基线中，路径是否正确，版本号是否正确			√	√		
10	项目计划基线中配置项的个数及内容是否与配置管理计划中的一致			√	√		
11	配置库的目录结构是否与配置管理计划中的一致			√	√		

● 功能审计

功能审计的目的就是为了确保软件项目配置管理员在每次出/入库操作时配置项内容的正确。配置项可能是技术类的文档, 也可能是程序代码, 那么配置管理员如何确保这些配置项的内容是正确的呢?

所有基线发布前都会进入“受控库”, 其目的是通过评审或测试来检验配置项的正确性, 因此配置管理员可以利用这些检查的结果来间接对其入库配置项的内容进行检查, 这样配置管理员就可以保证其功能的正确。功能审计的方法与物理设计相似, 都是由软件质量保证人员通过检查表的方式执行的。功能审计检查表的大致内容如表 6-7 所示。

表 6-7 对项目计划基线的功能审计检查表

项目名称:		北极光		项目经理:		拉布拉多	
QA 名称:		金毛		审计时间:		2009-5-16	
序号	检 查 项			审计结果	本次审计是否适用	备 注	
1	xxx_立项_项目立项书_V1.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
2	xxx_需求_需求调研计划_V1.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
3	xxx_需求_软件需求说明书_V2.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
4	xxx_需求_系统需求规格说明书_V1.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
5	xxx_计划_项目过程定义书_V1.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
6	xxx_计划_项目估算表_V1.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
7	xxx_计划_项目计划及从属计划_V2.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
8	xxx_计划_配置项列表_V1.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		
9	xxx_计划_WBS_V2.0 是否通过评审, 参与评审的所有人员是否都签字确认, 并没有遗留任何缺陷			√	√		

6.3 利用分支与合并进行软件配置管理工作

软件配置管理是软件工程的基础，如何在配置管理的理论下通过配置管理工具实现条理清晰并具有可回溯性的管理呢？下面将通过具体案例的分析与大家一起进行探讨。

【案例】

某软件公司产品经理小赵一直负责 OA 产品的研发，该产品已经位于 5.0 版本并正在向 6.0 版本迈进。最近有两个新的客户将购买 6.0 版本的 OA 系统，但都需要在 6.0 版本的基础上进行一些个性化的开发，那么小赵应该如何利用软件配置管理的理论来指导其工作呢？

【分析】

对两个新的客户所需要的功能进行分析，大部分的内容都是 OA 系统 6.0 版本中所具

备的，因此可以将本项目分为 3 个部分：

- ① 6.0 版本的 OA 系统
- ② A 客户的自定义功能
- ③ B 客户的自定义功能

那么小赵在制订项目计划时首先要实现 6.0 版本的功能，然后将项目团队分为两个小组，在 6.0 版本的基础上分别实现客户 A 和 B 的自定义功能。

项目进展 3 个月后，6.0 版本 OA 系统完工并通过测试，此时应该先发布一条基线来保存阶段性的工作成果。然后在开发库中利用配置管理工具将 6.0 版本 OA 系统的代码进行分支操作来应对客户 A 和客户 B 的后续开发。此时，“开发库”中就有了两套代码，分别是客户 A 和客户 B 的升级版。6.0 版本 OA 系统的代码此时位于“基线库”中并被保护起来。

如果在开发过程中两个项目小组发现 6.0 产品版遗留的缺陷，那么该缺陷肯定在 6.0 产品版和两个客户版本中都存在。所以首先修复各自发现的缺陷，然后对该缺陷的影响进行分析，看看会涉及哪些代码的更改。两个项目组可以约定一个时间互相交换《缺陷及影响分析报表》，利用配置管理工具的合并功能将受影响的代码在客户 A 和客户 B 版本中进行合并，然后通过测试来验证代码合并后的程序是否正确。

通过以上操作可以确保客户 A 和客户 B 版本中的产品版缺陷被修复，并且可以通过以上过程积累合并代码的经验。接下来就可以申请 6.0 产品基线出库，将出库后的 6.0 产品版代码放置于“开发库”中，此时需要注意不能覆盖客户 A 和客户 B 的代码。然后依据《缺陷及影响分析报表》将缺陷合并到 6.0 产品版中。

6.0 产品版合并操作结束后，就可以将 6.0 产品版的代码移至“受控库”中进行各种测试，测试通过后重新发布新的产品版基线到“基线库”中。

以上过程经常发生在以产品研发为核心的软件公司中，因此其配置管理的过程会比较复杂，希望通过本案例的分析对读者有所启示。

6.4 小结

软件配置管理工作贯穿整个项目的生命周期，项目启动后建立配置管理计划、创建配置库是首要的工作，这样才能保证项目条理清晰并且可回溯，项目组才敢继续诸如需求、设计、开发、测试等其他后续的工作。

软件开发项目经常会出现变更，软件从业人员一定要记得“变化是永恒的，不变是短暂的”，千万不要对项目中的变更产生恐惧。应该正视软件变更的存在，变更本身就是软件研发过程的一个组成部分，就像老朋友一样会经常见面。

软件配置管理的工作本身就是为了保证配置项和基线的完整性和正确性，但从事此工作的人总是有可能会犯错的，因此还需要软件质量保证人员对配置管理工作进行第三方的审计，确保万无一失。

6.5 思考题

1. 什么是配置项？
2. 基线与配置项的关系是什么？
3. 配置库通常划分为几个部分，分别是什么？
4. 通过什么样的方式对软件配置工作进行审计？

7

第 7 章

软件质量管理的客观洞察力——度量管理

现在社会遵循的是“摆事实，讲道理”，不能信口雌黄地忽悠人。软件行业做任何事情都要求非常严谨，否则当软件产品正式投入运营后将会对社会或企业带来不可预计的后果，因此软件项目的研发管理首先就应该客观、准确。在如何“讲道理”之前就应该学会客观、准确地“摆事实”，“摆事实”的唯一方法就是杜绝主观判断，而采用软件度量的方法和手段。

在以往软件研发过程中总是会出现“三拍”的现象：公司接了一个项目，高层领导认为某个项目经理技术不错，平时和自己也挺谈得来，就拍拍他的脑袋说：“项目经理就是你了”，这是“第一拍”；项目经理觉得项目不难，自己技术好，拍拍胸脯说：“领导，只要给我 5 个人就能搞定，请您放心”，这是“第二拍”；软件项目具有不确定性，项目经理又没有做好充足的准备，最后导致项目的失败，项目经理只好拍拍屁股走人了，这就是“第三拍”。这种现象在软件企业中是屡见不鲜的，公司高层领导觉得很奇怪，技术和资源等方面都比较充足，为什么项目会失败呢？项目经理也觉得很委屈，自己工作非常努力，领导对自己也很信任，但为什么项目就是做不好呢？

其实原因很简单，就是没有具备“摆事实”的基础，所有项目关系人对项目的判断和决策都只依赖于主观的判断，没有客观、准确的事实依据作为支撑。首先，公司高层领导不能依靠个人的喜好来确定项目经理的人选，对于项目经理提出 5 个项目成员的需求是否

合理应该经过客观的判断。项目经理是如何估算项目所需资源的，不能只靠拍胸脯，应该通过一系列的估算、预算数据作为提出申请的依据。除此以外还有项目成本、进度、工作量、项目规模等各种项目属性的估算和预算才能形成一份较为准确的项目计划。公司高层领导也要通过这份详细的项目计划来判断该项目经理是否胜任。只有通过这样客观的评判和决策过程才能确保项目顺利进行，软件产品的质量才能得到保证。

软件项目如何更好地“摆事实”？软件度量就是解决这个问题的最好方法。使用软件度量的数据作为研发管理的依据，不但可以提高项目计划的准确性，而且还确保了软件产品的质量，而且更可以提高企业和项目管理人员对软件项目和产品的洞察力和分析力。

7.1 软件度量管理概述

在研究软件度量之前，要先学会区分度量与测量的概念。

7.1.1 测量的基础知识

测量的概念非常简单，而且在日常生活中随时都会发生，例如测量一个人的身高是多少厘米、一个人的体重是多少公斤、一杯水的温度是多少摄氏度。由此可见测量就是通过某种刻度对物体进行刻画的过程。在整个测量过程中使用正确的测量单位是一个非常重要的环节，使用了错误的测量单位将会给理解和沟通带来极大不便。例如在中国不能说某人的身高是多少英寸，某人体重是多少磅，这样会让国人搞不明白。因此测量的刻度可以分为以下几类。

1. 标称刻度

标称刻度是指将最简单、底层的活动根据某个属性进行分类。例如可以把宗教分为佛教、道教、天主教、基督教等，可以将人种分为白种人、黄种人、黑种人等，也可以将开发模型分为瀑布式开发、螺旋式开发、迭代式开发等。定义标称刻度的要求是分类必须相互排斥，并且要覆盖到所有的内容，覆盖不到的地方可以使用“其他”来归类。但标称刻度并没有明确各类别之间的先后顺序。

2. 顺序刻度

顺序刻度是指各分类间具有一定的先后顺序，例如对职称的分类有助理工程师、工程师、高级工程师；CMMI 的等级分为“初始级”、“可重复级”、“已定义级”、“可管理级”、“优化级”。可以看到顺序刻度不但满足了标称刻度的要求，还将分类进行了排序。顺序刻度的特点是分类具有传递性，但它不能进行加减乘除的数学运算。

3. 区间刻度

区间刻度弥补了顺序刻度的不足，它可以进行加、减运算。例如开发人员小卢的生产

效率是 3 千行代码每天，小张的生产效率是 6 千行代码每天，则小张比小卢每天多生产 3 千行代码，在使用区间刻度时必须定义统一的计算单位。

4. 比率刻度

比率刻度在区间刻度的基础上更进一步，可以进行乘、除运算，因此也是最高级的刻度类型，它的特点是需要定义一个零值的点。例如小张的生产效率是小卢的 2 倍。在使用比率刻度时特别需要将其与区间刻度进行区分，例如夏天的最高温度是 40 摄氏度，冬天的最低温度是 -10 摄氏度。它们之间相差 50 摄氏度，但不能说夏天的最高温度是冬天最低温度的 5 倍。

7.1.2 度量的基础知识

在研究完测量的概念以后，接下来要讨论一下什么是度量。度量是在测量基础上进行的一系列活动，度量首先需要进行多次相同的测量工作，从而收集、存储大量的测量数据。然后将这些数据按照度量的目的进行统计和分析，项目管理人员再利用分析的结果进行判断和预测，最后将度量的结果分发给项目相关人员。由此可见测试只是度量活动的一个子集。

软件度量体系可以大致分为信息需要和目的、实体与属性、测量方法与收集存储、统计分析、决策与判断和预测、发送六个部分。在学习软件度量前需要区分以下几个概念。

- **Measure:** 指测量，是对某个对象的范围、数量、大小等属性提供一个定量的指示。
- **Measurement:** 进行测量的一种方法。
- **Metrics:** 度量就是对一个对象的属性进行一个定量的分析。
- **Indicator:** 指示器就是将统计分析后的度量结果展现出来，供相关人员进行分析和决策。

下面通过一个简单的例子来对度量进行理解：

某地选择了一所小学进行国民体质健康调查，需要抽查该学校 4 年级到 6 年级学生的身高和体重，从而判断当地 10~13 岁儿童的肥胖比例是否符合国家要求。

- **测量 (Measure):** 体重 (kg) 和身高 (cm)。
- **测量的方法 (Measurement):** 使用秤去称量体重，用尺子去量身高的行为。肥胖指数 = 体重 / (身高 - 105)。
- **度量 (Metrics):** 10~13 岁儿童的肥胖比例。
- **指示器 (Indicator):** 10~13 岁儿童各年龄段肥胖分布的比例，以及是否符合国家要求。

在本次调查中分别抽取了 10 岁儿童 90 人，11 岁儿童 86 人，12 岁儿童 95 人，13 岁儿童 90 人。经过统计和分析，如图 7-1 所示，该地区 13 岁儿童的肥胖比例较大，超过了国家制订的儿童肥胖比例应该控制在 50% 以内的规定，因此建议该地区儿童要加强体育锻炼。

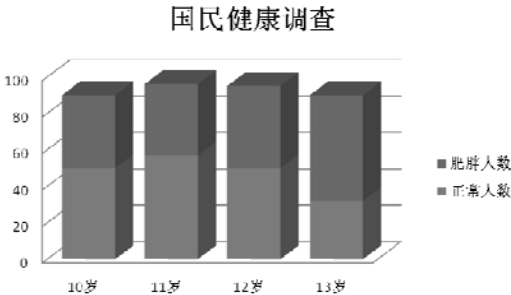


图 7-1 某小学国民体质健康调查

7.2 软件度量管理流程及最佳实践

软件度量的工作涉及软件产品研发的方方面面，贯穿于整个项目的生命周期。软件度量不但可以客观反映项目的真实情况，而且可以根据度量的结果来预测项目未来的发展。要想让软件度量发挥效力，就必须做好度量管理计划，明确度量的范围，并让公司提供相应的度量资源。

软件度量管理的体系模型如图 7-2 所示。度量的工作需要投入一定资源，也需要花费一定工作量，因此不是所有度量都需要做，原则上只做对公司和项目有用的度量，因此在度量体系模型中组织或项目的需要是发起度量的依据，这些度量的依据就通过一个个度量的目标进行体现。例如：公司高层领导有一天想看看公司所有项目的进展情况，度量的目标就是项目的进度偏差；公司领导想知道经过几年的过程改进后，公司的生产率是否得到了提高，度量的目标就是组织级生产率；项目经理想知道哪些模块的质量最稳定，度量的目标就是该项目缺陷按模块分布的情况。

确定了度量的目标后就等于确定了度量的范围，然后就可以识别所需度量的对象和属性，再找出适当的度量方法，最后根据决策准则和度量的结果来支撑整个度量的目标。

7.2.1 软件度量的目标

度量的需求就是对信息的需要，正确理解度量需求对制订一个合理的度量目标是至关重要的，下面将通过一个例子来对它进行讲解。

如果公司领导要求你测量水的温度，那么你会如何去做？也许你觉得这个问题很简单，当然是找个温度计测量一下了。对不起，如果你是这样想的，那么你就落入软件度量的一个圈套里了，因为你忽略了提出这个度量目标的人对该信息的需要。你应该先反问他为什么要你去做这个度量，他想通过这个度量了解哪些情况。因为不同的信息需要在进行度量时所选择的方法是不尽相同的。

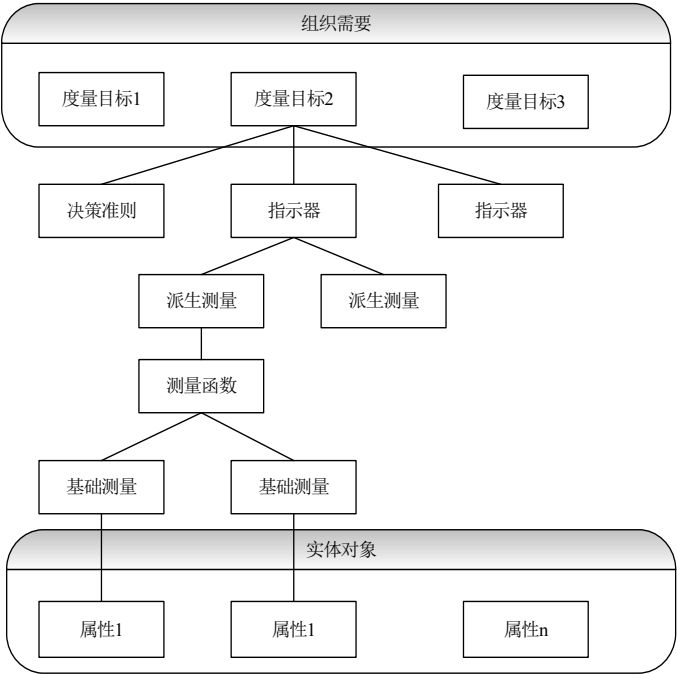


图 7-2 软件度量体系模型

如果这个度量的目的是为了判断冲茶的水是否已经烧开，那么通常情况下没有人会使用温度计进行测量，只要观察水是否煮沸，茶壶的盖子是否被水蒸气顶开就完全可以进行判断。

如果这个度量的目的是为了判断水温是否可以用于洗澡，那测量的方法就更加简单，用手感觉一下水温是否舒适即可，这也更加没有必要使用温度计进行测量。

通过以上这个小小的例子就可以看出度量对信息的需要是多么的重要，同样的度量目标在不同的信息需要下都会有不同的度量方法。如果不能清楚地理解公司或项目组对度量的需要，那么设计出来的度量目标和度量方法也将无法使用。

软件产品的度量需要通常来自于 3 个方面，其内容如表 7-1 所示。

表 7-1 软件产品的度量需要

度量的维度	侧 重 点	度量的目标
项目级别的度量	客观反映当前项目的情况和状态，针对具体项目开展相关度量的工作，用于对项目的预测和控制，对项目级别的度量具有战术层面的含义	项目的规模、成本、工作量、进度、生产力、风险发生的情况、同行评审的投资回报率等

续表

度量的维度	侧 重 点	度量的目标
产品级别的度量	客观反映当前组织或项目产品的质量状况，用于对产品质量的预测和控制	产品的缺陷率、各模块缺陷的分布情况、项目团队成员缺陷的产生情况、生命周期各阶段缺陷的产生情况、缺陷在不同严重等级下的分布情况等
过程级别的度量	客观反映组织的管理情况，对过程的度量更具有战略层面的含义	组织级的生产率、成熟度等

在不同度量需要下，不同人群的关注点都不尽相同，根据不同人的关注重点可以制订相应的度量目标和决策准则，以下列举了一些不同人群所关注的内容。

企业经营者关注与：

- 客户的满意度
- 项目风险的情况
- 项目收益情况
- 项目综合绩效情况
- 产品发布后的缺陷率
- 项目估算、预算的准确度

项目管理人员关注与：

- 开发过程中生命周期各个阶段的费用
- 项目团队成员的生产率
- 项目团队成员的缺陷率
- 项目团队成员的绩效情况
- 项目风险情况
- 项目的进度偏差
- 项目的成本偏差

项目团队成员关注与：

- 个人的工作量
- 个人任务的进度偏差
- 个人任务的质量状况

根据度量需要来制订度量目标时同样可以采取逐级分解的方法，如图 7-3 所示，要衡量一个软件企业的成熟度，可以先将其分解为软件过程、软件效率和软件成本三个方面，然后再将它继续分解，直到可以准确定义为止。例如：将软件过程的效率分解为两个目标，

分别是生产率、时间和进度。

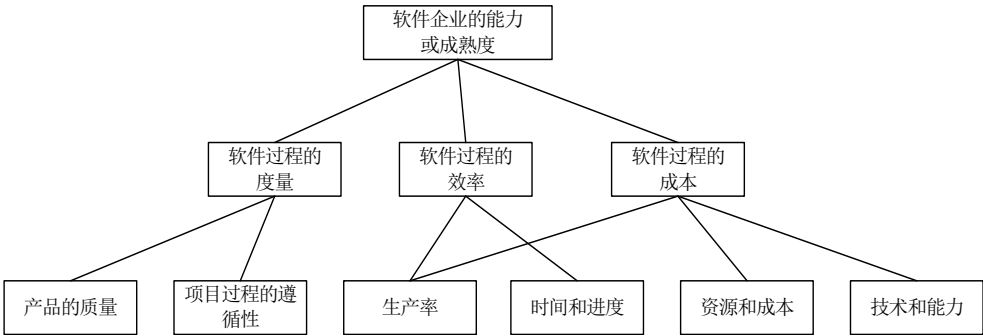


图 7-3 度量目标分解示意图

7.2.2 软件度量的实体与属性

识别实体是软件度量的基础，实体也就是被测量的对象，它可以是一个工作任务，也可以是一个交付的成果，或者是一个软件生命周期中的某个过程。属性就是该实体上的特征，一个实体可能有很多不同的属性，但并不是所有属性都适合测量。例如要度量一个班学生的平均身高是多少，那么测量的实体就是学生，测量的属性就是身高。在软件项目中常用的实体与属性如表 7-2 所示。

表 7-2 软件度量中常用的实体与属性

项目计划和管理类		
度量的目标	被测量的实体	被测量的属性
项目实际进行了多少个月	完整的项目	持续时间
项目的规模大小	代码	行数
	项目需求	功能点个数
项目的进度偏差	完整的项目或项目的某个阶段	进度的实际值和计划值
项目的成本偏差		成本的实际值和计划值
项目计划估算准确性	项目的各种参数	项目各种参数的估算值与实际值
项目的生产率	完整的项目或项目某个阶段	代码行和工时数
		功能点和工时数
项目发生了多少次变更	变更申请单	个数
同行评审的投资回报率	同行评审会议	与会人员的个数,与会人员的单位成本,会议的时长,缺陷的个数,缺陷的价值
编写文档的工作量	项目的各种文档	各种文档的页数

续表

项目计划和管理类		
度量的目标	被测量的实体	被测量的属性
项目各阶段工作量的分布情况	完整的项目或项目某个阶段	功能点的个数，项目生命周期阶段分类
		代码行的个数，项目生命周期阶段分类
项目培训成本	培训类的任务	资源的个数，资源的单位成本，培训任务的时长
配置管理员的成本	配置管理员	配置类任务的总时长，配置管理员单位成本
QA 质量保证人员的成本	QA 质量保证人员	质量保证类任务的总时长，QA 人员单位成本
度量工程师的成本	度量工程师	度量类任务的总时长，度量工程师单位成本
产品平均缺陷率	每千行代码和缺陷	缺陷个数
每个模块的缺陷率	各个功能模块和缺陷	缺陷个数
每个项目团队成员的生产率	项目团队成员	代码行和工时数
每个项目团队成员的缺陷率	项目团队成员和缺陷	人和缺陷个数
完成某个任务所花费的成本	任务和资源	资源的单位成本，任务的时长
缺陷修正率	完整的项目或项目某个阶段产生的缺陷	状态为 Closed 缺陷个数，缺陷总个数
缺陷未修复率	完整的项目或项目某个阶段产生的缺陷	状态为 Opened 或 New 的缺陷个数，缺陷总个数
缺陷未复查率	完整的项目或项目某个阶段产生的缺陷	状态为 Fixed 或 Rejected 的缺陷个数，缺陷总个数

7.2.3 软件度量的方法

软件度量的方法分为直接测量和间接测量。直接测量是指该实体的对象不涉及其他实体中的对象，例如度量某软件项目的实际成本，则该度量使用的方法是直接测量，因为统计项目的实际花费只需要简单的累计求和即可，不需要使用其他实体对象的数据或信息。如果要测量该物体的密度就需要通过该物体的重量和体积进行计算，这种方法就是间接测量的方法。

测量的方法也可以分为主观方法和客观方法。主观方法是量化了的判断或估计，例如在功能点估算中对复杂度的分类就使用了主观方法，它分为高、中、低三类。客观方法是基于数字的一种量化方式，例如代码行数就是一个以数字为基础的客观测量的方法。在确定测量

方法时最好选择客观的测量方式，因为客观的测量方式更为准确，而且具有可重复性。

根据度量中所使用的方法不同，度量的数据也可以分为基础数据和派生数据两种。基础数据就是直接测量中所使用的数据，该数据的特点是都属于同一实体对象的同一类型的属性，例如项目实际成本的对象是项目本身，属性是该项目所花费的成本。这个度量数据的实体和属性都是单一的。派生数据是指经过间接测量后得到的结果，是经过一系列数学运算或者经济模型将直接数据转换后得到的结果。例如产品缺陷率就是将两个基础数据“缺陷个数”和“功能点个数”进行除法运算而得到一个比值，该比例就是度量的派生数据。软件度量过程中基础数据与派生数据、直接测量与间接测量之间关系如图 7-4 所示。

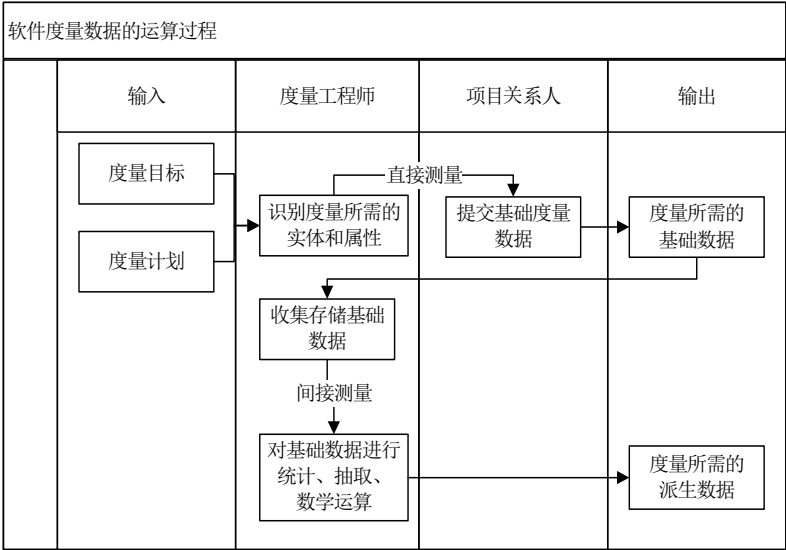


图 7-4 软件度量数据的运算过程

软件度量结果是否准确，这与收集得到数据的有效性、准确性有着直接的关系。开展度量工作的前提有以下 3 点：

① 在进行软件度量前要统一每个度量目标中测量方法所使用到的单位。例如：某些项目团队成员提交的当天工作量是以代码行为单位的，有些则是以功能点为单位的，那么单位的不统一将无法对度量的数据进行直接测量或间接测量。

② 度量工作涉及项目的所有关系人，所以必须全员参与。例如：只有部分项目团队成员参与了度量工作，那么收集上来的数据将不能体现项目的完整情况，因此也失去了度量目标所蕴涵的意义。另外参与度量的团队成员在日后也很有可能效仿其他人回避度量的工作，最终导致度量工作无法开展。

③ 度量涉及的人员广，并且有一定的工作量，因此需要高层领导的支持。

7.2.4 软件度量的指示器

指示器是进行一次或多次测量后得到结果的体现，它是支持用户分析和决策所导出的信息，这种信息通常以图形或图表的形式进行展现。指示器通常用来将实际值与期望值进行比较，期望值有可能是一些历史的数据，也有可能是项目计划的数值，或者一些限定值和阈值。指示器配合决策的准则就可以支持度量的目标的信息需要。软件开发项目中常见的指示器类型有以下几种。

1. 散点图

散点图可以用于表示两个因素之间可能存在的一种因果关系，如图 7-5 所示，缺陷数与软件的规模之间存在着一种因果关系。

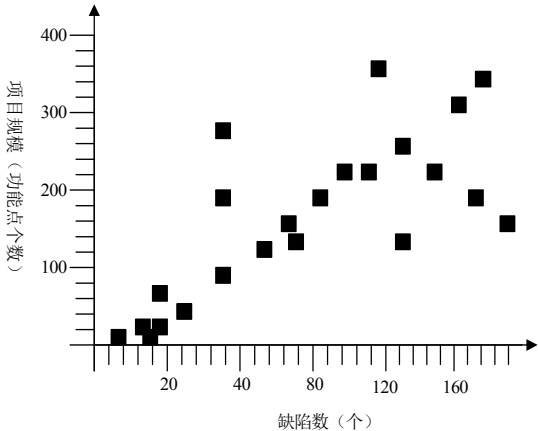


图 7-5 散点图

散点图具有对未来预测的能力，如图 7-5 所示，当项目组要完成一个软件规模达到 500 个功能点的项目时，该项目所产生的缺陷个数大约会是 200 个左右。在项目计划时对缺陷个数的预测，就可以预先在项目进度计划中安排一定数量的修改缺陷的时间，这样的项目计划才会更加合理。

2. 直方图

直方图用于表示某个事件或事物，在一个连续的时间或观察值内的变化或发生的频率。如图 7-6 所示为某软件项目生命周期各阶段缺陷产生的情况。

直方图不具有对未来预测的能力。

3. 对比直方图

对比直方图与直方图的用法十分类似，但对比直方图比直方图多了在某个时间段或观察值内，对比某个事件或事物的不同属性间的功能。如图 7-7 所示为某项目在生命周期的各个阶段产品遗留缺陷的数量，公司管理层通过对该图的分析可以确定，该软件项目不能算作最终的完工。

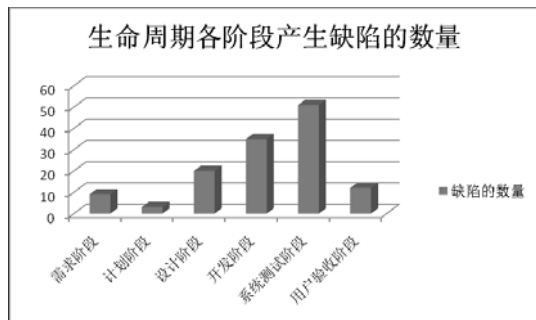


图 7-6 直方图

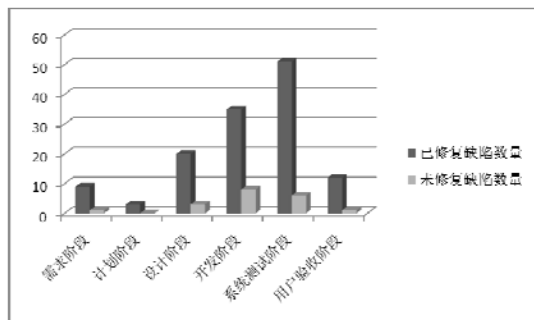


图 7-7 对比直方图

对比直方图也不具备对项目未来预测的功能。

4. 比例直方图

比例直方图的用法与对比直方图非常相似，可以用于观察某个时间段或观察值内，某个事物或事件各属性所占百分比的情况。如图 7-8 所示为某软件项目在各个阶段未修复的缺陷占总缺陷数的百分比。

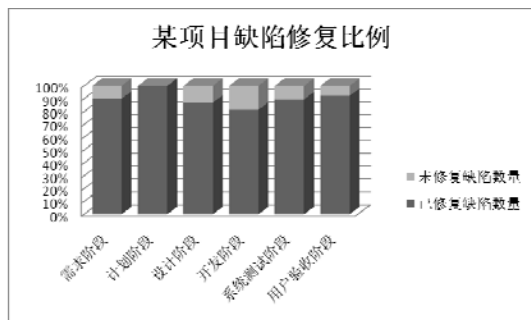


图 7-8 比例直方图

5. 趋势图

趋势图的横坐标是时间，表示随着时间的变化某个事物或事件发生的情况，用于对观察事物或事件发展趋势的图形。趋势图对未来具有预测的功能。从趋势图的发展趋势来看

可以划分为上升趋势图和下降趋势图。

项目成本的累计花费是具有上升趋势的，如图 7-9 所示，某软件项目在开发阶段已经累计了 16 万元的花费，由此可以预测，当该项目完工时项目总成本将在 20 万以内。

项目待完成的工作量随着时间发展具有下降趋势，如图 7-10 所示为某软件项目从 1 月份开工时 300 人/天的工作量发展到 8 月份还剩下 100 人/天的工作量，按照这个趋势，完成本项目还需要 4 个月。

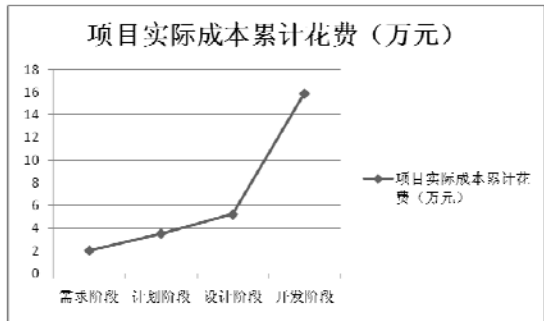


图 7-9 上升趋势图

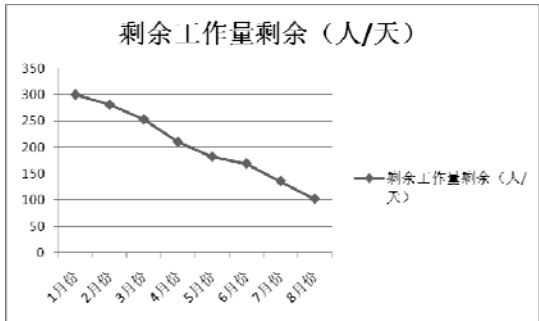


图 7-10 下降趋势图

6. 帕累托图

帕累托图是根据帕累托理论得来的，帕累托理论是对以下现象进行了概率的统计，例如：20%的人掌握了 80%的财富；80%的缺陷是由 20%的原因造成的。帕累托图将事物或事件发生的原因进行分类，按照从高到低的方式进行排序。

帕累托图在软件研发过程中经常用来查找关键性问题，这是快速提高软件产品质量的重要方法。如图 7-11 所示，某软件项目在开发测试阶段对单元测试的结果进行分析，将所发现的问题进行分类，按照帕累托理论，只要将代码进行优化，严格遵循编码规范，那么软件产品的质量将会得以提高。

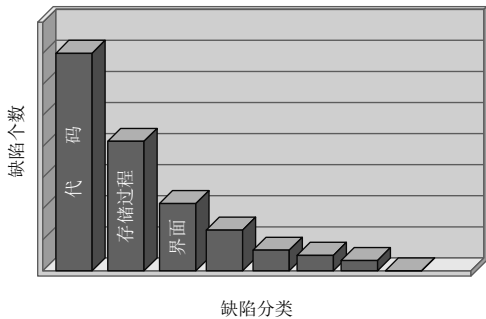


图 7-11 帕累托图

7. 控制图

控制图可以对开发过程中的各种参数进行采样，科学的分析它是随机波动还是异常波动，从而对开发过程中的异常趋势提出预警，以便项目管理人员及时采取行动，消除异常从而达到提高和控制质量的目的。通常只要在控制界限内的过程结果都是可以接受的，如果过程结果超出了控制界限，通常称其为失控。如果有连续的 7 个或 7 个以上的点分布在中心线的同一侧，或者出现同向发展的趋势，即使它们都处于控制界限内，也意味着它出现了一定的问题或者受到了外界因素的干扰，此时也将视为失控状态。

如图 7-12 所示，某软件项目按月对严重程度为 1 和 2 的缺陷发生情况进行统计，在软件研发过程中，5 月及 6 月的质量最不稳定，经过调查研究发现 5 月及 6 月的项目进度出现延迟，项目组为了赶进度而造成了产品质量的下降。

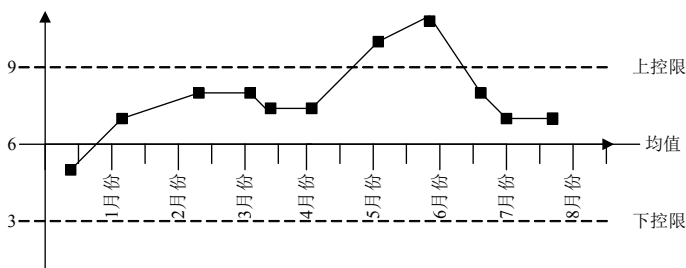


图 7-12 严重级别为 1 和 2 的缺陷统计控制图

7.2.5 软件度量管理的流程

通过以上内容的讲述，本节将软件度量管理的流程进行总结，如图 7-13 所示。软件度量管理的工作是周期性反复进行的过程，在项目计划阶段，当确定了度量的目标和计划后，就要在项目中各个阶段反复测量，直至项目结束。

7.3 软件度量管理常见问题及案例分析

7.3.1 如何提高软件度量的准确性

【案例】

某软件公司 CRM 项目的项目经理小赵正在写项目结项报告，在这个为期 6 个月的项目中一切都比较顺利，没有发生需求的变更。可是在项目结束时对项目规模进行最后一次估算，其结果比项目计划时估算的结果多出 9%，这个偏差已经算是相当大了。假如真是项目计划时对项目规模进行度量的结果是错误的，那么该项目过程中出现的种种赶工、加班的原因就找到了。但如何证明那次度量的数据是正确的，应该如何提高软件度量的准确性呢？

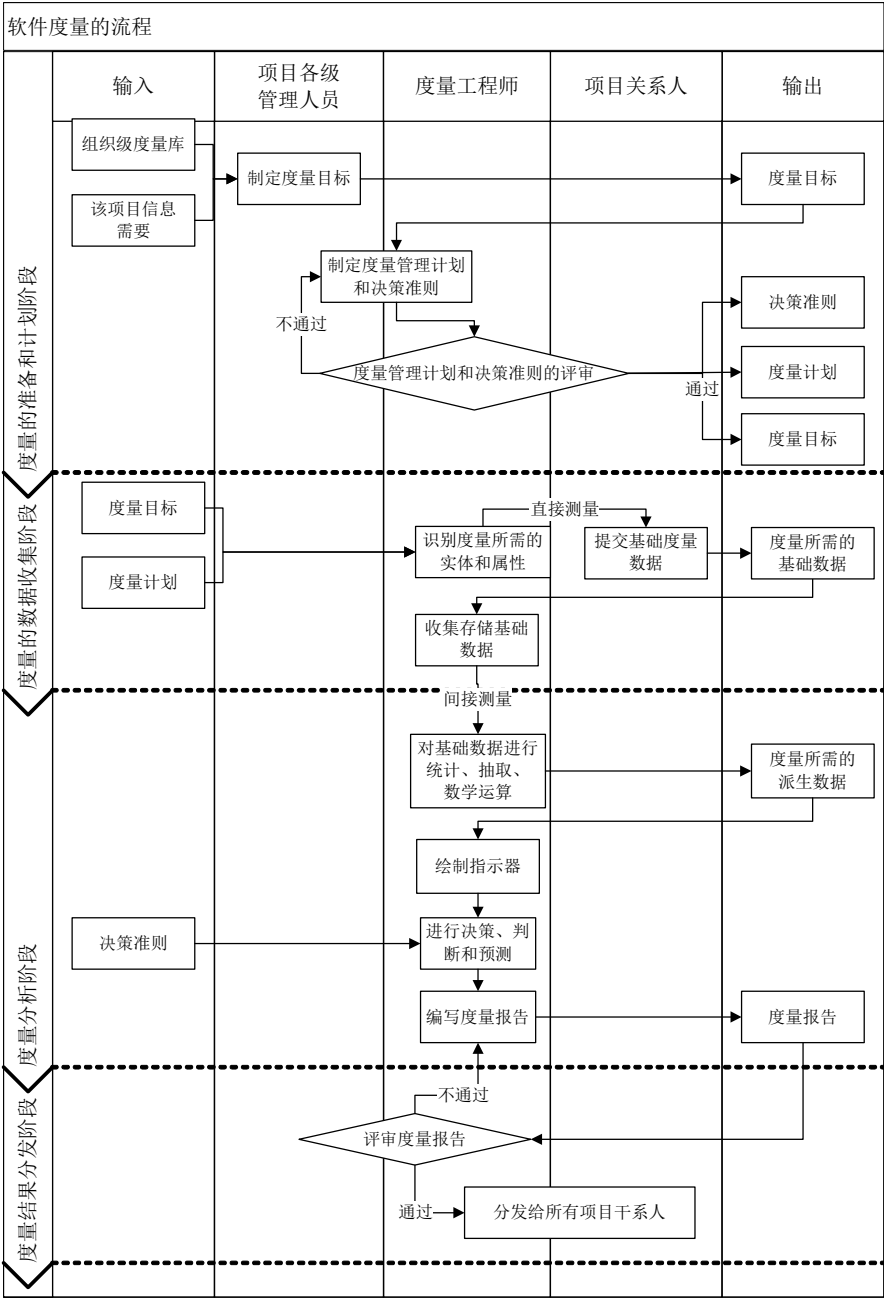


图 7-13 软件度量管理流程

【分析】

项目经理小赵找到了另外一个项目经理老张，希望可以从老张那里得到一些启发。在老张介绍他的项目是如何收集和度量数据时，小赵发现了一个值得借鉴的细节。软件度量应该是客观的、量化的，这个目标是好的，但是很多度量方法中还是存在一定主观的因素。例如在标准功能点估算法中对复杂度进行判断时就离不开主观的因素，而且就算参加同一种估算方法的两个人，由于个人上课的专心程度、理解力、工作经验不同，培训后对本次培训的理解和掌握情况都是不同的，就算是同一个人在不同年龄段对某些事物的看法也是不同的。

因此要想提高软件度量的准确性首先要有一个通用的软件度量的软件或平台来记录和收集度量的基础信息。当某些度量的基础信息较为复杂时，例如软件规模估算所使用的功能点，那么就应该采用多人同时估算的方法进行度量，并记录该度量数据是如何计算得来的。这样就可以通过对多人度量数据的对比发现不一致的地方，然后再对不一致的地方进行讨论，直到达成统一意见为止。

项目经理小赵觉得老张的做法十分可取，他决定在下个项目中进行使用。

7.3.2 从哪里可以收集到度量所需的数据

【案例】

某软件公司教育事业部刚刚接受完 CMMI 的培训和辅导，公司高层领导觉得软件度量所提供的客观洞察力对公司来说十分重要，由于现在公司没有开展相关的度量工作，所以公司要求各个管理人员回去进行思考，每人都要在下周二的管理层会议上进行发言。教育事业部项目经理小蔡找到另外一家软件公司的过程改进负责人老张，希望从他那里可以得到一些启发。

【分析】

老张给小蔡的建议是要想将软件度量真正开展起来首先要得到公司高层领导的支持，因为度量管理工作是要投入一定成本的，这个成本可以分为度量管理工具的成本和执行度量工作时所花费的人力成本。

在得到公司高层的支持后就必须全员开展度量管理工作，起初可以进行试点，但是最终要进行全员推广。度量管理工作主要增加了管理工作的负担和成本，对于一般软件开发和测试人员来说所增加的工作量并不大。因此要对管理人员进行重点的培训，最好可以有专职的软件度量工程师。

要购买或自己研发一套收集度量基础数据的软件或平台，因为软件度量很多时候要对项目计划和项目实际情况进行各种各样的对比，项目计划可以通过很多工具来完成，例如：

Microsoft Project、Word、Excel 等，但是项目的实际数据该依靠什么工具进行收集，这是很多软件公司所没有的。这个工具至少应该包含以下内容：

- ① 可以将项目计划中的任务导入该系统的功能。
- ② 项目经理可以将任务进行分派的功能。
- ③ 项目团队成员在接受分派的任务后，可以针对该任务填写当天工作日志的功能。
- ④ 工作日志要可以填写详细的度量信息，例如：任务的开始时间和结束时间、任务的优先级别、任务的状态、任务完成的百分比、该任务工作成果的具体数据等。
- ⑤ 报表查询和统计的功能。

项目经理小蔡将和老张的谈话逐一进行了记录，而且老张给出的建议正是他所需要的内容，老张也真心希望小蔡所在的软件公司可以将内部研发管理工作做到实处，让软件度量可以发挥它客观洞察力和预测未来的能力。

7.4 小结

软件度量和测量之间有着较大的区别，测量可以算是软件度量的一个子集。在日常的软件研发工作中，并不是所有度量目标都要去做，项目团队可以根据度量的信息需要对度量的整体目标进行裁剪。软件度量也是软件工程的基础环节，是科学地管理软件研发的依据。软件度量的工作覆盖了软件生命周期模型的各个环节和产出物，它为软件企业的各级管理人员提供了客观的洞察力。

7.5 思考题

1. 度量和测量有什么区别？
2. 什么是基础数据，什么是派生数据？
3. 什么是软件度量所使用的指示器？
4. 常用的指示器有哪些？

8

第 8 章

软件质量管理的预警措施——风险管理

风险管理是一门新兴的管理学科，在软件项目管理和软件质量管理体系中都是不可缺少的一个重要组成部分。

风险管理最早于 1930 年起源于美国，在 20 世纪 30 年代一场世界性的经济危机正影响着全球，也正是这场经济危机导致了第二次世界大战的爆发，在这场经济危机中，当时美国有大约 40% 银行和企业破产。为了应对这场经济危机，保险是风险管理中的一种预防措施，许多知名企业都纷纷设立了保险管理部门，负责企业的各种保险项目。但当时应对风险的手段主要还是依赖保险。

20 世纪 40 年代左右，美国的知名企业开始研究科学的方法来对风险进行管理，并逐渐积累这个过程上的经验和教训。终于在 20 世纪 50 年代，一个新兴的管理学科“风险管理”诞生了，也正是从那时起，“风险管理”一词广泛运用在质量管理、项目管理等众多领域。

20 世纪 70 年代，风险管理风靡全球，法国、英国、德国等众多欧洲国家也都是从那时起开始对风险管理进行更加深入的研究。

1983 年在美国召开的风险和保险管理协会年会上，世界各地的风险管理学者和专家共聚纽约，经过共同讨论和协商，著名的“101 条风险管理准则”诞生了，它标志着风险管理的发展已经进入了一个新的时代，是风险管理史上一个重要的里程碑。1986 年，为了扩

大国际领域的风险管理交流，由欧洲 11 个国家共同发起并成立了“欧洲风险研究会”。

在 20 世纪 80 年代，中国也开始了风险管理的研究工作，但当时中国绝大多数企业对风险管理的意识比较薄弱，也没有建立风险管理的组织或机构。随着改革开放的深入，很多外资企业在中国设立了分支机构，对外的交流也更加方便，一些新的管理理念逐渐被大家认可。特别是当今的中国，风险管理的意识和手段无处不在，小到每个老百姓的“五险一金”，大到国家处理各种自然灾害的“应急响应”，风险管理已经深入到生活的方方面面。

8.1 软件风险管理的概述

风险管理在软件研发和质量管理体系中有着非常重要的作用，风险的起因来自于软件项目和产品研发过程中的不确定性。软件研发的过程是将人头脑中主观的最佳实践转换为客观的代码行，需求的变更、理解的错误等诸多情况就是这种不确定性的表现，而这种不确定性就很可能给整个软件产品的质量带来各种各样的隐患。

要想提高软件产品的质量，将软件质量管理的工作进行深化，软件研发和质量管理人员就必须事先制订好软件的风险管理计划，并且在软件的整个生命周期中对风险进行跟踪和不断的识别。

软件风险管理是指对项目的风险进行规划、识别、分析、应对和监控的全部过程。风险管理的目的是尽量扩大对项目有利因素的可能性和影响，尽量缩小对项目不利因素的可能性和影响。因为风险总是事出有因的，风险一旦发生，必将会给项目带来有利或不利的后果。对于积极一面的风险，大家可以将它理解为是一种机会。例如：某 OA 产品按项目计划对某部门经理进行需求调研，但半个月前公司就安排该经理那天要去外地出差，为了完成本次调研的工作，客户临时安排了一个该部门的员工接受访谈。这就是一个风险，该风险的起因是“在项目计划中对项目关系人分析时，没有考虑该重要关系人的资源日历”，当这个风险发生后必然会产生相应的结果，而且可能不止一个结果：

- ① 本次调研的结果不准确。
- ② 需求调研阶段里程碑可能被推迟。

由此可见，风险管理不仅是对项目不利因素的管理，而且也是对项目有利因素的发扬光大，因此风险管理是个中性词。这里提倡的是“居安思危”的意识，希望广大软件项目和质量管理人员不要一提到风险就有种紧张的感觉。

8.1.1 风险的类型

风险的类型分为以下几种：

① “已知-已知”的风险：这类风险的发生原因、发生的可能性和严重性的程度都是已知的。这种类型的风险可以对它进行识别、分析、制订防范和应对的策略。例如：某项目计划中没有安排系统测试用例的评审，那么这个风险是已知的，该风险的后果也是已知的，就是“测试用例的覆盖率无法判断”。

② “已知-未知”的风险：“已知”是指风险的起因是已知的，“未知”是指风险的发生可能性和严重性大小是未知的。这类风险可以对它进行识别和分析，但由于其可能性和严重性无法预测，因此很多软件项目会在项目计划中制订一些风险储备来对其进行应对。

③ “未知-未知”的风险：该风险的起因和结果都无法确定，这种风险由于全部未知。例如：在“非典”发生之前，没有人知道“非典”是什么，也没有人知道“非典”的严重性有多大，因此项目组也无法对它进行预测。但是当“非典”爆发时，很多项目都只能停工，对项目造成的影响有多大也无法准确判断。像这种“未知-未知”的风险，项目组只能通过公司层面的风险储备进行消化。

④ 经营风险：在项目或企业运作过程中的风险。例如：一家对美软件外包的公司在2007~2008年遇到人民币升值的汇率变化，这对企业的运营来说成本提高了，利润降低了。

⑤ 可保的风险：对那些只会亏损而不会为企业带来盈利的机会，通过转嫁或由第三方共同承担的手段来进行应对，往往此类风险都会购买某种类型的保险。例如：项目周期为8个月而且项目组成员要经常出差，期间无法预知是否会有人发生某些交通意外，因此公司要购买“业主责任险”来对此类风险进行转嫁。

8.1.2 风险的来源

风险是一种因果关系，对风险来源的识别也是非常重要的，软件项目 10 大常见风险如表 8-1 所示。

表 8-1 软件项目 10 大常见风险

风险来源	风险的起因	风险的分类	预防措施	应对措施
客户风险	客户方的关系人不具有代表性，不能提供足够的需求资料或信息	需求类风险	做好关系人的识别工作，与客户方多沟通	及时将情况反馈给客户；让客户在需求文档上签字
	客户对需求没有给出承诺		获取客户的承诺作为一个项目进度计划中的一个任务；提前通知客户方安排需求的评审	与客户方进行沟通，书面告知不承诺的危害；通过高层与客户方领导进行沟通
范围风险	客户方多次对软件的需求进行变更	需求类风险	要求客户对需求文档进行确认	申请变更；与客户方进行沟通，将变更内容在以后版本中实现

续表

风险来源	风险的起因	风险的分类	预防措施	应对措施
进度风险 成本风险	没有进行项目估算，或者项目估算的方法不合理	项目计划类风险	在项目计划中预先定义项目估算的方法和流程，并进行相关培训	多人分别进行估算，对估算结果差异大的地方进行共同讨论
	项目进度安排过于紧张		增加估算力度，运用合理的估算方法	申请变更；增加资源
	进度计划中遗忘了例如项目周会等重复性的支持类活动		增强项目计划评审的合理性；对项目经理进行培训	申请变更
内部风险	关键人员的离职	团队建设类型风险	多与团队成员进行沟通，提早发现隐患；提前培养后备人员	挽留关键人员直至项目结束；尽快补充相应人员
	开发人员没有类似产品的开发经验；人员搭配不合理		在立项时申请有能力的人员；加强相关人员技术培训	申请人员变更；加强技术培训
	需求文档不能够准确表达客户的需求；不能够作为测试人员的测试依据	文档类型风险	修改组织级需求文档的模板，使之具有可操作性；让软件测试人员参加需求文档的评审	对需求文档进行变更；对测试人员进行业务培训
	项目中采用尚未经过实践的新技术；	技术类风险	进行技术可行性方案的讨论；使用成熟的技术进行替代	采用螺旋式开发模型；申请掌握该技术人员；项目外包
质量风险	项目组没有掌握白盒测试的方法和技巧	技术类风险	增加开发人员培训	申请掌握白盒测试的人员；开展白盒测试培训
外部风险	汇率变化	市场类风险	关注日常汇率的变化；减少固定总价合同的签署数量	降低研发成本

- ① 范围风险，例如：项目范围变更等相关风险。
- ② 质量风险，例如：没有做单元测试将会导致代码质量低下。
- ③ 进度风险，例如：某任务没有完成将会导致需求阶段里程碑延后。
- ④ 成本风险，例如：项目组只能通过加班来确保需求阶段里程碑按期完成，这将导致项目成本的增加。
- ⑤ 外部风险，例如：人民币对美元汇率的变化将导致软件外包公司成本的变化。
- ⑥ 内部风险，例如：某个系统分析员的离职将会导致项目延期。
- ⑦ 外部不可预知的风险，例如：政府某些政策的改变会给项目带来有利或不利的影响。
- ⑧ 假设的事件或前提，例如：软件项目管理人员在制订各种计划时，计划的模板中都会有“假设”这个章节，在这个章节内的所有内容都将是项目风险的组成部分。
- ⑨ 法律风险，例如：某软件还没有购买相应数量的 license，将会导致知识产权的纠纷。
- ⑩ 市场风险，例如：某外包项目在汇率变化中存在风险。

8.1.3 风险的应对策略

风险发生的结果有好有坏，因此应对积极风险和消极风险的措施也不相同。

积极的风险其实可以理解为对项目的一种机会，获取机会的做法正好与应对消极风险的做法相反。其目的是使获得机会的可能性尽量扩大，产生的影响尽量扩大，通常可以采用以下 3 种应对策略：

- 支持

支持是指提供更多的资源和支持，使项目能够获得相应的机会。

- 共享

为了获得某些机会单靠一己之力往往有些力不从心，将机会共享出来与最有可能把握机会、实现项目利益最大化的第三方合作，会是最佳选择。

- 提高

提高的策略是指寻找机会发生的驱动因素，最大程度地扩大机会发生的概率。在日常生活中自我学习、继续深造的目的就是为了使机会发生的概率增大。

在应对消极风险对项目目标的威胁时可以采取以下 3 种应对策略：

- 回避

回避是指通过调整原来的计划或调整项目范围，从而排除风险发生的条件，以保护项目目标不受风险的影响。例如：某公司的移动增值业务项目，由于硬件供应商交付的延迟可能导致项目出现风险。这种情况的应对措施可以采取回避的方法，当此风险发生时就可以通过变更项目计划来进行应对。

- 转嫁

转嫁是设法将风险的影响和相关连带责任转嫁给第三方，这是风险应对的一种策略，不是推卸责任的做法。通常的做法可以为项目购买某些保险来进行应对。

- 降低

降低是指降低风险发生的可能性和严重性。提前采取措施使其影响降到最低，这比亡羊补牢要好得多。在项目风险列表中针对每条风险的预防措施就是降低风险发生可能性和严重性的最好方式。

8.2 软件风险管理流程及最佳实践

不管项目采用何种开发模型，软件项目的风险管理均贯彻项目的整个生命周期。如图 8-1 所示，在项目启动之初，对于一切都是未知的，因此各种风险发生的可能性最大，同样由于是项目刚刚启动，还没有什么投入，此时如果发生风险，那么其影响也是最小的。当

项目逐步开展接近完工时，特别是在系统测试和用户验收阶段，此时项目的各方面情况趋于明朗，项目的不确定性也越来越小，但是如果客户要在此时进行变更，那么风险的影响将是最大的。由此可见，风险管理对软件项目质量的影响有多大，只有尽早对风险进行管理，软件产品的质量才会提高。

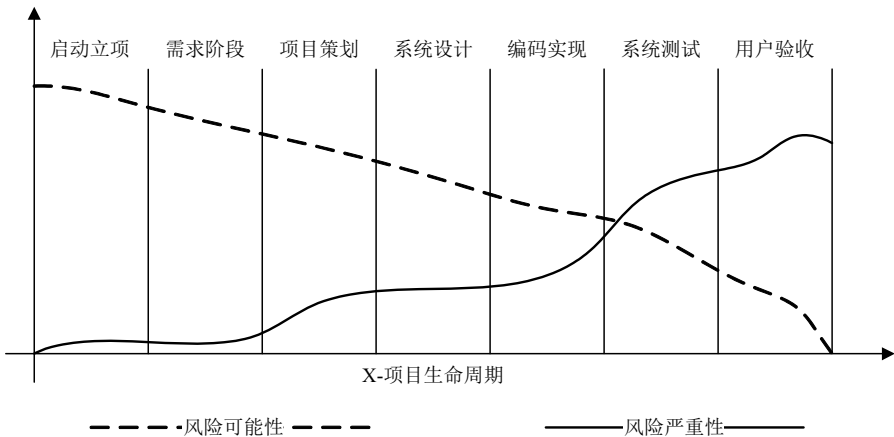


图 8-1 风险在项目生命周期中的体现

软件风险管理准备阶段如图 8-2 所示，可以分为 3 个大的步骤：风险的策划阶段、风险的识别阶段、风险的分析应对阶段。

风险监控过程是伴随软件整个生命周期的，软件监控的过程其实就是对风险再次识别、再次分析、再次策划的过程。在软件开发生命周期的每个里程碑或者发生重大变更时，都要重新做一遍风险管理准备阶段的工作，这就是对风险监控的过程。

8.2.1 建立组织级风险库

一个项目中发生的部分风险很可能与其他项目中的部分风险雷同，当启动一个新的项目时，如果可以参考一下历史项目的风险数据，那将对该项目的管理和质量有非常大的帮助，因此这里推荐的第一个风险管理的最佳实践就是建立组织级的风险库。组织级风险库的操作流程如图 8-3 所示。

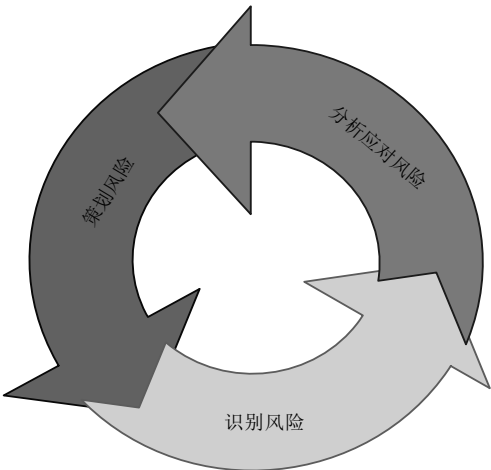


图 8-2 风险管理准备阶段 3 个步骤

组织级风险库中记录的内容均是每个项目结束后提交上来的风险数据，如表 8-2 所示。这些风险的数据包括了每个项目中风险的发生频率、风险的来源和类别、风险的起因、风险发生的可能性和严重性、阈值，以及风险的预防和应对措施等信息。

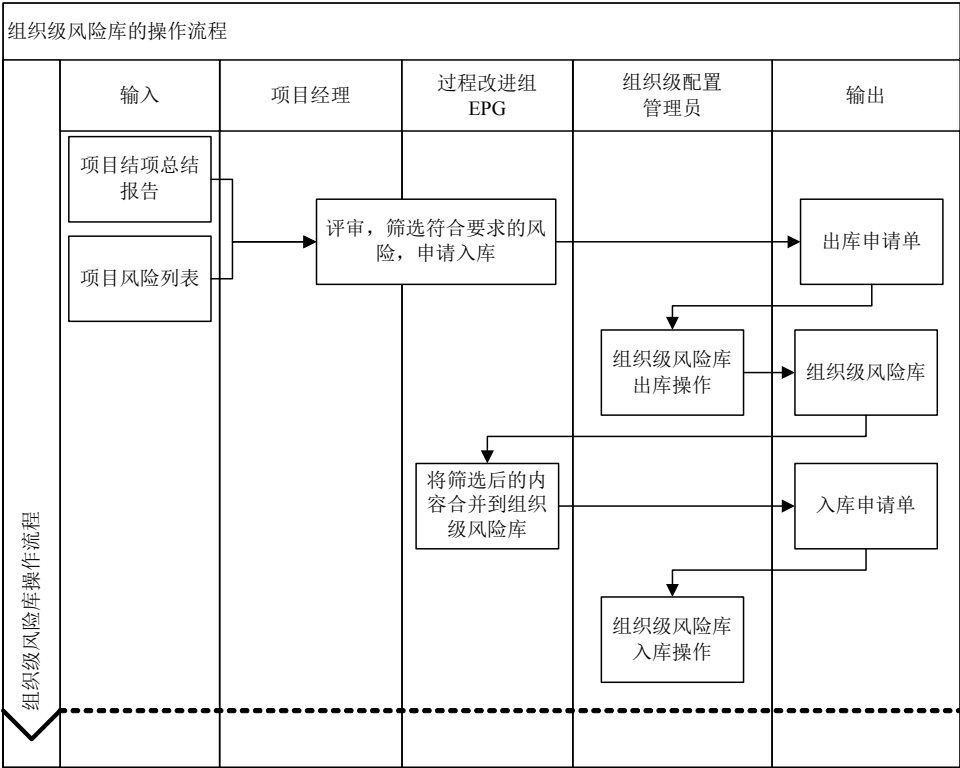


图 8-3 组织级风险库操作流程

表 8-2 组织级风险库模板

风险来源	风险分类	风险起因	风险的可能性	风险的严重性	风险值	阈值	风险预防措施	风险应对措施	风险发生次数

续表

风险来源	风险分类	风险起因	风险的可能性	风险的严重性	风险值	阈值	风险预防措施	风险应对措施	风险发生次数

8.2.2 识别项目风险、定义风险的属性

在软件研发过程中除了项目策划阶段要识别项目的风险外，在项目的里程碑处、重大变更发生时都需要重新识别项目的风险。因为项目是动态变化的，项目生命周期的每个阶段所关注的重点也不尽相同，所以要定期判断是否有新的风险发生，已发生风险的应对措施是否奏效，是否还有残留的风险。识别项目风险的方法有很多，下面我们详细介绍常用的几种。

1. 头脑风暴法

头脑风暴法可以分为直接头脑风暴法和质疑头脑风暴法。直接头脑风暴法是在项目团队集体决策时尽可能地激发创造性，产生尽可能多的设想和方法。质疑头脑风暴法是对某人提出的设想或方案逐一进行质疑，分析其实现的可行性。在风险识别的过程中建议采用直接头脑风暴法，目的是为了发现项目更多的风险。但在进行头脑风暴时也要避免“群体思维或横向思维”的情况出现，项目经理要善于控制讨论的主题，让其始终围绕识别风险进行讨论，并且将识别出来的风险逐一记录在项目风险列表中。

2. 德尔菲法

“德尔菲”这一名称起源于古希腊有关太阳神阿波罗的神话，传说中阿波罗具有预见未来的能力。因此这种群体决策方法被命名为德尔菲法。1946 年兰德公司首次使用这种方法，后来该方法被迅速广泛采用。德尔菲法采用匿名方式征求群体的建议，即参与讨论的项目团队成员之间没有互相交流，避免头脑风暴法中可能发生的横向思维。经过多轮次的问卷调查，反复总结反馈回来的建议，最终形成讨论的决议。在风险识别过程中，德尔菲法可以通过以下步骤进行：

STEP 01 确定参与风险识别人员的名单，一般不超过 20 人。

STEP 02 向所有人员提出本次所要讨论的重点是识别项目的风险，并附上项目的背景材料，同时询问个人还需要什么材料，然后由参与讨论的人员做出书面答复。

STEP 03 参与讨论的人员根据他们所收到的材料，提出自己识别出来的项目风险，并说明自己是怎样利用这些材料发现项目风险。

STEP 04 将参与讨论人员第一次反馈的项目风险进行汇总、对比，填写风险列表，再分发给参与讨论的每个人，让他们比较自己同他人的不同意见，修改自己的意见和判断。

STEP 05 将所有人员识别的风险再次收集、汇总、对比，并再次分发给参与讨论的每个人。逐轮收集意见并反馈信息是德尔菲法的主要环节，此过程一般要经过三到四轮，直到每一个人都不再改变自己的观点为止。

STEP 06 形成最终的项目风险列表。

3. 参考历史资料

“历史总是惊人的相似”，在软件项目中发生的风险有相当一部分是雷同的，因此参考组织级风险库中的历史数据对项目识别风险是非常有帮助的。

4. 其他的风险识别方法

除了以上常用的风险识别方法外，还有以下方法可以使用：

① **SWOT 分析法**：Strength（优势）、Weakness（弱点）、Opportunity（机会）、Threat（威胁）。

② **鱼骨图法**：鱼骨图是由日本管理大师石川馨先生所发明出来的，故又称为石川图。鱼骨图是一种发现问题“根本原因”的方法，它也可以称为“因果图”。

③ **检查表法**：检查表可以根据历史资料进行编制。检查表法的好处是风险识别过程迅速简便，其缺点是检查表中的检查项不可以包罗万象。

项目团队可以根据实际情况选择适当的风险识别方法，如图 8-4 所示，在风险识别的过程中，风险管理计划和项目范围说明书或客户的原始需求是风险识别的重要输入内容。风险识别是个群体决策的过程，也可以邀请项目团队之外的专家一同进行，这样准确性更高。最后当大家达成一致意见后，风险识别的结果要填写到项目风险列表中进行保存。

8.2.3 分析风险并对风险进行排序

风险识别完成以后，就要对风险的各项属性进行定义，常用的风险属性如下：

- ① 风险发生的可能性
- ② 风险影响的严重性
- ③ 风险值 = 风险发生的可能性 × 风险的严重性
- ④ 阈值（临界值）。例如：当风险值大于阈值时，将执行风险的应对措施
- ⑤ 风险的预防措施，执行此措施可以降低风险发生的可能性
- ⑥ 风险预防措施责任人

- ⑦ 风险的应对措施，执行此措施可以将风险的严重性降为最小
- ⑧ 风险应对措施责任人

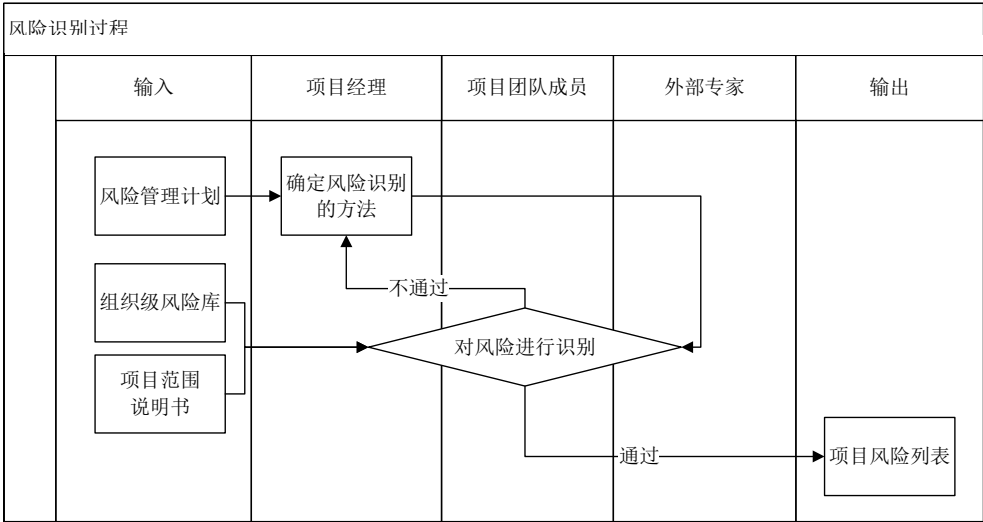


图 8-4 风险识别的过程

项目团队同样可以通过头脑风暴法、德尔菲法或参考历史资料的方法对识别出来的风险进行属性定义。风险预防和应对措施的责任人可能是同一个人，也可能不是同一个人，这要视具体情况而定，但责任到人是必须遵循的原则。阈值、风险的预防措施和应对措施之间的关系如图 8-5 所示。

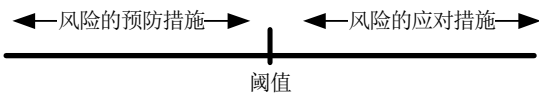


图 8-5 阈值、风险的预防措施和应对措施之间的关系

风险本身就体现了一种因果的关系，很多风险导致的后果是相似的，因此可以将这些风险进行合并。项目团队可以通过鱼骨图法来对风险进行合并，并对风险的预防与应对措施进行识别。使用鱼骨图法对风险进行合并，这里合并的是风险的后果，而不是风险的起因。项目团队一定要注意风险的起因是万万不能合并的，因为风险监控的对象主要是风险的起因。这里之所以要合并风险的后果，是因为它们可以采用相同的措施进行应对。

绘制鱼骨图的步骤说明如下：

- STEP 01** 填写鱼头，画出主骨。
- STEP 02** 画出大骨，填写大的要因。

STEP 03 画出中骨、小骨，填写中、小的要因。

STEP 04 用特殊符号标识重要因素。

制订风险预防措施的最主要目的是降低风险发生的可能性，因此在画鱼骨图时鱼头应该向左，鱼头代表识别出来的风险，鱼刺代表每个预防的措施。如图 8-6 所示，风险的预防措施不一定只有一个，解决方案可以有多种。

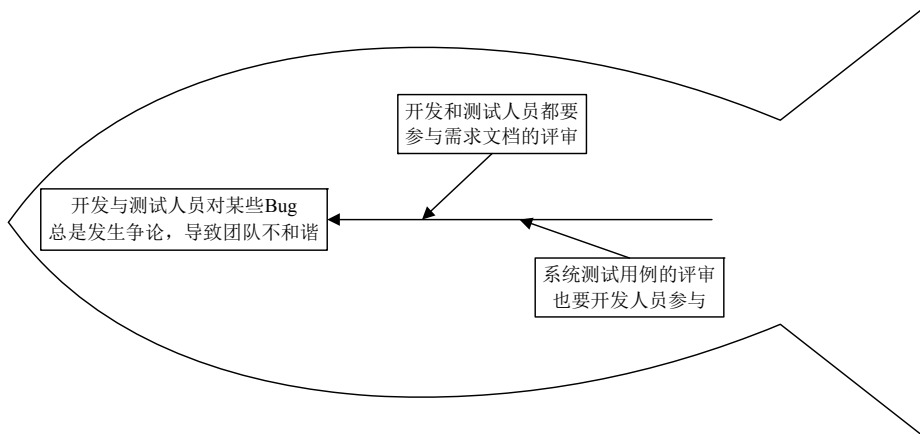


图 8-6 使用鱼骨图法分析风险的预防措施

在制订风险应对措施时，鱼头应该向右，鱼骨上的每根鱼刺均代表一个识别出来的风险，通过对这些风险的分析、合并，可以发现它们产生的影响基本相同，因此也可以采取相同的风险应对措施。如图 8-7 所示，风险的应对措施标记在鱼头处。

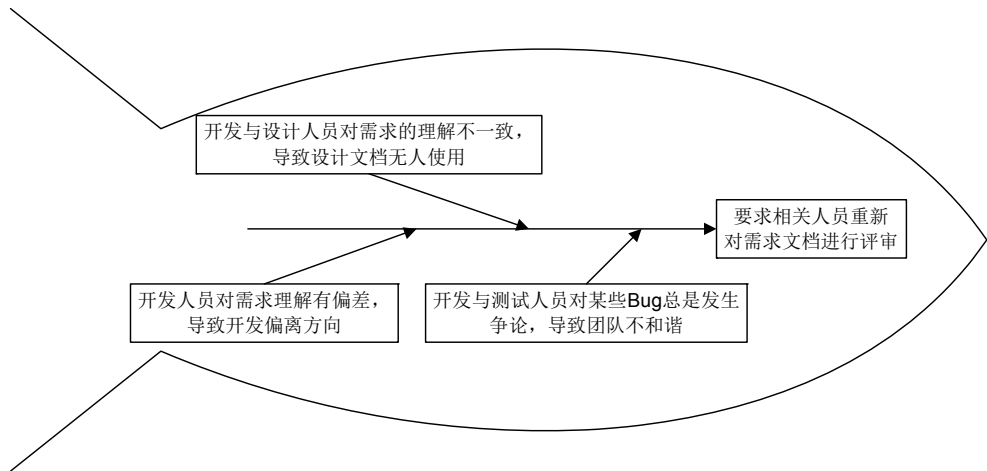


图 8-7 使用鱼骨图法分析风险的应对措施

由于每个软件项目识别出来的风险会有很多，作为项目管理人员如果想让风险的管理工作更有成效，就需要制订风险排序的策略，并对风险进行排序。在对项目风险进行监控时，管理人员只需要重点关注“高优先级”的风险，尽快执行风险的预防措施，尽量避免高优先级风险的发生，当其发生后必须立即执行风险的应对措施；定期跟踪“中等优先级”的风险，当其风险值接近阈值时执行风险的预防措施，风险发生后立即执行风险的应对措施；最后甚至可以忽略或接受“低优先级”的风险，这样就可以根据帕累托的“80/20”理论把握重点，使风险管理的工作更加有效。

风险的两个因素是风险的可能性和风险的严重性， $风险值 = 风险可能性 \times 风险严重性$ ，因此风险值综合反应了风险的大小。如表 8-3 所示，如果将风险的可能性和严重性各设置 5 个级别，那么风险的优先级分类就一目了然了。在项目风险列表中，项目经理可以对应风险矩阵，使用相应的颜色来表示每条风险。

- 高优先级风险：15~25
- 中优先级风险：6~12
- 低优先级风险：1~5

表 8-3 风险优先级矩阵

风险严重性	风险可能性					风险优先级
	很高 5	较高 4	中等 3	较低 2	很低 1	
很高 5	25	20	15	10	5	
较高 4	20	16	12	8	4	
中等 3	15	12	9	6	3	
较低 2	10	8	6	4	2	
很低 1	5	4	3	2	1	

8.2.4 风险的跟踪

项目生命周期不同阶段风险的可能性和严重性也是不同的。例如：在需求调研阶段，客户方不能提供准确参考数据的可能性非常高，但是在系统测试阶段，软件产品已经完成，需求已经明朗，此时该风险发生的可能性会降低。再例如：在需求调研阶段，客户方不能提供准确参考数据的严重性不高，因为软件产品还没有开始研发，项目的投入不大。但是在系统测试阶段，软件产品已经完成，此时如果要对需求进行变更，返工的成本将非常高，此时该风险发生的严重性也会非常大。因此风险是一个动态变化的事物，风险的跟踪也需要定期进行。

风险跟踪所要做的具体工作如下：

- ① 对已经发生的风险，检查其应对措施是否已经执行，执行的效果如何。

② 对已经发生的风险，在执行应对措施后，要检查是否还有残留的风险，是否会引发新的风险。

③ 关闭未发生的风险：有些风险是在生命周期的某些特定阶段才会发生的，一旦这个阶段完成，那么这些风险也要随之关闭。

④ 识别新的风险：通常风险的识别都是在项目计划阶段，但由于项目的不确定性而无法准确判断未来阶段的情况，所以风险在每个阶段都需要进行重新识别。

⑤ 每个阶段风险的可能性和严重性都在变化，因此要定期更新每条风险可能性和严重性的值。

⑥ 定期将风险值与阈值进行对比，及时采取相应的风险预防与应对措施。

⑦ 由于每个阶段风险的可能性和严重性都在变化，从而导致风险值也随之变化。因此要定期更新风险的优先级别，以保证管理人员始终关注重要的风险。

8.3 软件风险管理常见问题及案例分析

风险管理是个复杂的过程，它不但贯穿项目的整个生命周期，而且它还是动态变化的。另外风险管理与项目的范围管理、时间管理、成本管理等都有非常紧密的联系，因此软件研发管理中的很多问题都是由于风险管理没有做好而导致的。

8.3.1 为什么风险识别总不准确

【案例】

某软件公司 OA 项目的项目经理小刘最近有些苦恼，在项目启动时项目团队一起对风险进行了讨论和识别，现在项目已经进展到编码实现阶段，可是在里程碑会议上总是发现项目风险列表中记录的风险不太适用。导致现在小刘开始怀疑软件项目是否要进行风险管理，因为他的项目中风险管理感觉上只是走过场的形式主义，起不到多大作用。那么应该如何将风险管理落到实处，使之切实可行呢？

【分析】

软件过程改进专家老张针对小刘的问题给出以下建议：

风险的识别对整个风险管理起着指导性作用，其准确性的高低是风险管理成败的首要条件。风险管理贯穿项目整个生命周期，而且风险又是动态变化的，因此风险的识别工作也不能只在项目启动阶段做一次，而要定期地进行风险识别工作。

通常进行风险识别时可以按阶段进行识别，先重点识别当前阶段的风险，更新上一阶段的风险状态，然后再识别下一阶段的风险。例如：以瀑布模型为例，在项目立项里程碑

处，先重点识别需求阶段的风险，然后再更新立项阶段风险的状态，最后可以简单识别一下项目计划阶段的风险。这样做的好处是完全符合风险动态变化的特性，识别出来的风险更有针对性，而且每次风险识别的工作量也不大。

项目经理小刘听完老张的建议后终于明白要想让风险管理切实可行，风险识别的工作必须定期执行，而且分阶段进行风险识别的方法也具有可操作性。

8.3.2 为什么项目计划总是不准确

【案例】

某软件公司项目总监张先生发现财务系统项目组在每个里程碑前都会出现大量加班的情况，张先生感觉这种周期性的加班应该是一个项目管理的问题，张总监找来该项目负责人小徐进行谈话。小徐也觉得比较奇怪，明明项目计划都经过了公司的评审，而且在开发过程中都有严格按照公司流程进行，为什么项目计划就是不准确呢？

【风险】

经过了解，小徐是个忧患意识非常强的人，他很重视项目的风险，一旦风险接近或达到阈值，他就会安排人员执行风险的预防或应对措施。这些措施都是实实在在需要人来完成的工作任务，但是这些任务都没有体现在项目计划中，执行了这些风险的措施而导致计划内的任务没有完成，这就是项目组为什么在里程碑前大量加班的原因。

针对这个情况项目总监首先表扬了小徐，因为他有风险管理的意识，而且他也确保了里程碑的按期完成。最后项目总监给了小徐一个建议，就是在制订项目计划时要预留一些风险储备的时间，这样项目计划就会更加准确，更具有指导性。

8.3.3 如何在项目进行风险跟踪

【案例】

某软件公司项目经理小高在接受完 PMP 培训后，觉得风险管理对于软件研发来说非常重要，他对风险的计划、识别、定性分析、定量分析、制订风险应对规划的环节都非常清楚，但就是对如何进行风险跟踪还一时搞不太懂。到底风险跟踪应该何时来做，都做哪些内容呢，又是通过哪些文档来实现呢？

【分析】

软件过程改进专家老张向项目经理小高提出以下解决方案：

根据项目的规模和难度以周或月为单位对项目风险进行定期监控，在监控时要根据风险列表中记录的风险进行回顾，关闭已经结束或未发生的风险；修改列表中其他风险可能性和

严重性的数值；识别是否有新的风险已经或即将发生；验证风险预防或应对措施的有效性。

如果项目组选择以周为单位对风险进行跟踪，那么可以在周报中增加风险管理的内容，其格式与风险列表中的内容一致。风险识别的工作可以在周会上进行，并将识别出来的结果记录在周报里。周报中记录的风险包括新识别出来的风险、上周遗留下来的风险和需要继续跟踪的风险，在周会上就可以在周报中更新风险发生的状态，以及记录风险状态发生变化的时间。在每周进行风险识别时，对于已经结束或未发生的风险，就将其更新回项目风险列表里，当项目结束后，就将最后一次项目周报里的所有风险更新回项目风险列表中。

8.4 小结

风险体现了一种因果的关系，风险的发生必将对项目产生有利或不利的影响，风险在项目整个生命周期中又是不断变化的。风险的识别对风险管理来说有着指导性的作用，风险的跟踪是将风险管理落到实处。只有将风险管理工作做好，软件项目管理才能更加顺畅，软件产品的质量才能得到保障。

8.5 思考题

1. 风险的可能性和严重性在项目生命周期中是如何变化的？
2. 对于“已知-未知”类型的风险应该采取什么样的应对措施？
3. 风险识别的方法有哪些？
4. 常用的风险属性有哪些？

9

第 9 章

软件质量管理的统筹规划—— 项目集成管理

试着设想一下，如果你是一名项目经理，你负责的不是一个小规模的项目，那么在项目启动后，你所要做的第一件事是什么？

如果你的回答是开展需求调研，那么你就错了，因为你还没有选择项目应该采用哪种生命周期模型。

如果你的回答是制订项目计划，那么你又错了，因为你还没有对项目进行整体规划。

对于中等或大型项目来说，项目管理人员所做的第一件事是要对项目的整体运作进行策划，这个过程被称为项目集成管理，或者项目整体管理、项目综合管理，它是一种全局性的、综合性的策划工作。如图 9-1 所示，在整个项目集成管理过程中要将软件工程的不同知识领域关联起来，还要将项目与组织的行为规范关联起来。

项目集成管理的做法是依据组织标准过程所裁剪而成的、集成的、已定义的过程来建立和管理项目，并对相关项目关系人进行管理。由项目集成管理的做法可以引申出一个问题，为什么在项目启动时要利用裁剪组织的标准过程来对项目的整体进行策划呢？

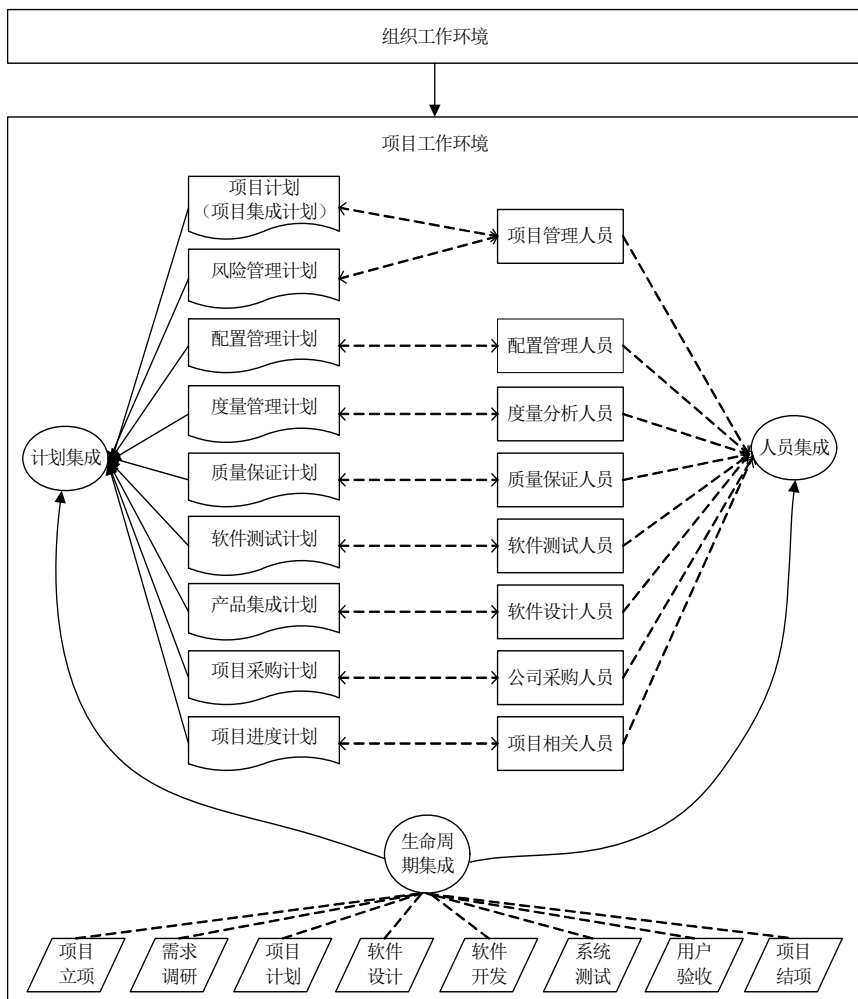


图 9-1 项目集成管理示意图

首先组织财富库中保存了大量历史数据可以供项目参考和估算，其次任何一个软件生命周期模型都是软件工程的方法论，例如：迭代式、瀑布式、XP 极限式、原型式等。在现实工作中没有一个项目可以完全照搬任何一种软件生命周期模型来对项目进行开发。

因此在软件项目启动之初就应根据项目的周期、合同金额等属性判断项目的级别，以便在组织财富库中寻找类似级别的历史项目进行参考。在项目集成管理中要先对生命周期模型进行裁剪，例如有些外包项目不需要进行需求调研，那么就没有“需求调研”的阶段；

例如某些项目的规模不大，因此“需求调研”阶段和“项目计划”阶段可以合并。当软件生命周期模型确定后就可以知道项目会有哪些工作产品，为了对项目工作产品进行更好的管理，就需要制订相关的管理计划，其中项目计划也称为项目集成计划，它是所有计划的核心。项目是软件公司运作的一种方式，软件项目永远也不会超越组织的范畴，软件项目也要依据组织的工作环境来裁剪、定义项目的工作环境，例如：公司严格规定不允许在办公场所吸烟，那么项目也应该遵循组织的此项规定。以上就是对项目进行整体策划和管理的过程，其总体流程如图 9-2 所示。

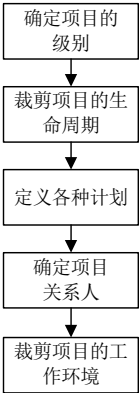


图 9-2 项目集成管理总体流程

软件工程是一个系统工程，需要对软件工程的各个方面制订管理计划，例如度量管理计划、配置管理计划、测试管理计划等。但在这方方面面的计划中都会定义所需的资源、工作的任务、任务计划完成的时间等，如果要对计划进行变更那就非常麻烦，而且不利于对项目的全局进行分析。因此需要一份计划来将这方方面面的计划进行整合，以该计划为主，其他计划都将作为其从属计划，这样的计划被称为项目集成计划，也就是日常工作中所用的项目计划。其实在软件管理的日常工作中大家都在不知不觉地对项目计划进行集成，例如：项目的进度计划中都会包含了配置管理的任务安排、度量工作的任务安排、质量保证工作的任务安排，而没有人会为每个计划单独制订一份进度计划或成本计划。

9.1 项目整体策划的流程及最佳实践

1. 策划项目的过程

以裁剪指南为指导，从组织标准过程裁剪而来的已集成、已定义的、有连贯的生命周期称为项目已定义过程，当项目完成过程定义并通过评审，项目组就要保证对自己已定义的过程无条件执行。

公司内所有项目都是从组织标准过程裁剪而来的，这样的好处是项目间的差异性通常会减少，项目彼此之间就可以更容易分享过程资产、数据和学习心得。

对项目进行“裁剪”的过程也就是对项目整体进行策划的过程，如图 9-3 所示，在此过程中需要过程管理专家或过程改进小组的参与，他们会给项目管理人员建议并辅助他们来完成“裁剪”。不管是“增加”还是“减少”某个生命周期的阶段还是某个流程，或者某些工作产品，裁剪的前提都是不能给项目带来风险。

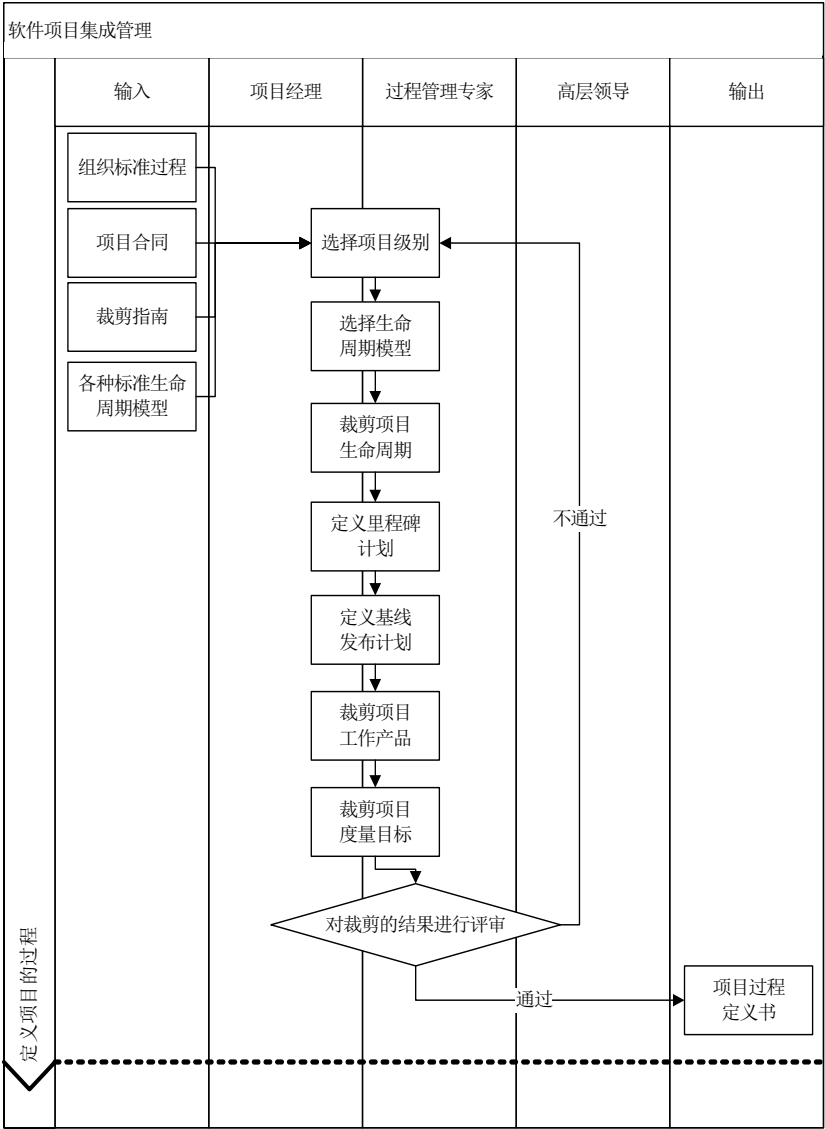


图 9-3 使用裁剪指南定义项目过程

对组织标准过程进行裁剪后生成已定义的项目过程将会被记录到《项目过程定义书》中，项目管理人员还要记录对某些内容进行裁剪的原因。例如：某软件项目是个外包类型的项目，项目金额 160 万，属于 A 类项目，因此选择瀑布式模型作为生命周期。依据裁剪指南对项目基线的指导，裁剪后的项目基线如表 9-1 所示。

表 9-1 基线裁剪的结果及发布计划

软件生命周期模型	阶段	基线名称	发布计划	裁剪原因
瀑布式模型	项目立项阶段	项目立项基线	2009-1-25	
	需求调研阶段	项目需求基线	被裁剪	本外包类项目没有需求的相关工作
	项目计划阶段	项目计划基线	2009-2-21	
	软件设计阶段	软件设计基线	被裁剪	本外包类项目没有软件设计的相关工作
	编码开发阶段	编码开发基线	2009-6-5	
	系统测试阶段	系统测试基线	200-7-29	
	产品验收阶段	产品验收基线	2009-8-12	
	项目结项阶段	项目结项基线	2009-8-31	

2. 选择项目估算用的历史数据

在对项目进行具体计划时首先要对项目进行估算，估算的方法之一就是参考组织的相关历史数据，那么应该参考哪些历史数据呢？这就是项目集成管理对项目整体策划的内容之一。

在项目估算时使用到的历史数据分别来自于以往项目的详细资料以及组织级的度量数据。组织级的度量数据在项目计划时用于对项目的整体进行估算，在对项目整体进行策划时通常会选择组织级度量库中的以下数据，如图 9-4 所示。

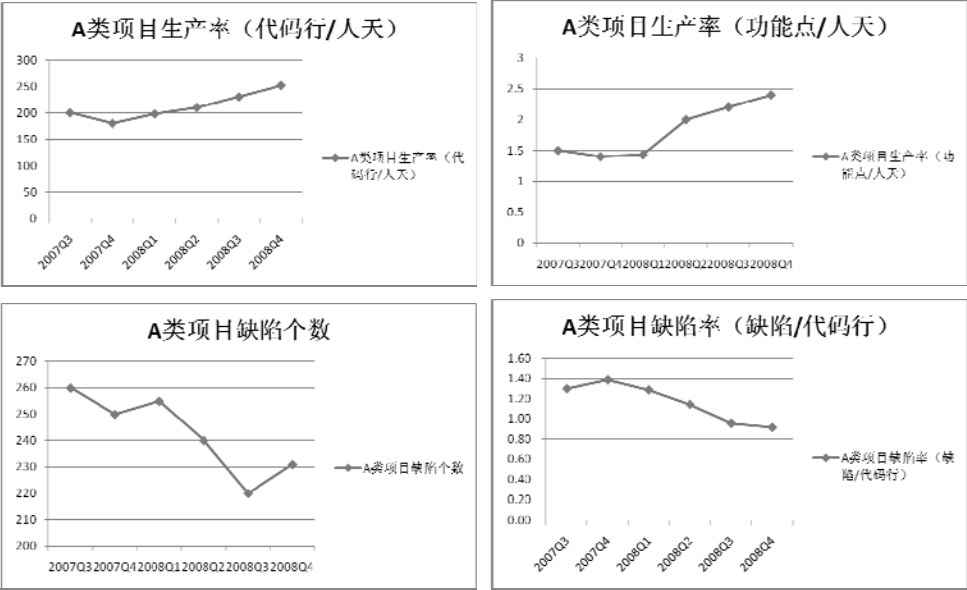


图 9-4 组织级历史数据

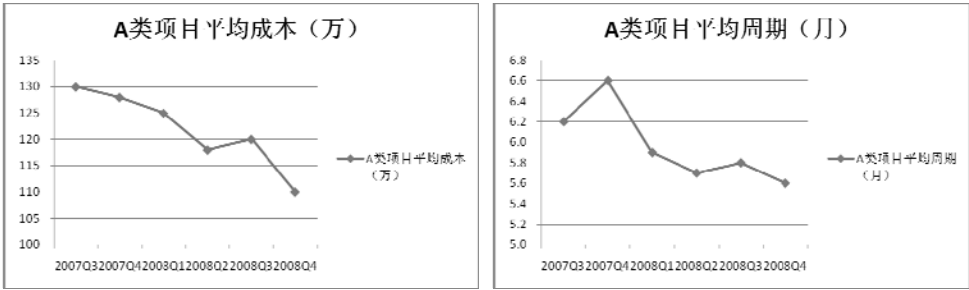


图 9-4 组织级历史数据（续）

在选择以往项目进行估算时要筛选同样级别的项目进行参考，某 A 类瀑布式模型项目的参考数据如表 9-2 所示。

表 9-2 某 A 类项目的参考数据

工作量分布	CRM 项目	OA 项目
项目立项	1.00%	2.00%
需求开发	5.00%	5.13%
计划	11.00%	8.96%
设计	22.00%	19.90%
开发	30.00%	26.30%
测试	15.00%	25.40%
验收	10.00%	9.63%
结项	6.00%	3.00%
支持类工作	CRM 项目	OA 项目
配置管理工作量	4.62%	5.58%
质量保证工作量	4.53%	5.47%
项目度量工作量	4.30%	4.78%
人力资源数	CRM 项目	OA 项目
项目经理	1	1
需求人员	1	1
设计人员	2	2
开发人员	5	5
测试人员	4	3

3. 创建项目工作环境

软件开发过程中有开发用的环境，软件测试过程中有测试用的环境，项目团队的日常

生活和工作同样有相应的工作环境。工作环境一般分为硬件环境和规章制度两个方面，常见的项目硬件工作环境有：电脑、宽敞明朗的办公场地、白板、各种颜色的签字笔、绿色植物等。规章制度也是工作环境的一个部分，只不过它是没有形状的，例如：开发人员的编码规范、公司上下班及节假日安排、图书的借阅制度等。

在组织级标准过程文档中都有工作环境的定义和描述，项目组可以根据组织级的工作环境来定义项目级的工作环境。有些时候项目组觉得组织级的某些规章制度对该项目束缚性特别大，那么就可以通过定义项目级工作环境对其进行修改，这样又不会违反公司的规定，又可以给项目组更多的发挥空间。例如：某项目组在组织级工作环境基础上还需要增加2台空调；组织级原有的上班时间是早上9点，项目组可以根据实际情况将其调整到8点半。当项目级工作环境被审批后，整个项目组就可以按照它开展日常的工作。

4. 集成项目计划

通过对生命周期模型及里程碑、基线和工作产品的裁剪，就可以知道项目组需要制订哪些计划。有些规模较大的项目需要制订《度量管理计划》、《配置管理计划》、《风险管理计划》等，可有些小的项目只需要制订《项目计划》和《项目进度计划》。当项目计划较多时，就需要制订一份主计划来整合其他项目计划的内容，这样便于对计划进行维护和阅读。通常这样的计划被称为项目计划或项目集成计划。

首先根据对项目工作产品的裁剪结果识别项目都需要制订哪些计划，然后分别安排相关人员制订这些计划，最后将这些计划中的成本、进度、资源、职责分配等共性的内容进行提取，统一放在《项目计划》中进行描述，如果发现各个计划出现矛盾或不一致的地方，则需要与相关人员进行讨论，直到达成共识，其流程如图9-5所示。

5. 识别和管理关键关系人

在项目集成管理阶段需要对项目关键关系人进行识别，例如：项目需要制订《配置管理计划》，那么项目组是否需要1名配置管理员呢？如果还需要制订《软件质量保证计划》，那么质量保证人员应该由谁来担任呢？这些都是对项目关键关系人的识别。因此项目管理人员可以根据《项目过程定义书》的内容进行识别，最好能具体到某个人，否则至少也要识别出相关的角色，以便在制订详细项目计划时可以再对其进行细化。

在项目管理中的日历有两种，一种是项目中使用的进度日历，一种是硬件或人力资源可用时间的资源日历。在对关键关系人进行识别时，特别需要注意关键关系人的资源日历是否与项目的进度日历相冲突。例如：项目组在2月份将开始界面设计的工作，但是UI工程师在2月份却在另外一个项目中工作，那么就会出现资源日历与项目进度日历冲突的情况。如果发生冲突，项目管理人员就需要与相关人员进行沟通和协调，必要时可以向高层领导寻求支持。

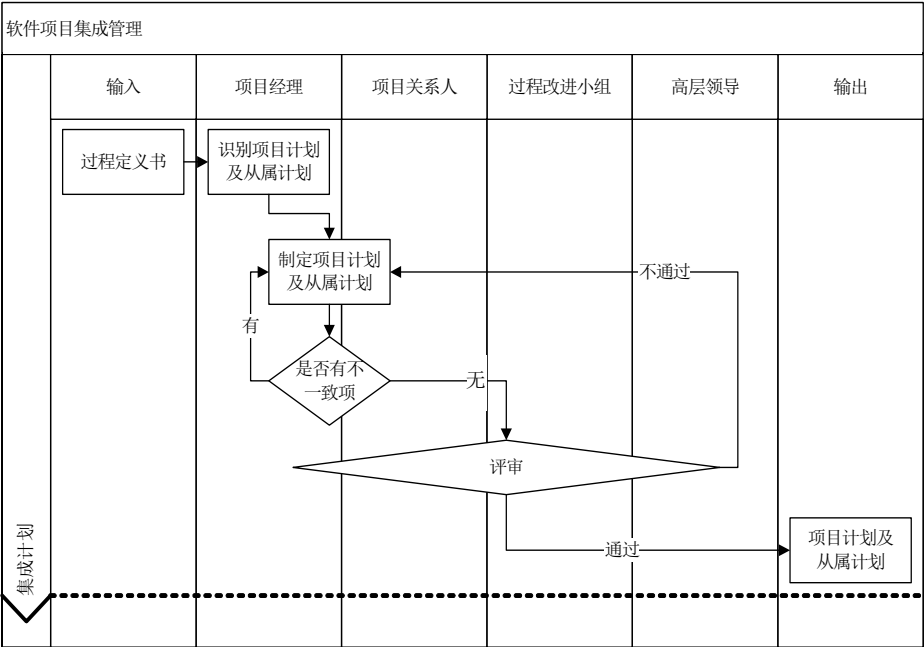


图 9-5 集成计划

6. 识别和管理项目依存关系

在软件项目中存在着很多重要的依存关系，如果不能尽早对其进行识别，那么将会导致项目进度计划的变更，还会给项目带来非常大的风险。因此要在项目整体策划时就开始对它进行识别，常见的依存关系有以下 3 种：

- 强制性依赖：是指所从事的工作性质中固有的依存关系，又称为硬逻辑。例如：某软件项目采用 XP 极限式开发，那么就必须先写测试用例才能写代码，这个顺序不能颠倒。
- 可选的依赖：是项目管理团队所规定的依赖关系，通常是一些经验或最佳实践，也称为优先选用的逻辑关系或软逻辑。例如：当《软件需求说明书》评审通过后，软件测试人员就可以开始编写测试用例，而不用等到编码完成后才开始。
- 外部依赖：是指项目活动与项目外部活动之间的依赖关系。例如：项目需要采购一台服务器，但是代理商却迟迟没有交货。

当项目的重要依存关系都识别出来后，项目管理人员就要想办法获得相关人员对它的承诺，然后在项目监控的过程中对相关人员的承诺进行跟踪，以保证项目可以顺利进行。

7. 向组织财富库贡献项目的资产

软件项目在进行估算时需要参考其他项目的历史数据以及组织级的度量数据，这些数据都来自于各个项目结束后向组织财富库作出的贡献。如图 9-6 所示，通常项目结束后会提交以下资产给组织财富库：

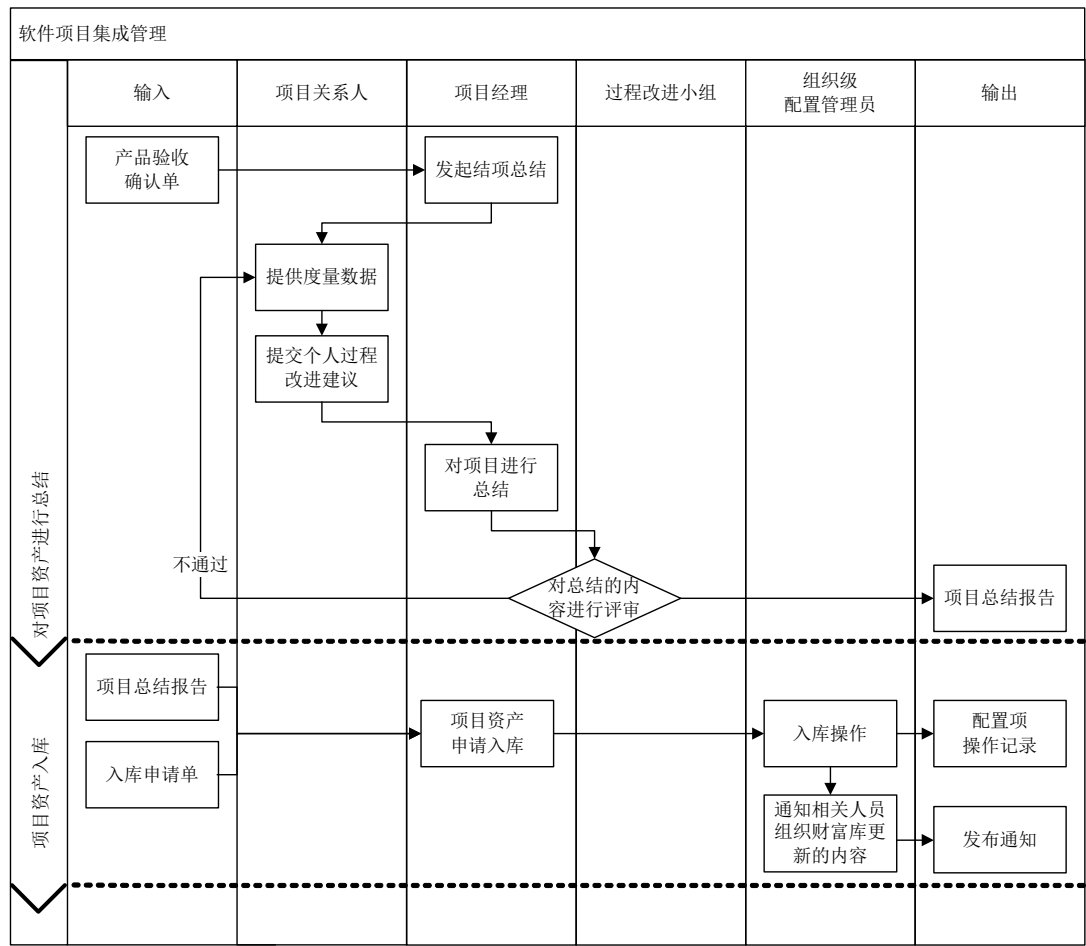


图 9-6 向组织财富库贡献项目资产

- 项目计划及从属计划
- 项目度量报告
- 项目的需求文档
- 可执行的程序及各版本的代码

- 项目培训的记录
- 各种检查表
- 缺陷数据库
- 学习心得体会
- 经验教训
- 优秀案例及最佳实践
- 过程改进建议

9.2 XP 极限式开发模型与 CMMI 的比较

在对项目进行整体策划时可以说所有内容都是围绕软件生命周期模型展开的，以下与大家一起探讨一个有关软件生命周期模型的话题。

【案例】

很多软件技术人员都认为 XP 极限式开发是他们最欣赏的开发模型，他们中的绝大多数都比较抵触 CMMI，认为 CMMI 是一种臃肿的、文档烦琐的、效率比 XP 极限式开发差的开发模式。到底 XP 和 CMMI 的矛盾根源是什么呢？

【分析】

首先 CMMI 讲的是软件工程，而 XP 极限式开发讲的是软件工程中的一个部分：软件生命周期模型，例如瀑布式、原型式、迭代式开发。如果 XP 极限式开发非要找个可以与之对比的对象，那么应该是与各种软件生命周期模型进行对比，而不是代表软件工程的 CMMI。

为什么说 XP 极限式开发不应该与 CMMI 进行对比呢？XP 极限式开发的项目离不开“配置管理”、“度量管理”、“日常的各种培训”等软件工程中的一些内容，但是 XP 极限式开发理论中却没有提及这个部分，这是为什么呢？原因很简单，因为 XP 极限式开发的定位就是生命周期模型，而不是软件工程，而 CMMI 却是软件工程的集中体现。

XP 极限式开发讲的是拥抱变化，这是非常好的理念，但先进的软件工程也提出“变化是永恒的，不变是短暂的”。由此可见出发点都是相同的，而 XP 极限式开发的最佳实践中却没有提到如何避免变化的发生，这就有些范围蔓延的嫌疑。如果限制项目的成本和时间，那么 XP 项目的项目经理是否还会拥抱所有的变化呢？

CMMI 的管理过程并不是那么烦琐和庞大，CMMI 最佳之处就是它具有“项目集成管理”这个部分的内容，CMMI 强调的是对不同项目进行“量体裁衣”，而不能使用标准的

过程来强加、硬套在任何一个项目上。但很多没有研究过 CMMI 的人却不了解这一点，这也就是很多人对 CMMI 的最大误解。采用 XP 极限式开发的项目经理回想一下你的项目是否借鉴了 XP 的所有最佳实践，是不是也对 XP 的一些内容进行了裁剪。正是有了 CMMI 提出的“裁剪”概念，软件工程才有了生命力。

9.3 小结

项目集成管理是软件项目管理工作的第一步，是对软件项目管理工作的“量体裁衣”的过程，只有为软件项目确定一个适合的管理模式，项目才会有一个良好的发展基础，有了这个基础，软件产品的质量才会有保障。

9.4 思考题

1. “裁剪”与“裁减”有什么不同？
2. 项目的依存关系有几种，分别是什么？

10

第 10 章

软件质量管理的策划——项目计划

古人云：“凡事预则立，不预则废”、“谋定而动”，这里的“预”和“谋”指的就是做计划，做任何事情之前都需要深入的思考，制订一份适合的、可行的、周密的计划。

软件质量管理的定律之一“变化是永恒的、不变是短暂的”，由于软件项目的特点，在软件研发过程中各种变化是经常发生的，如果不进行很好的变更管理，就会导致项目计划的指导性不强，这也就是很多项目管理人员不愿意做计划的根本原因。以爱因斯坦“相对论”的视角来分析世界万物，所有的事情都会有两面性，“计划”与“变化”就是相对存在的两种相对的事物，如果没有计划，那么也没有变化。

制订一份合理的计划后就需要执行计划、监控计划，并对计划的有效性进行管理，虽然这个过程的工作量是很大的，但这也正是项目管理人员所需要完成的本职工作。有的项目管理人员会说：“我的沟通能力强；我的条理性好；我的随机应变的能力强，我有能力处理好所有发生的意外”，因此如果不做计划，那么这些执行计划、监控计划和变更管理计划的工作就都可以省略，这样才有发挥他们以上能力的空间和平台。如果项目管理人员有以上思想那就完全错了，不做计划的好处是可以省略大量的管理工作，但是项目就会直接面对突如其来的风险和失败，也许有着这种思想的管理人员可以完成一个项目，但是这个项目的整体质量却会非常低下。

1. 制订项目计划的原则

软件项目计划是对软件研发整体质量的详细策划，它是对“软件质量管理的统筹——项目集成管理”的进一步细化，它起着承上启下的作用，在制订项目计划时要把握以下原则：

- 目的明确：任何项目都是有目标的，软件项目的目标不是为了交付一个产品，而是帮助客户实现他的商业目的，因此项目管理人员不要本末倒置搞错了项目的真正目的，否则项目一定会失败。
- 有整体性思维：由“项目集成管理”可知项目计划指的是“项目主计划”或“项目集成计划”，项目计划是由一份主计划和若干份从属计划构成的整体。以“木桶理论”为依据，每份计划都负责管理软件质量体系中的一个部分，如果要想提高软件的整体质量，就必须从质量整体的角度来编制项目所需要的各种计划。
- 可行性强：项目计划一定要切实可行，绝对不能“假、大、空”，千万不要在项目计划中作假，这样害人害己。制订项目计划的过程也是用于判断项目可行性的最好手段，一份准确的项目计划也是可行性分析最好的证明。制订可行的计划目的是尽量避免变更，这样才能做到“指哪打哪”，而不是“打哪指哪”。
- 波浪形的编制方式：软件项目独特性和临时性注定了软件项目就是一个逐步细化、逐步完善的过程，因此在项目之初是无法将所有项目内容全部准确计划的，这才有了软件工程中的迭代式、螺旋式、XP 极限式、原型式的生命周期模型。在制订项目计划时也要采用波浪形滚动的方式来制订当前阶段的计划，初步制订下个阶段的计划，日后再去细化下个阶段的计划。
- 主从计划间的匹配性：要将每份从属计划中的进度和任务的安排、成本、资源、相关人员的安排与职责分派都集中到主计划中，这样便于对众多计划进行协调和解决计划间的冲突，也减少了计划变更时的工作量。

2. 项目计划的层次

软件项目计划的制订是一项科学的、复杂的，而又富有经验性的工作，项目计划的好坏直接关系到项目的成败。一套完整的软件项目计划包含了若干个从属计划，因此项目计划的制订不是项目经理一个人完成的，制订项目计划的责任也不全在于项目经理，而是整个项目团队的工作和责任。一般计划可以分为以下3个层次，如图10-1所示。

- 大项目/复杂项目计划：制订这类计划的人被称为计划经理，它是为了实现组织战略目标而制订的一组具有战略性的计划。例如：“863计划”就是国家为了实现科技强国这个战略性目标而制订的一组计划，在这个大的计划中又会分为若干个具体的项目，每个项目都有自己的计划。再例如某个软件公司的产品经理他需要对公司的产

品发展制订“产品路线图”，这个产品就是该公司战略的体现，在路线图中的每一步都会作为一个或多个项目进行开发，就像微软的“Windows 视窗”从 Win95、Win98、Win2000 一直不断地发展。

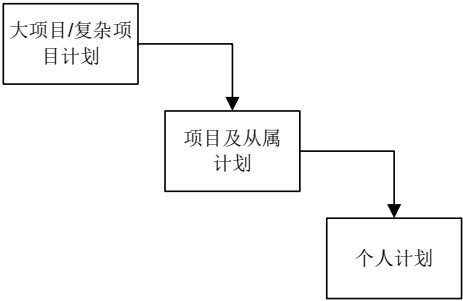


图 10-1 3 个层次的计划

- 项目及从属计划：这个计划是由项目经理负责整合的，而不是全部由项目经理制订的，这个计划是以项目相关人员制订的个人计划为基础进行整合得来的。例如：某软件项目需要“软件测试计划”，那么这个计划就会由测试经理负责起草，由项目经理进行整合，遇到有矛盾的地方再由项目经理进行协调、解决。
- 个人计划：参与项目的每个人都要为项目计划及从属计划贡献自己的知识和力量。配置管理员要负责起草“配置管理计划”；质量保证人员要负责起草“软件质量保证计划”；需求人员要负责起草“需求调研计划”等。就算是一般软件开发人员也要对自己负责的任务进行时间和成本的估算，以支持项目经理来制订计划。

3. 项目计划的目标

在制订软件项目计划时要充分考虑以下 5 个目标，如图 10-2 所示。之前已经反复多次强调了客户出资让软件公司研发一个产品的目的不是要一个程序，而是要通过这个软件实现客户的商业目的，因此软件项目的商业目标是第一位的。

在明确了解商业目标后就可以对软件的质量制订目标，这个质量并不局限于产品，而是整个软件研发的质量。有了质量目标后才能对产品和项目的成败进行判断。

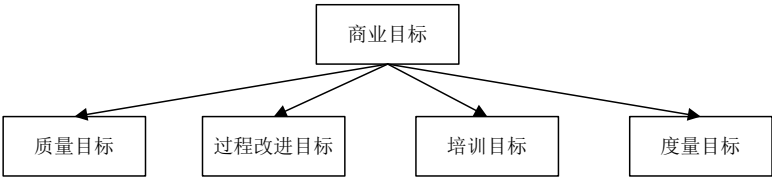


图 10-2 项目计划的 5 个目标

任何一个公司都在向前发展，以往项目中出现的风险和问题就不希望在日后的项目中重蹈覆辙，因此要为每个项目制订过程改进的目标。

任何一个项目能够获得的资源都不会十全十美，如果个别开发人员的能力不够，过程项目需要采用新的技术，那么必然要开展相关的培训，因此项目管理人员要在启动时识别和制订本项目的培训目标。

为了更好地评价商业目标是否实现、质量目标是否达到、培训目标是否完成，需要通过制订适当的度量目标来进行考查。

软件项目计划的这 5 个目标之间也有着一定的层次关系。如图 10-2 所示，项目的商业目标可以分解为项目的质量目标、过程改进目标、培训目标和度量目标。

如图 10-3 所示，项目度量目标的来源就是商业目标、质量目标、过程改进目标和培训目标。例如某项目的商业目标就是客户的满意度；质量目标就可以分解为缺陷率和项目进度及成本偏差率等；过程改进目标就是项目的成熟度；培训目标就可以分解为培训的效果和项目组成员参与培训的时间和成本等。

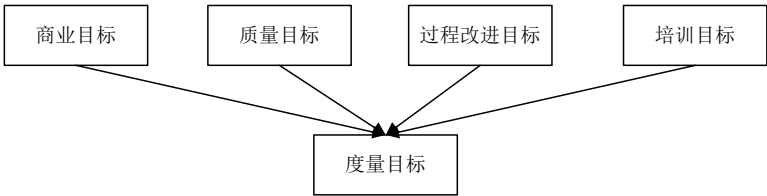


图 10-3 项目计划 5 个目标的层次关系

10.1 软件项目计划概述

1. 如何制订项目计划

此处的项目计划是指项目的进度计划和成本计划，其他的从属计划（如配置管理计划等）请参考其他相关章节。项目计划是项目实施的指导性文件，是所有项目关系人进行沟通的依据。项目计划的制订是一项复杂的、科学的过程，在确定项目目标后就根据“项目集成管理”过程中生成的《过程定义书》来制订项目范围、进度、成本的计划，其过程如图 10-4 所示。

首先会利用“项目集成管理”过程中筛选出来的组织级历史数据对项目进行估算，为了提高项目估算的准确性，项目管理人员可以事先将项目需求进行一些简单的分解，生成顶级的 WBS。

当项目需求确定后，就可以对其进行 WBS 工作结构分解，以确定项目的范围和基准。

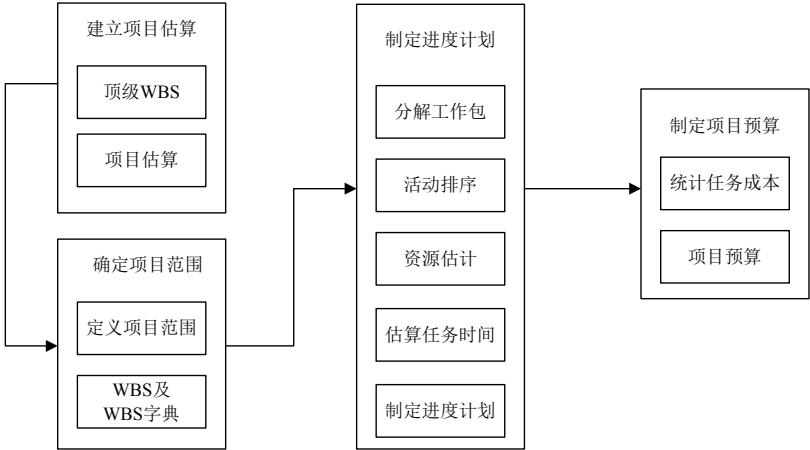


图 10-4 项目计划制订的步骤

确定了范围后就可以将工作产品进行任务分解，对任务进行排序，对任务所需要的资源和耗时进行估计，最后制订出项目进度计划，它是项目进度的基准。

针对每个任务的资源进行成本核实，最后将进度计划中所有任务的费用相加生成项目的预算，它也是项目成本的基准。

2. 软件项目成本分类

在整个项目的过程中所有的任务和活动归根结底都与费用有关，在一个项目的生命周期中，有 5 个地方会统计项目的总费用，如图 10-5 所示。

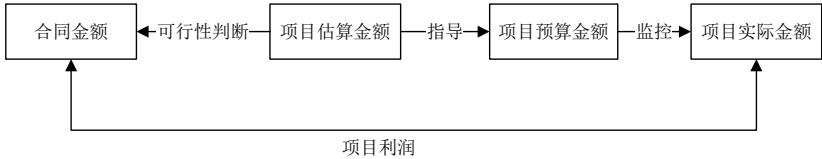


图 10-5 项目成本关系图

在项目立项时合同上注明的金额被称为合同金额，项目结束后由财务人员统计的项目实际花费被称为项目实际金额，它们之间就构成了项目的利润空间。

在项目计划时首先会对项目进行估算，估算时所统计的金额被称为项目估算金额，项目管理人员会将项目估算金额与项目合同金额进行对比，以判断项目的可行性。

在制订项目详细计划时，项目管理人员根据项目的进度基准统计出的金额被称为项目预算金额，项目预算金额是用于指导项目实施的，是项目的成本基准，项目预算金额也被用于与同项目实际金额进行比较，以发现项目的成本偏差。

3. 制订软件项目计划的流程

软件项目计划制订的过程简单来讲就是要解决“3W2H”的问题，它们分别是：“What（做什么）”、“When（何时做）”、“Who（谁来完成）”、“How（如何做）”以及“*How much*（需要多少成本）”。综上所述，项目计划的完整流程如图 10-6 所示。

10.2 软件计划的流程及最佳实践

按照如图 10-6 所示的内容，接下来将对软件项目计划各个环境中所使用到的具体方法进行详细的讲解。

10.2.1 对项目进行整体估算

对项目进行估算的意义在于对项目可行性进行评估，如果一个项目的合同金额是 200 万，可项目估算的结果是 220 万，那么这个项目的可行性就不大。就算公司将该项目作为“政治性”任务完成，那么通过项目估算也可以让公司领导对该项目有更加清楚的认识，以便调整公司对该项目的期望值和绩效指标。

对项目进行估算常采用“对比估算法”或称为“专家判断法”，在使用该方法时，参与估算的人员至少要有两个人，并且他们要独立估算。对估算后的结果进行比较时，如果对某些参数的估算偏差较大，则需要通过讨论确定最终结果。在估算期间使用的所有估算数据都是来自于“项目集成管理”过程中所筛选的组织级历史数据。

下面将以一个 200 万的瀑布式模型软件项目为例，讲解如何利用组织级历史数据对项目进行估算，估算的结果均经过多人共同讨论并确认。

1. 估算项目的范围

该项目合同金额是 200 万，那么根据公司项目级别的定义，该项目属于 A 类项目。在最近 1 年半的时间里，公司共有 6 个 A 类项目相继结束，这些项目的规模集中分布在 490～510 个功能点之间，如图 10-7 所示，因此可以根据这些历史数据估算出本项目的规模大概为 510 个功能点。

2. 估算项目的工作量

同样对这 1 年半内完成的 6 个 A 类项目进行统计，如图 10-8 所示，A 类项目的生产率在 0.6～0.8 个功能点/人天之间。由于公司在 1 年半前就获得了 CMMI 3 级的认证，又经过了这么久时间的积累和锻炼，而且该项目经理觉得他的能力不会比其他项目经理差，另外从图 10-8 上可以看出这 1 年半内生产率有着上升的趋势，因此他估计本项目的生产率是 0.8 个功能点/人天。

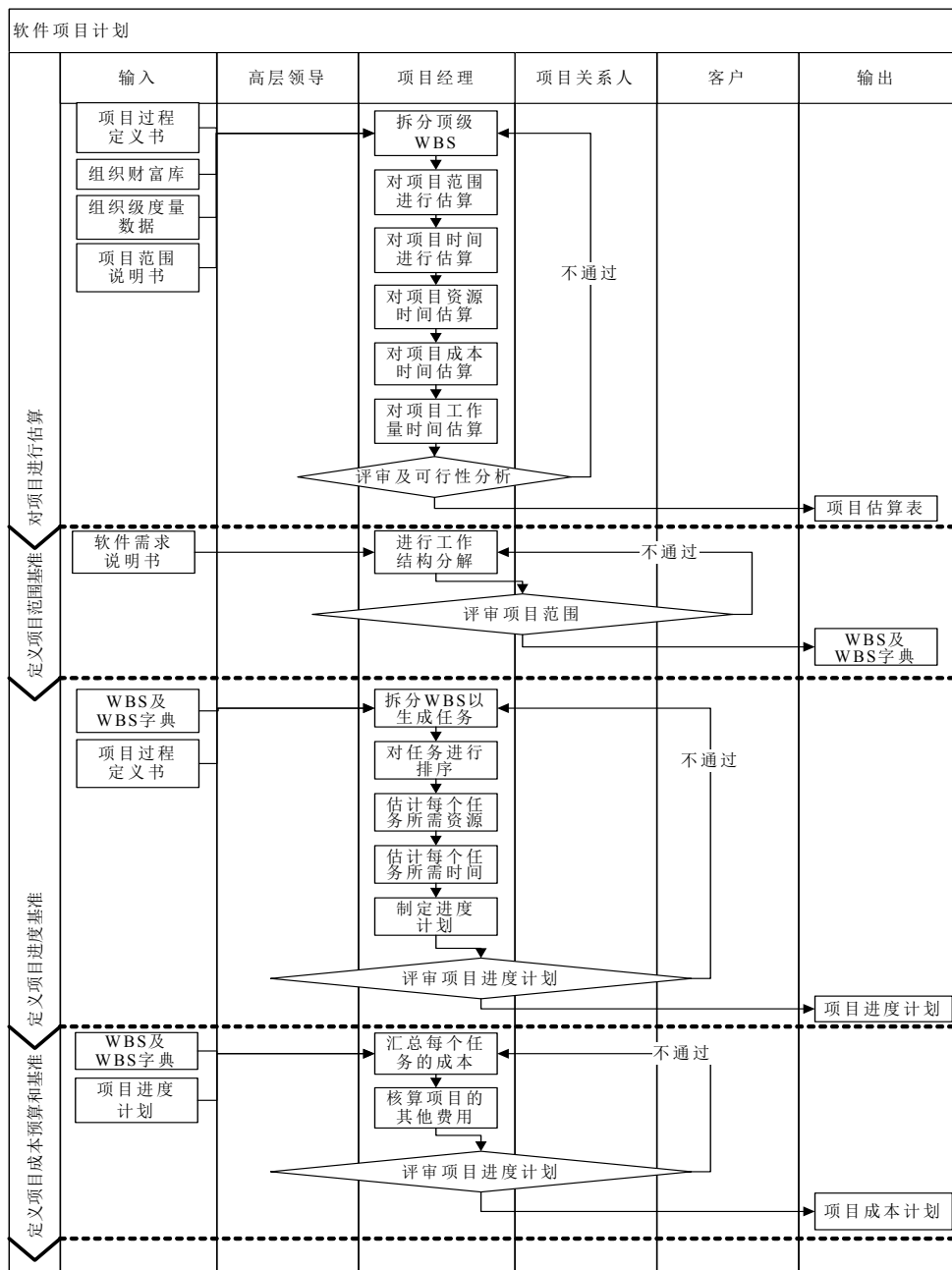


图 10-6 项目计划制订流程图

所以该项目的工作量 = $\frac{510\text{个功能点}}{0.8\text{个功能点 / 人天}} = 637.5\text{人天}$

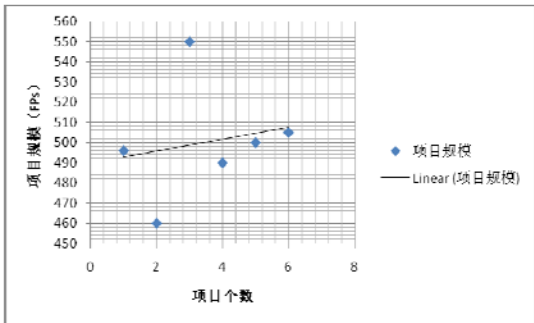


图 10-7 A 类项目规模的历史数据

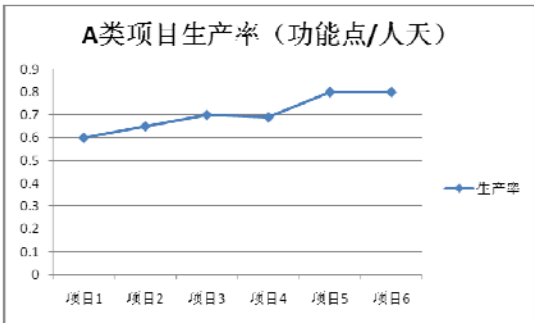


图 10-8 A 类项目的生产率

以这 6 个历史项目的生命周期各阶段工作量分布进行估算，本项目生命周期各阶段工作量分布如表 10-1 所示。

表 10-1 某 A 类项目生命周期各阶段工作量分布

生命周期	本项目 (%)	项目 1 (%)	项目 2 (%)	项目 3 (%)	项目 4 (%)	项目 5 (%)	项目 6 (%)
项目立项	2.00	2.00	1.00	1.82	2.02	2.09	1.78
需求开发	5.00	5.13	4.89	6.21	5.09	5.02	5.57
项目计划	9.00	8.96	8.05	9.23	9.06	9.20	8.64
软件设计	21.00	19.80	21.22	22.87	18.50	19.86	20.35
编码开发	26.00	26.30	25.30	26.34	27.10	26.80	27.05
软件测试	26.00	25.20	26.69	21.44	27.00	24.85	25.40
产品验收	9.00	9.61	9.98	9.93	9.65	9.53	8.65
项目结项	2.00	3.00	2.87	2.16	1.58	2.65	2.56

由于本项目总的工作量已知，生命周期各阶段工作量分布比例也已知，因此可以估算出本项目生命周期各个阶段的工作量，如表 10-2 所示。

表 10-2 某 A 类项目生命周期各阶段工作量

本项目总工作量 637.5 人天		
生命周期	分 布 率	工 作 量
项目立项	2.00%	12.75
需求开发	5.00%	31.88
项目计划	9.00%	57.38

续表

生命周期	分 布 率	工 作 量
软件设计	21.00%	133.88
编码开发	26.00%	165.75
软件测试	26.00%	165.75
产品验收	9.00%	57.38
项目结项	2.00%	12.75

3. 估算项目的资源

如表 10-3 所示，对这 6 个项目的人力资源进行统计并以此为依据估算得出本项目所需要的人力资源。其中配置管理员和软件质量保证人员都与其他项目公用，项目度量的工作由项目经理兼任。

表 10-3 某 A 类项目人力资源估算表

人力资源类别	本项目	项目 1	项目 2	项目 3	项目 4	项目 5	项目 6
项目经理	1	1	1	1	1	1	1
需求人员	1	1	1	1	1	1	1
设计人员	2	2	1	1	2	2	2
开发人员	5	5	4	5	6	5	5
测试人员	4	3	2	3	4	3	3
配置管理员	0.5	0.5	0.5	0.5	0.5	0.5	0.5
质量保证人员	0.2	0.2	0.2	0.2	0.2	0.2	0.2

4. 估算项目的进度

将表 10-2 中每个生命周期阶段的工作量与表 10-3 中项目所需人员的角色关联起来，将这些角色按照需要分配到各个阶段，如表 10-4 所示，就可以得到完成生命周期各个阶段所需要的时间。

表 10-4 项目生命周期各个阶段时长的估算

生命周期	工作量（人天）	参与角色及工作量比例	阶段时长（工作日）	备 注
项目立项	12.75	项目经理 1 人（100%）	12.75	本阶段大约需要 12.75 天
需求开发	31.88	需求开发 1 人（90%） 项目经理 1 人（10%）	28.69 (3.19)	本阶段大约需要 28.69 天，其中项目经理参与需求评审大约花费 3.19 天
项目计划	57.38	项目经理 1 人（60%） 其他人员 12.7 人(40%)	34.43 (1.81)	本阶段大约需要 34.43 天，其中项目相关人员参与计划评审大约花费 1.81 天

续表

生命周期	工作量（人天）	参与角色及工作量比例	阶段时长（工作日）	备 注
软件设计	133.88	设计人员 2 人（90%） 开发人员 5 人（10%）	60.24 （2.68）	本阶段大约需要 60.24 天，其中开发人员参与设计评审大约花费 2.68 天
编码开发	165.75	开发人员 5 人（90%） 设计人员 2 人（10%）	29.84 （8.92）	本阶段大约需要 29.84 天，其中项目设计人员进行代码走查大约花费 8.92 天
软件测试	165.75	测试人员 4 人（100%）	41.43	本阶段大约需要 41.43 天
产品验收	57.38	需求开发 1 人（50%） 项目经理 1 人（50%）	28.69 （28.69）	本阶段大约需要 28.69 天，其中项目经理与需求人员共同完成
项目结项	12.75	全体成员 13.7 人共同参与	0.93	本阶段大约需要 0.93 天

假如项目是从 2009 年 2 月 16 日启动，如图 10-9 所示，那么本项目将于 2010 年 1 月 12 日完工，共耗时 237 个工作日，大约是 11.29 个月。

5. 估算项目的成本

统计这 1 年半完工的 6 个 A 类项目中各个角色的单位成本，其结果如图 10-10 所示。以此为依据对本项目的成本进行估算，其结果如表 10-5 所示。

生命周期	开始日期	完成日期	持续时间
立项阶段	2009-2-16	2009-3-4	12d 6h
需求开发阶段	2009-3-4	2009-4-14	28d 5.52h
项目计划阶段	2009-4-14	2009-6-1	34d 3.44h
软件设计阶段	2009-6-1	2009-8-25	60d 1.92h
编码开发阶段	2009-8-25	2009-10-5	29d 6.72h
软件测试阶段	2009-10-5	2009-12-2	41d 3.44h
产品验收阶段	2009-12-2	2010-1-12	28d 5.52h
项目结项阶段	2010-1-12	2010-1-13	7.44h

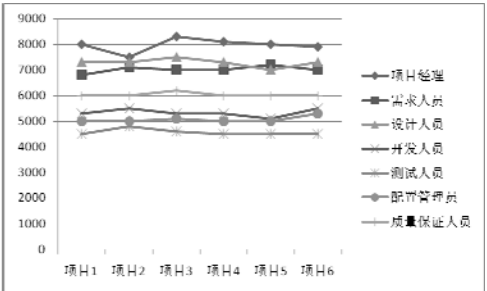


图 10-9 项目进度估算表

图 10-10 历史项目中各项目角色单位成本

表 10-5 估算项目各个角色单位成本（元/月）

项目关系人	单位成本（元/月）	人 数	成本合计（元）
项目经理	8000.00	1	8000.00
需求人员	6800.00	1	7000.00
设计人员	7300.00	2	15000.00

续表

项目关系人	单位成本（元/月）	人 数	成本合计（元）
开发人员	5300.00	5	25000.00
测试人员	4500.00	4	19200.00
配置管理员	5000.00	0.5	2500.00
质量保证人员	6000.00	0.2	1200.00
项目估算成本（元/每月）	77900.00		
项目估算总成本（元）	$77900 \times 11.29 = 879157.14$		

6. 项目估算总结

以最近 1 年半内完工的 6 个 A 类项目为基础，配合组织级的度量数据对本项目进行了估算，项目大约耗时 11.29 个月，估算成本为 879157.14 元，假如项目在 2009 年 2 月 16 日启动，那么将在 2010 年 1 月 13 日完工，本项目需要项目经理和需求人员各 1 人、设计人员 2 人、开发人员 5 人、测试人员 4 人，配置管理员和质量保证人员均与其他项目共享。

10.2.2 对项目范围进行管理

项目的范围是项目诸多属性中的核心，项目成本和进度的计算都是以项目范围为依据的，一旦项目范围发生变化，那么之前计算的项目成本和进度也就失效了。项目范围的变化对软件项目的成功和质量有着非常大的影响，依据不完全统计，绝大多数项目的失败都是与项目范围的变化有关。对软件范围进行管理的目的就是为了确保项目完整，且只完成所必需的工作。

在软件研发中经常会提到项目和产品两个不同的概念，那么产品的范围指的是产品的功能或特性，项目范围指的是完成产品、服务的功能或特性所需要的工作。产品范围是否完成是以产品的功能或特征是否实现为衡量的标准。项目范围是否完成是以项目范围说明书、WBS 和 WBS 字典、项目范围的量化指标为衡量的标准。

范围基准中的项目范围说明书指的就是软件项目中的《软件需求说明书》，其相关内容可以参见本书“需求工程”一章的内容，在这里重点介绍范围基准中的另外 2 个部分：WBS 和 WBS 字典，以及项目范围量化的指标——“标准功能点估算法”。

1. WBS 及 WBS 字典

WBS（Work Breakdown Structure）中文称为工作分解结构，由字面即可知道它的核心由 3 个部分组成：

- Work（工作）：指可以产生有形工作成果的任务。
- Breakdown（分解）：对 Work 进行逐级的细分。

- **Structure（结构）**：体现了分解后的层次关系。

因此 WBS 是面向工作成果的分解，它详细描述了项目所需要完成的所有工作，WBS 的底层可以用来估算项目的进度、资源和成本的预算。

WBS 底层的节点被称为工作包，工作包是指最小可交付的工作产品，它通常可以在 8~40 个小时内完成。

记录 WBS 中所有节点详细信息的文档被称为 WBS 字典，它记录了每个节点所完成工作产品的详细信息、资源、成本、时间、负责人，以及识别出来的风险等内容。

对项目工作产品进行 WBS 分解有以下目的：

- 能够明确项目的范围，使人一目了然
- 准确定义项目的边界，明确所需要完成的工作
- 便于确定所需技术和人员及职责
- 提高了对项目时间、成本和资源估算的准确性
- 是各种计划、预算、进度、成本达成共识的依据
- 防止需求的蔓延
- 有利于识别风险
- 便于对工作产品进行 Make or Buy 的分析

如图 10-11 所示，WBS 分解的过程是个循序渐进的过程，也是一个循环的过程，其步骤说明如下：

STEP 01 识别并确认项目的主要组成部分或生命周期。

STEP 02 编制顶层的 WBS。

STEP 03 对顶层的 WBS 按照工作内容、生命周期等线索进行逐级分解。

STEP 04 为每一个 WBS 节点分配唯一编码进行标识。

STEP 05 为每个 WBS 节点分配负责人和权限。

STEP 06 确定每个 WBS 节点的组成要素，例如：该工作产品是重用还是需要购买或自制。

STEP 07 识别是否存在风险。

STEP 08 核实分解是否正确，主要包括以下几点：

- WBS 是以工作产品为导向进行分解的。
- WBS 上每一个节点必须是下一层工作之和。
- 工作包只能从属于上一层的某个节点，避免出现交叉从属的关系。
- 工作包要具有可验证性。
- 工作包要是必要的并且充分的。
- 工作包要可以支持对项目成本、进度和资源进行估算。
- 工作包要有唯一的责任人，但可以有多人参与。

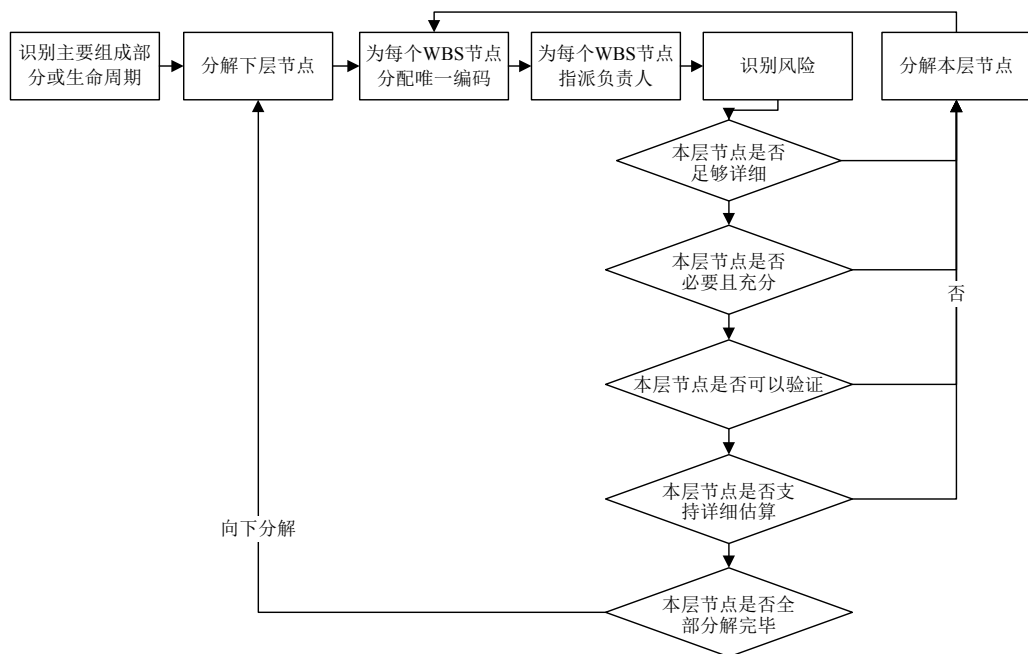


图 10-11 WBS 分解步骤

在软件研发项目中，WBS 分解最简单、最常用的方法就是将 WBS 与软件项目所选择的生命周期模型相关联。如图 10-12 所示，按照生命周期模型的每个阶段进行逐级分解，同时考虑项目管理和项目支持类的工作。在 WBS 分解完成后，项目管理人员还可以将底层所有工作包的内容与“项目集成管理”中对工作产品裁剪的结果进行比较，以免遗漏某些内容。

2. 国际标准功能点估算法

对软件规模估算的常用方法有：功能点估算法和代码行（Line Of Code, LOC）法。

代码行法是统计软件实际有效代码行的一种方法，它通常在项目结束后使用会比较准确。

国际标准功能点估算法（International Function Point Users Group, IFPUG）是由“国际功能点用户组织”提出的一种用于软件项目范围估算的标准。它是从用户角度出发度量软件规模的一种方法，如图 10-13 所示，IFPUG 将系统分为数据功能和人机交互功能两大类，在估算时以《软件需求说明书》为依据，根据其内容将软件分解为若干个可以被单独估算的功能单元，然后对其进行估算，最后还会按照估算人的经验通过调整因子对估算的结果进行调整。IFPUG 常在项目开始或项目需求基本明确时使用，它的好处是估算人无须懂得软件使用何种开发技术。在软件研发的过程中并不是只有在项目开始时才使用，在项目变

更时同样需要对软件的规模进行估算，以对本次变更的大小进行衡量，最后在项目结束时还需要进行一次软件规模的估算，以确定该产品的最终大小。

WBS编码	WBS名称	责任人	风险描述	其他
1	某瀑布式模型软件项目			
1.1	项目立项阶段			
1.2	需求调研阶段			
1.3	项目计划阶段			
1.4	软件设计阶段			
1.4.1	概要设计文档	设计人员		
1.4.2	详细设计文档	设计人员	模块A有技术风险	
1.4.3	设计评审报告	设计和开发人员		
1.4.4	里程碑评审报告	全体项目组成员		
1.5	编码开发阶段			
1.5.1	模块A	开发人员		
1.5.1.1	功能A1	开发人员		
1.5.1.1.1	组件A1	开发人员	有技术风险	需要采购
1.5.1.2	功能A2	开发人员		
1.5.1.2.1	功能A2.1	开发人员		
1.5.2	模块B			
1.5.3	模块C			
1.6	系统测试阶段			
1.7	产品验收阶段			
1.8	项目结项阶段			
1.9	项目管理支持类工作产品			
1.9.1	项目周报	项目经理		
1.9.2	项目度量报告	项目经理		
1.10	质量保证类工作产品			
1.10.1	各种产品检查表	QA		
1.10.2	各种过程检查表	QA		
1.11	项目培训类工作产品			
1.11.1	项目培训计划	项目经理		
1.11.2	项目组成员培训登记表	项目经理		

图 10-12 WBS 分解示意图

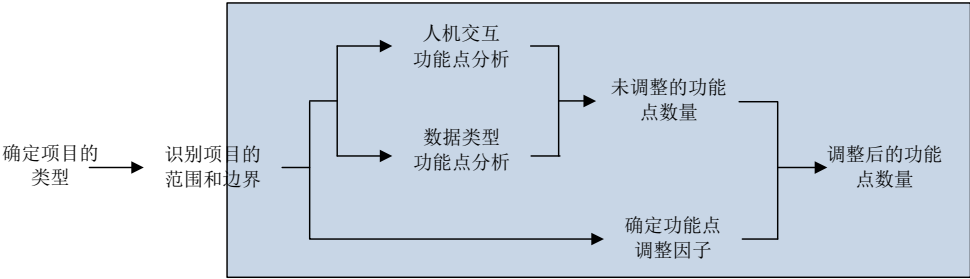


图 10-13 功能点估算法的步骤

■ 确定项目的类型

功能点估算法将软件项目分为以下 3 类：

- ① 新开发项目：从无到有地开发一个系统。
- ② 功能增强的项目：在原有系统基础上新增、完善甚至删除已有的功能。

③ 已存在的项目：通常是对现有软件进行功能点计数。

■ 识别项目的范围和边界

识别项目范围和边界的最好方法就是通过 UML 中的 UseCase 图来完成，如图 10-14 所示，某软件项目的“汇率查询转换”功能是一个外部程序“汇率系统”的功能，而“录入订单”、“修改订单”等模块才是本项目所需要实现的功能。因此掌握画 UseCase 图的使用方法对软件项目范围界定很有帮助。

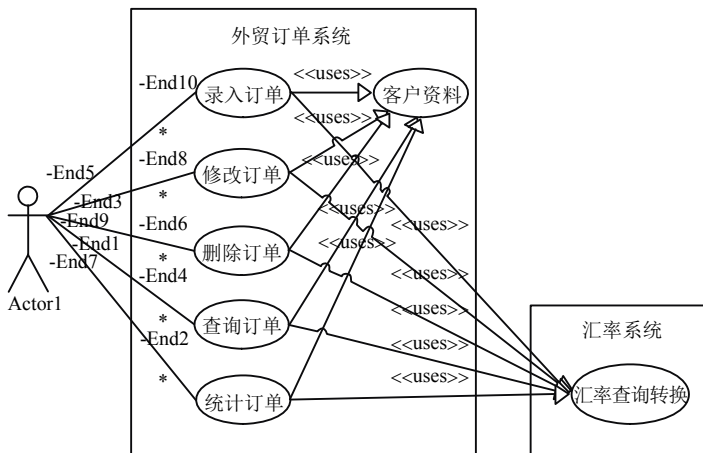


图 10-14 利用 UseCase 图来识别项目的范围和边界

■ 功能点分类

如图 10-13 所示，以一个应用系统的软件为例，可以将其分为数据库开发和程序开发两个部分。如果要对它进行估算，那么就要分别进行。对数据库进行评估时通常会关心数据库中表的数量和数据表中字段的数量。代码开发出来的功能通常是给用户使用的，用户通过程序界面完成了人机交互的操作。因此功能点估算法将功能点分为以下几个类别：

● 数据类型功能点

- ILF: Internal Logical File, 内部逻辑文件
- EIF: External Interface File, 外部接口文件

● 人机交互类型的功能点

- EI: External Input, 外部输入
- EO: External Output, 外部输出
- EQ: External Inquiry, 外部查询

■ 数据类型功能点的估算方法

如图 10-15 所示，对数据类型功能点进行估算时首先要对软件产品的数据文件进行识别，将其分为内部逻辑文件（ILF）和外部接口文件（EIF）。然后通过 RET 和 DET 的个数来计算 ILF 和 EIF 的复杂度，最后可以根据其复杂度得到其功能点的个数。在学习数据类型功能点估算方法之前，先来明确 ILF 和 EIF 的标准定义。

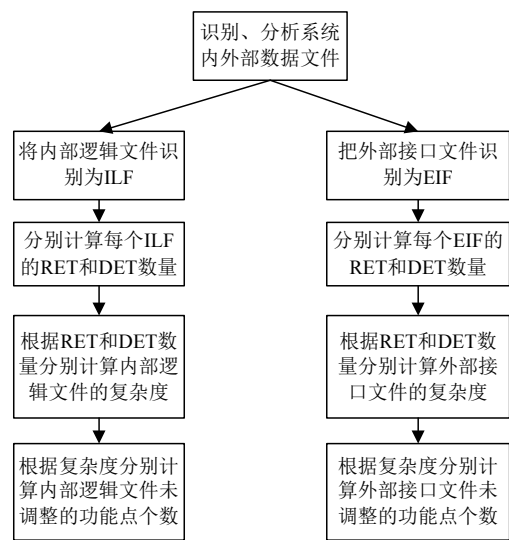


图 10-15 数据类型功能点计算流程

内部逻辑文件是指一组以用户角度识别的在应用程序边界内且被维护的逻辑相关数据或控制信息。ILF 的主要目的是通过应用程序的一个或多个基本处理过程来维护数据。

外部接口文件是指一组在应用程序边界内被查询但在其他应用程序中被维护的，以用户角度来识别的逻辑上相关的数据。从“相对论”的角度来看，一个应用程序中的 EIF 必然是其他应用程序中的 ILF。EIF 的主要目的是为边界内的应用程序提供一个或多个通过基础操作过程来引用的一组数据或信息。

ILF 和 EIF 类型功能点的复杂度取决于 RET（Record Element Type）和 DET（Data Element Type）的数量，下面介绍如何识别 DET 和 RET。

DET 是一个以用户角度识别的、非重复且有业务逻辑意义的字段。它有以下几种规则：

① 通过一个基本处理过程的执行，对 ILF 进行维护或从 ILF/EIF 中返回一个特定的、用户可识别的、非重复的字段，那么每个这样的字段算一个 DET。

例：添加一个外贸订单时需要保存“订单号码、订单日期、地址、邮编”，那么对于 ILF 订单来说它的 DET 就是 4 个。

例：保存订单时还会保存订单的明细，订单的明细往往作为一个子表进行保存，那么

“订单号码”在主表和子表中都同时存在（主外键），但以用户角度来识别时，存盘是一个最小的操作单位，那么“订单号码”只能算做一个 DET。

② 当两个应用程序维护和/或引用相同的 ILF/EIF, 但每个应用程序分别维护/引用它们相应的 DET 时，这些 DET 在这两个应用程序的维护或引用中将单独计算。

例：一个应用程序的两个基本处理过程（Elementary Process）都需要使用到“地址”的信息，地址的信息又可以细分为“国家、城市、街道、邮编”。那么对于其中一个基本处理过程来说，它将整个地址信息作为一个整体进行处理，那就只算一个 DET，另外一个基本处理过程使用每个地址的详细信息，那么 DET 就是 4 个。

RET 是指一个 EIF/ILF 中用户可以识别的 DET 的集合。RET 在 ILF、EIF 中分为两种类型：可选的 RET（Optional）和必选的 RET（Mandatory），其识别规则如下：

在一个 ILF 或 EIF 中，每一个可选或必选的集合都被计算为一个 RET。

或者

如果一个 ILF/EIF 没有子集合，则 ILF/EIF 被计算为一个 RET。

例：在外贸订单系统中添加一个订单时会同时保存客户信息，也有可能保存部门信息。那么订单系统 ILF 就包含了以下 3 个 RET，依据以上判断规则可以得出 RET 的个数是 3。

- 订单信息（必选的）
- 客户信息（必选的）
- 部门信息（可选的）

在对一个软件项目的 ILF 和 EIF 类型功能点的 RET 和 DET 分别统计后，如表 10-6 所示，就可以计算出 ILF 和 EIF 的复杂度。

表 10-6 ILF/EIF 复杂度计算矩阵

	1~19 个 DET	20~50 个 DET	超过 51 个 DET
1 个 RET	低	低	中等
2~5 个 RET	低	中等	高
6 个以上 RET	中等	高	高

当 ILF 和 EIF 功能点的复杂度统计出来以后，就可以根据表 10-7 计算出相应未调整的功能点个数。

表 10-7 ILF/EIF 功能点计算矩阵

功能点类别	低	中 等	高
ILF	7	10	15
EIF	5	7	10

对功能点估算时必须要以用户的角度对需求文档进行分析，但为了便于记忆，客户所提供的每种纸质表单基本上都可以作为一个 ILF。可以将 ILF 和 EIF 的 RET 记忆成数据库中的表，将 DET 记忆成数据表中的字段，再配合 RET 和 DET 的其他规则就很好记忆了。下面我们用一个具体范例进行详细说明。

某软件项目要开发一个员工管理系统，需求如下：

① 员工资料分为 5 个部分：员工的一般信息、员工的教育情况、工作经历、家属信息和所属部门，其详细内容如表 10-8 所示。

表 10-8 员工资料的详细内容

员工一般信息（必填）	部门信息（必填）	受教育情况（可选）	工作经历（可选）	家属信息（可选）
员工 ID（标签控件）	所属部门 ID（标签控件）	受教育的时间	工作时间	亲属的姓名
员工名称	所属部门名称	学校名称	工作单位	之间关系
性别		所学专业	工作部门	亲属年龄
生日			工作职务	工作单位
婚否				

- ② 要有对员工信息进行管理的功能。
- ③ 要有对部门信息进行管理的功能，部门资料的详细内容如表 10-9 所示。
- ④ 员工的工资由另外一个软件系统提供，本项目只需要提供员工的 ID 即可获取相应的工资信息。员工工资资料的详细内容如表 10-10 所示。

表 10-9 部门资料的详细内容

部门信息（必填）
部门 ID（标签控件）部门名称

表 10-10 员工工资资料的详细内容

员工工资信息（必填）
员工 ID（标签控件）
工资单位名称
月工资数量

STEP 01 分别识别系统的 ILF 和 EIF。

如图 10-16 所示，根据以上范例的需求可以画出它的 UseCase，并判断出该系统的 ILF 和 EIF。

- ILF：员工资料、部门资料。
- EIF：员工工资信息。

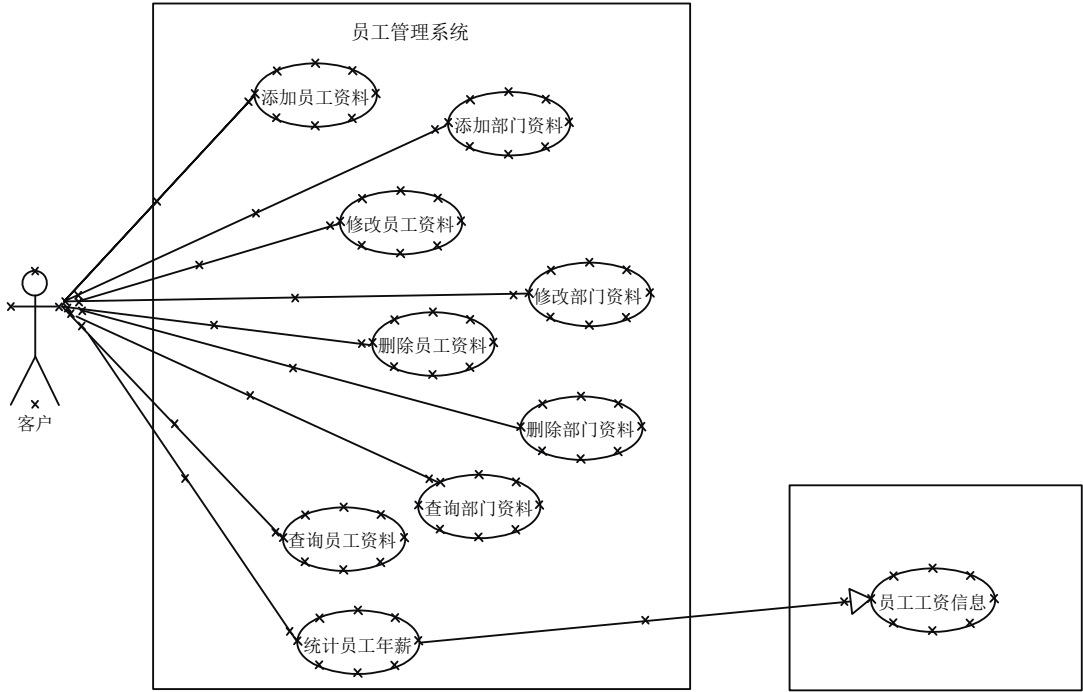


图 10-16 范例-UseCase 图

STEP 02 逐个判断每个 ILF 和 EIF 的 RET 和 DET。

根据表 10-8~表 10-10 的内容，依据 ILF 和 EIF 对 RET 和 DET 的判断规则，可以判断出每个 ILF 和 EIF 的 RET 和 DET，其内容如表 10-11 所示。

表 10-11 数据类型功能点个数

	ILF 员工信息	ILF 部门信息	EIF 员工工资信息
RET	5 个	1 个	1 个
DET	18 个	2 个	3 个
复杂度	低	低	低
未调整的功能点个数	7	7	5

STEP 03 逐个判断每个 ILF 和 EIF 的复杂度。

根据表 10-6 的内容，对每个 ILF 和 EIF 识别出来的 RET 和 DET 的个数进行复杂度判断，其内容如表 10-11 所示。

STEP 04 逐个判断每个 ILF 和 EIF 未调整的功能点个数。

根据表 10-7 的内容，对每个 ILF 和 EIF 识别出来的复杂度进行判断，其未调整的功能

点个数如表 10-11 所示。

■ 人机交互类型的功能点

基本处理过程 Elementary Process 对客户来说是一个有意义的最小的业务操作单位, EI、EO、EQ 的识别都是基于该处理过程的。

EI (External Input, 外部输入) 是处理来自于应用程序边界外部的一组数据的输入操作, 它的主要目的是维护一个或多个 ILF 并/或更改系统数据的行为, 它的识别规则如下:

① 从应用边界之外收到数据。如果进入系统边界内的数据不是一个改变系统行为的控制信息, 那么至少也要改变一个 ILF。

② 对于已识别的基本处理过程, 至少满足以下条件之一。

- 该基本处理过程的逻辑与本应用系统中其他基本处理过程的逻辑不同。该基本处理过程应该具有唯一性。例如: 不能存在两个一模一样的存盘操作。
- 在应用程序边界内, 该基本处理过程所使用的这组数据应该与其他基本处理过程所使用的数据不同。

EO (External Output, 外部输出) 是输送数据到应用程序边界外部的过程。它的主要目的是通过逻辑处理过程向用户呈现信息。该处理过程必须包含至少一个数学公式或计算方法, 或生成派生数据。一个 EO 也可以维护一个或多个 ILF 并/或改变系统行为。它必须全部满足以下内容才能被视为一个 EO。

① 先要从外部发送数据或控制信息到应用程序边界内。

② 为了识别这个过程, 以下三点至少满足一个:

- 该基本处理过程逻辑上必须是唯一的, 该唯一性是指其在应用程序中与其他 EO 或 EQ 的逻辑性上保持唯一。
- 该基本处理过程所使用的数据应该是唯一的, 该唯一性是指其在应用程序中与其他 EO 或 EQ 所使用的数据不同。
- 该基本处理过程所引用的 ILF 或 EIF 文件应该是唯一的, 该唯一性是指其在应用程序中与其他 EO 或 EQ 所引用的 ILF 或 EIF 文件不同。

③ 除了要满足以上规则外, 还要至少满足下列规则之一:

- 在基本操作过程中至少包含一个数学公式或计算方法。
- 在基本操作过程中要产生派生数据。
- 在基本操作过程中至少要修改一个 ILF。
- 在基本操作过程中要改变系统的行为。

EQ (External Inquiry, 外部查询) 是向应用程序边界外发送数据基本处理的过程。其主要目的是从 ILF 或 EIF 中通过恢复数据信息来向用户呈现。该处理逻辑不包含任何数学

公式或计算方法，也不会生成任何派生数据。EQ 不会维护任何一个 ILF，也不会改变应用程序的系统行为。EQ 必须全部满足以下内容。

- ① 先要从外部发送数据或控制信息到应用程序边界内。
- ② 为了识别这个过程，以下三点至少满足一个：
 - 该基本处理过程逻辑上必须是唯一的，该唯一性是指其在应用程序中与其他 EO 或 EQ 的逻辑性上保持唯一。
 - 该基本处理过程所使用的数据应该是唯一的，该唯一性是指其在应用程序中与其他 EO 或 EQ 所使用的数据不同。
 - 该基本处理过程所引用的 ILF 或 EIF 文件应该是唯一的，该唯一性是指其在应用程序中与其他 EO 或 EQ 所引用的 ILF 或 EIF 文件不同。
- ③ 除了要满足以上规则外，还要至少满足下列规则之一：
 - 基本操作过程从 ILF 或 EIF 中获取数据。
 - 基本操作过程不能包含数学公式或计算方法。
 - 基本操作过程不能生成派生数据。
 - 基本操作过程不能维护任何一个 ILF。
 - 基本操作过程不能改变系统的行为。

EI、EO 和 EQ 虽然都是人机操作的行为，但是它们彼此间有相似之处，可它们的目的性却不相同，具体区别如表 10-12 所示。

表 10-12 EI、EO 和 EQ 的目的性对比

目 的	EI	EO	EQ
改变应用程序的属性或行为	主要目的	次要目的	不允许
维护一个或多个 ILF	主要目的	次要目的	不允许
显示信息给用户	次要目的	主要目的	主要目的

如表 10-13 所示，EI、EO 和 EQ 除了目的性不同外，它们的行为也有着显著的区别。

表 10-13 EI、EO 和 EQ 行为的对比

行 为	EI	EO	EQ
数学公式或计算被执行	可选	至少选择一次	不可以
至少一个 ILF 被修改	至少选择一次	至少选择一次	不可以
至少一个 ILF 或 EIF 被引用	可选	可选	必选
派生数据被创建	可选	至少选择一次	不可选
应用程序的行为或属性被修改	至少选择一次	至少选择一次	不可选
准备或呈现信息到系统边界外	可选	必选	必选
接受进入系统边界内数据的能力	必选	可选	可选

如图 10-17 所示，EI、EO 和 EQ 的复杂度取决于 FTR 和 DET 的数量。在计算时首先要以用户角度识别最小的业务操作，然后根据 EI、EO 和 EQ 的识别规则将其进行分类，统计每个最小业务操作 FTR 和 DET 的数量，最后根据其复杂度得出功能点的个数。DET 的定义与上面相同。

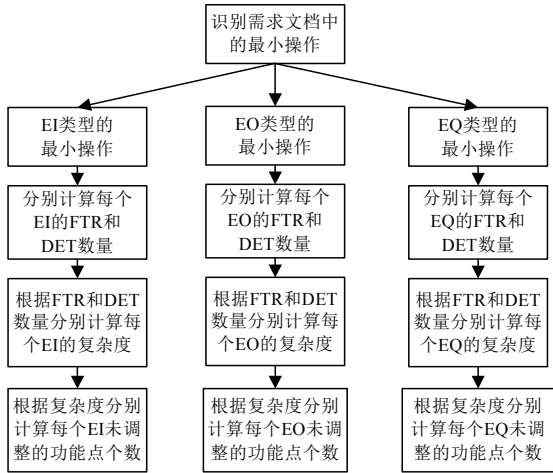


图 10-17 人机操作类型功能点计算流程

FTR（File Type Referenced）是被一个基本处理过程读取或维护的 ILF，或者是被一个基本处理过程读取的 EIF。

EI 对 FTR 的识别规则如下：

- 每一个 ILF 应该算做一个 FTR。
- 通过 EI 读取操作的每个 ILF 或 EIF 都应该被计算为一个 FTR。
- 既被 EI 维护又被读取的 ILF 仅计算一个 FTR。

EI 对 DET 的识别规则如下：

- 在 EI 的过程中，以用户角度识别的，通过应用系统边界输入系统内部的非重复的字段，那么该字段应算一个 DET。
- 在 EI 过程中，只要没有通过系统边界输入，就算它存在于系统内的一个 ILF 中，也不能算为一个 DET。

例：外贸订单系统中，订单的金额是被单价和数量自动计算的，而且金额被数据库保存，那么金额是没有通过系统边界输入的，因此在 EI 操作中就不算做一个 DET。

- 在应用程序的 EI 操作时，系统提示的错误信息应该被分别计算为一个 DET。

例：在网站注册用户信息时，由于输入错误系统会显示提示信息，那么这些提示信息

应该被逐个计算为一个 DET。

- 当 EI 操作完成时系统提示并显示出来的信息，应该被计算为 DET。

例：系统提示“存盘成功”。

- 在 EI 操作中如果遇到主外键的字段，应该算作一个 DET。

根据以上规则可以对 FTR 和 DET 的数量进行识别，然后根据表 10-14 判断其复杂度。

表 10-14 EI 复杂度计算矩阵

	1~4 个 DET	5~15 个 DET	多于 16 个 DET
0~1 个 FTR	低	低	中等
2 个 FTR	低	中等	高
大于 2 个 FTR	中等	高	高

EQ 对 FTR 的识别规则是每个在 EQ 处理过程中读取的 ILF 或 EIF 算一个 FTR。

EO 对 FTR 的识别规则如下：

- 每个在 EO 处理过程中读取的 ILF 或 EIF 算一个 FTR。
- 在 EO 处理过程中每个被维护的 ILF 算一个 FTR。
- 在 EO 处理过程中既被读取又被维护的 ILF 仅算一个 FTR。

EO 和 EQ 对 DET 的识别规则如下：

- 用户可识别的非重复的字段；进入应用边界并指明该操作处理了什么以及何时进行的处理或处理的方式，由 EO/EQ 返回或产生，那么这样的每个字段算一个 DEL。

例：在报表中的每个字段都是一个 DET。

例：报表的查询条件都是一个 DET。

- 在应用边界内以用户角度识别的，非重复字段算一个 DET。
- 在报表上起到解释或备注作用的文字信息，不管它是一个字、一个词或一段话，都当作一个 DET。
- 某种编号或日期，就算它被物理存储在不同字段中，但从用户角度来看是一个整体的信息，因此被算作一个 DET。
- 在 EO 或者 EQ 操作中，如果对系统进行输入或读取操作时，相同的字段只计算一个 DET。

例：在报表查询时，输入的字段在报表上也有显示，那么将算作同一个 DET。

- 在应用程序的 EO 或 EQ 操作时，系统提示的错误信息或完成操作的信息，应该被计算为 DET。

例：用户查询一个列表时被拒绝，那么拒绝的提示信息就算为一个 DET。

- 在 EO 或 EQ 操作中如果遇到主外键的字段，应该算作一个 DET。

- 如果在 EO 或 EQ 过程中，只要没有通过系统边界输入，就算它存在于系统内的一个 ILF 中，也不能算为一个 DET。
- 例：在打印工资条的时候，员工对应的状态信息被更改，但这个状态信息的更新没有通过人机操作的方式完成，因此状态字段不能算作一个 DET。
- 页面的标题等类似的信息不计算 DET。
 - 系统字段生成的记号不能被算作一个 DET。
- 例：在打印页面上的“页码、页数、时间、上一页、下一页”等信息。
- 在饼图中百分比和分类算作不同的 DET。
- 根据以上规则可以对每个 EO 和 EQ 的 FTR 和 DET 的数量进行识别，然后根据表 10-15 判断其复杂度。

表 10-15 EO 和 EQ 复杂度计算矩阵

	1~个 DET	6~9 个 DET	多于 20 个 DET
0~个 FTR	低	低	中等
2~个 FTR	低	中等	高
大于 4 个 FTR	中等	高	高

当 EI、EO 和 EQ 功能点的复杂度统计出来以后，就可以根据表 10-16 计算出相应未调整的功能点个数。

表 10-16 EI、EO 和 EQ 功能点计算矩阵

功能点类别	低	中等	高
EI	3	4	6
EO	4	5	7
EQ	3	4	6

- 首先要从业务角度识别最小的操作，例如：提交部门经理审批操作、存盘操作、查询操作等。在识别 EI 的 DET 时要特别注意，凡是没有通过界面输入的字段都不能算作一个 DET。下面我们用一个具体的范例进行详细说明。
- 某员工管理系统需求的描述如下：
- ① 添加员工信息时，员工的 ID 是系统自动生成的。
 - ② 查询员工信息时，客户可以对查询条件进行组合，查询条件有“员工 ID、员工姓名、部门名称”。查询后显示的内容有“员工 ID、员工姓名、部门名称、性别、生日、婚否”。
 - ③ 统计员工年薪时，客户可以根据以下条件进行统计：“员工 ID、部门 ID”。统计后显示的内容有“员工 ID、员工姓名、部门名称、月薪、年薪、金额单位”。

④ 删除员工信息的操作如下：在对员工查询后，单击“员工 ID”就可以打开删除员工资料的界面，客户单击删除按钮就可以删除该“员工 ID”的资料。

⑤ 修改员工信息的操作如下：在对员工查询后，单击“员工 ID”就可以打开修改员工资料的界面，客户可以对除了员工 ID、姓名外的所有内容进行修改。

⑥ 添加、修改、删除员工信息的操作使用同一个界面，只是个别控件的只读属性设置不同。

⑦ 添加部门信息时，部门的 ID 是系统自动生成的。

⑧ 查询部门信息时，客户可以对查询条件进行组合，查询条件只有“部门名称”。查询后显示的内容有“部门 ID、部门名称”。

⑨ 删除部门信息的操作如下：在对部门查询后，单击“部门 ID”就可以打开删除部门资料的界面，客户单击删除按钮就可以删除该“部门 ID”的资料。

⑩ 修改部门信息的操作如下：在对部门查询后，单击“部门 ID”就可以打开修改部门资料的界面，客户可以对“部门名称”进行修改。

⑪ 添加、修改、删除部门信息的操作使用同一个界面，只是个别控件的只读属性设置不同。

STEP 01 识别 EI、EO 和 EQ。

如图 10-16 所示，根据 EI、EQ 和 EO 的识别规则，可以将其最小的业务操作进行分类，结果如表 10-17 所示。由于添加、修改和删除操作都会修改 ILF，因此它们都属于 EI；查询操作没有数学运算，也没有产生派生数据，因此是 EQ；统计员工年薪是会进行数学运算并生成派生数据，因此是 EO。

表 10-17 EI、EQ、EO 的识别

最小业务操作	EI	EQ	EO
添加员工信息	√		
修改员工信息	√		
删除员工信息	√		
查询员工信息		√	
统计员工年薪			√
添加部门信息	√		
修改部门信息	√		
删除部门信息	√		
查询部门信息		√	

STEP 02 识别 EI 的 FTR 和 DET。

该范例的 FTR 和 DET 如表 10-18 所示。

表 10-18 FTR 和 DET 的识别

最小业务操作	类别	FTR	DET	备 注
添加员工信息	EI	2	16	根据该范例的需求，参考表 10-8 中员工资料的详细内容，在添加操作时员工 ID 是系统自动生成的，部门的 ID 是不能够输入的，因此添加员工信息操作的 DET 为 18-2=16 个； 该操作会对 ILF 员工资料进行维护，也会读取 ILF 部门资料的信息供客户选择，因此其 FTR 为 2
修改员工信息		2	15	根据需求的描述，在修改员工信息时除了员工 ID 和姓名不能修改外其他都可以修改。另外界面上的部门 ID 是标签控件 18-3=15 个； FTR 判断的方法与添加员工信息相同
删除员工信息		2	1	删除员工信息的操作是以“员工 ID”为条件的，因此 DET 为 1； FTR 判断的方法与添加员工信息相同
添加部门信息		1	1	根据该范例的需求，参考表 10-9 中部门资料的详细内容，在添加操作时部门 ID 是系统自动生成的，因此添加部门信息操作的 DET 为 2-1=1 个； 该操作会对 ILF 部门资料进行维护，因此其 FTR 为 1
修改部门信息		1	1	根据该范例的需求，参考表 10-9 中部门资料的详细内容，在修改操作时只能对部门名称进行修改，因此 DET 为 2-1=1 个； FTR 判断的方法与添加部门信息相同
删除部门信息		1	1	根据该范例的需求，参考表 10-9 中部门资料的详细内容，删除部门信息的操作是以“部门 ID”为条件的，因此 DET 为 1； FTR 判断的方法与添加部门信息相同
查询员工信息	EQ	2	3	根据该范例的需求，查询条件是“员工 ID、员工姓名、部门名称”。查询后显示的内容有“员工 ID、员工姓名、部门名称、性别、生日、婚否”，但“员工 ID、员工姓名、部门名称”发生重复，因此 DET 为 6-3=3 个； FTR 判断的方法与添加员工信息相同
查询部门信息		1	1	根据该范例的需求，查询条件只有“部门名称”，查询后显示的内容有“部门 ID、部门名称”。但“部门名称”发生重复，因此 DET 为 2-1=1 个； FTR 判断的方法与添加部门信息相同
统计员工年薪	EO	3	5	根据该范例的需求，根据以下条件进行统计：“员工 ID、部门 ID”，统计后显示的内容有“员工 ID、员工姓名、部门名称、月薪、年薪、金额单位”。但是“员工 ID”发生重复，因此 DET 为 6-1=5 个； 在统计员工年薪时会查询 ILF 员工资料和部门资料，还会查询 EIF 员工工资信息，因此 FTR 为 2+1=3

STEP 03 识别 EI、EQ 和 EO 的复杂度。

根据表 10-18 中 FTR 和 DET 的个数，参考表 10-15 的复杂度计算矩阵，可以得到该

员工管理系统各个 EI、EQ 和 EO 的复杂度，其内容如表 10-19 所示。

表 10-19 范例-人机交互类型功能点个数

最小业务操作	类 别	FTR	DET	复杂度	未调整的功能点个数
添加员工信息	EI	2	16	中等	4
修改员工信息		2	15	中等	4
删除员工信息		2	1	低	3
添加部门信息		1	1	低	3
修改部门信息		1	1	低	3
删除部门信息		1	1	低	3
查询员工信息	EQ	2	3	低	3
查询部门信息		1	1	低	3
统计员工年薪	EO	3	5	低	4

STEP 04 计算 EI、EQ 和 EO 的功能点个数。

当统计完 EI、EQ 和 EO 的复杂度后，就可以根据表 10-16 的指导来分别计算未调整的功能点个数，结果如表 10-19 所示。

● 确定功能点调整因子

功能点的调整因子是通过 14 个常规系统特性及其影响程度（DI）来评定的，对每个 DI 又分为 0~5 共 6 个级别：

- 0 毫无影响
- 1 偶然影响
- 2 适度影响
- 3 一般影响
- 4 重要影响
- 5 强烈影响

当依次对这 14 个系统常规特性进行打分后，就可以代入以下计算公式算出功能点的调整因子：

VAF(Value Adjustment Factor)=(sum of(DI)×0.01)+0.65

下面我们分别对 14 个常规系统的特性进行详细介绍。

① 数据通信

数据通信指的是应用程序与处理器通信的程度。通常都是通过某种通信手段来实现在一个应用程序中使用数据和控制信息的接收和发送。协议指的是两个系统或者两个设备之间的一种约定。所有的数据通信链接都需要某种协议。数据通信对系统影响程度如表 10-20 所示。

表 10-20 常规系统特性——数据通信

0	应用程序是单纯的批处理或者单独的一台 PC
1	应用程序是一种批处理过程，但是包含远程数据的入口或远程打印
2	应用程序是一种批处理过程，但是包含远程数据的入口和远程打印
3	应用程序包括了查询系统的数据收集或在线链接批处理或远程处理的终端应用
4	应用程序不单只是终端应用，而且仅支持一种远程处理通信协议
5	应用程序不单只是终端应用，还支持多于一种的远程处理通信协议

② 分布式数据处理程度

分布式数据处理程度是指应用系统内部组件之间传递信息的程度。这个特性是在应用系统边界内体现的。分布式数据处理对系统影响程度如表 10-21 所示。

表 10-21 常规系统特性——分布式数据处理程度

0	应用程序不支持组件之间的数据传输和处理功能
1	应用程序可能有进行分布式处理的数据（例如使用电子表格或者数据库等）
2	应用程序所准备的数据被传输并被其他组件所使用
3	分布式处理和数据传输是在线的，并且是单向的
4	分布式处理和数据传输是在线的，并且是双向的
5	处理功能由系统中最恰当的组件动态地去执行

③ 性能

性能是吞吐量、响应时间等指标对开发的影响。用户所提出的性能要求将直接影响到系统的设计、实施、安装和支持。性能对系统影响程度的分级如表 10-22 所示。

表 10-22 常规系统特性——性能

0	用户没有提出性能方面的要求
1	用户提出了性能和设计方面的要求，但不需要采取特定措施
2	响应时间或吞吐量达到系统峰值的时间是关键的，但是对 CPU 的利用不需要采取相应性能方面的特殊设计。处理的极限在日后考虑
3	在任何时候的响应时间或吞吐量都是关键的，但是对 CPU 的利用不需要采取相应性能方面的特殊设计。处理的完成期限是强制的
4	除了 3 的要求外，由于对性能的要求比较严格，在设计阶段就要进行性能分析
5	除了 4 的要求之外，在设计和实施阶段需要使用性能分析工具来判断性能要求的完成情况

④ 高强度的配置

高强度配置指的是计算机资源对应用系统的影响程度。它对设计有特殊要求，是必须

考虑的一个系统特性，例如：某软件需要在一台高配置的系统上运行。它对系统影响程度如表 10-23 所示。

表 10-23 常规系统特性——高强度配置

0	没有提出明确的运行方面的限制
1	有运行方面的限制，但是不需要采取特别的措施以满足该要求
2	提出了一些安全和时间方面的限制
3	应用程序的某些部分对性能有特定的要求
4	用户提出的运行限制对应用程序的性能有特殊的要求
5	除了 4 之外，还对应用系统的分布式组件提出了限制

⑤ 事务处理的速度

事务处理的速度是用户对业务交易处理速度的要求，对系统的设计、实施、安装和支持等有影响。它对系统影响程度如表 10-24 所示。

表 10-24 常规系统特性——事务处理率的速度

0	预计不会出现周期性的事务处理高峰期
1	预计会有周期性的事务处理高峰期（例如：每月、每季、每年）
2	预计每周都会出现事务处理高峰期
3	预计每天都会出现事务处理高峰期
4	用户在应用程序需求或者服务协议中对事务处理率要求很高，因此必须在设计阶段进行性能分析
5	用户在应用程序需求或者服务协议中对事务处理率要求很高，因此必须进行性能分析，并在设计、开发和安装阶段中使用到性能分析工具

⑥ 在线数据输入

在线数据输入是指数据通过交互的方式进入系统程度。它对系统影响程度如表 10-25 所示。

表 10-25 常规系统特性——在线数据输入

0	所有事务都是批处理的
1	1%~7%的事务是以交互式的方式完成的
2	8%~15%的事务是以交互式的方式完成的
3	16%~23%的事务是以交互式的方式完成的
4	24%~30%的事务是以交互式的方式完成的
5	30%以上的事务是以交互式的方式完成的

⑦ 最终用户效率

最终用户效率是指应用系统对操作友好性，以及使用的便捷方面考虑的程度。下列功能设计是针对最终用户效率的，每选择一项加 1 分，它对系统影响程度如表 10-26 所示。

- 页面导航（例如：快捷键、跳转等）
- 菜单
- 在线帮助或文档
- 光标自动跳转
- 屏幕可以滚动
- 在线远程打印
- 预定义的功能键
- 在线方式完成批量处理任务
- 光标可以选取界面上的数据
- 使用大量反白、高亮、下画线或其他标识
- 用户文档
- 鼠标拖动功能
- 弹出窗体
- 使用最少的界面完成某种商业功能
- 双语言支持（如果选择了这个就加 4 分）
- 多语言支持（如果选择了这个就加 6 分）

表 10-26 常规系统特性——最终用户效率

0	一个都不选择
1	1~3 分
2	4~5 分
3	6 或 6 分以上，但是没有用户对效率有要求
4	6 或 6 分以上，对用户使用效率有较高要求，因而必须考虑界面方面的设计（例如，最少击键次数、尽可能提供默认值、模板的使用）
5	6 或 6 分以上，用户对效率的要求使得开发人员必须使用特定的工具和流程来判定是否满足用户的要求

⑧ 在线更新

在线更新是指内部逻辑文件（ILF）被在线更新的程度。它对系统影响程度如表 10-27 所示。

表 10-27 常规系统特性——在线更新

0	没有在线更新
1	包含 1~3 个 ILF 被在线更新。更新的数据量少，恢复容易

续表

2	包含对 4 个以上的 ILF 被在线更新。更新的数据量少，恢复容易
3	包含对主要 ILF 的更新
4	除了 3 之外，在设计和实施中要考虑对数据的完整性
5	除了 4 之外，大量的数据恢复工作要考虑成本，同时包含了高度自动化的恢复流程

⑨ 复杂度处理

复杂度处理描述了逻辑处理对应用系统的影响程度。它包含以下要素，其对系统影响程度如表 10-28 所示。

- 权限控制（例如特殊的审核过程）或程序特定的安全处理
- 大量的逻辑处理
- 大量的数学运算
- 因为意外处理造成的需要重新处理的情况（例如通信中断等）
- 多种可能的输入/输出造成的复杂处理

表 10-28 常规系统特性——复杂处理

0	上面一个都不满足
1	只满足一个
2	只满足两个
3	满足三个
4	满足四个
5	都满足

⑩ 可重用性

应用系统中的功能或代码经过特殊设计，可以在其他应用系统中复用。它对系统影响程度如表 10-29 所示。

表 10-29 常规系统特性——可重用性

0	没有可复用的代码
1	代码在应用系统之内复用
2	程序中被其他系统复用的部分不足 10%
3	程序中被不止一个系统重用的部分超过 10%
4	应用系统按照一种易于复用的方式被打包和文档化，用户通过源代码来进行复用
5	应用系统按照一种易于复用的方式被打包和文档化，用户通过参数来进行复用

⑪ 易安装性

易安装性指应用系统安装的容易程度。它对系统影响程度如表 10-30 所示。

表 10-30 常规系统特性——易安装性

0	用户对安装没有特定的要求
1	用户对安装没有特定的要求，但对安装环境有特定要求
2	用户提出了安装的要求，安装指南经过评审后会提供给用户，但是打包的影响对应用程序并不重要
3	用户提出了安装的要求，安装指南经过评审后会提供给用户，打包的影响对该应用程序来说非常重要
4	除了 2 的要求之外，需要提供经过测试的自动化安装工具
5	除了 3 的要求之外，需要提供经过测试的自动化安装工具

⑫ 易操作性

易操作性是应用系统提供的一种特性，它尽量减少人为操作的要求。例如：有效的启动、备份和恢复对系统的影响。它对系统影响程度如表 10-31 所示。

表 10-31 常规系统特性——易操作性

0	用户没有指定除正常备份程序外的其他特定要求
1	提供高效的启动、备份和恢复的流程，但需要人为操作
2	在 1 的基础上，提供高效的启动、备份和恢复的流程，不需要人为操作
3	在 2 的基础上，应用程序对磁带的需求最小化
4	在 3 的基础上，应用程序对硬拷贝处理的需求最小化
5	应用系统设计为全自动的模式。全自动模式的意思是除了启动和关闭之外，不需要人为对系统进行任何操作。 应用系统要有错误自动恢复的能力

⑬ 多场地

多场地指应用系统经特殊设计可以在多个组织或多个地点同时运行的能力。它对系统影响程度如表 10-32 所示。

表 10-32 常规系统特性——多场地

0	用户需求不含多场地或组织的要求
1	考虑了多场地的要求，但是应用系统要求在不同的场地使用相同的软、硬件环境
2	考虑了多场地的要求，但是应用系统要求在不同的场地使用相似的软、硬件环境
3	考虑了多场地的要求，但是应用系统要求在不同的场地使用不同的软、硬件环境
4	在 1 或者 2 的要求之上，提供了经过测试或评审的多场地文档和支持计划
5	在 3 的要求之上，提供了经过测试或评审的多场地文档和支持计划

⑭ 支持变更

支持变更指的是应用系统在设计上考虑了允许处理逻辑和数据结构变化的程度。可以具有如下的特性，它对系统影响程度如表 10-33 所示。

- 提供可以处理简单复杂度要求的组合查询或报告功能，例如对一个 ILF 进行与/或逻辑运算。
- 提供可以处理中等复杂度要求的组合查询或报告功能，例如对多个 ILF 进行与/或逻辑运算。
- 提供可以处理高复杂度要求的组合查询或报告功能，例如对多个 ILF 进行的与/或逻辑的组合运算。
- 业务控制数据被保存到数据表中，用户使用在线的方式对数据表进行维护，但维护的结果只会在第二个工作日生效。
- 业务控制数据被保存到数据表中，用户使用在线的方式对数据表进行维护，但维护的结果即时生效。

表 10-33 常规系统特性——支持变更

0	一个都不满足
1	合计满足一个
2	合计满足二个
3	合计满足三个
4	合计满足四个
5	合计满足五个

下面我们用一个具体的范例进行详细说明。

对员工管理系统的常规特性进行逐一打分，如表 10-34 所示。

表 10-34 员工管理系统常规特性

软件系统常规特性	得 分
1 数据通信	3
2 分布式数据处理程度	4
3 性能	3
4 高强度的配置	4
5 事务处理率的速度	4
6 在线数据输入	3
7 最终用户效率	2
8 在线更新	4
9 复杂处理	4
10 可重用性	3

续表

软件系统常规特性	得 分
11 易安装性	2
12 易操作性	2
13 多场地	3
14 支持变更	4
总分	45

根据调整因子的计算公式可以算出该员工管理系统的调整因子是：

$$VAF=(45\times 0.01)+0.65=0.45+0.65=1.10$$

● 功能点计算公式

在学习功能点计算公式以前先了解以下计算公式的常用术语，如表 10-35 所示。

表 10-35 功能点计算公式常用术语

术 语	英 文	中 文
ADD	Added functionality	被添加的功能点个数
CFP	Conversion functionality	被转换的功能点个数
CHGA	UFP of changed functionality after enhancement	功能增强后改动的功能所贡献的未调整的功能点个数
DEL	Deleted functionality	被删除的功能点个数
UFP	Unadjusted functional point count	未调整的功能点个数
VAF	Value adjustment factor VAF=(sum of(DI)×0.01)+0.65	功能点的调整因子的计算公式 VAF=(sum of(DI)×0.01)+0.65
VAFA	Value adjustment factor after enhancement	功能增强后的功能点调整因子
VAFB	Value adjustment factor before enhancement	功能增强前的功能点调整因子

功能点计算的公式根据项目类型的不同而有所变化，功能点最原始的公式如下所示：

功能点的个数（FP）=未调整的功能点个数（UFP）×调整因子（VAF）

有时新开发的软件项目也需要与其他现存的软件系统进行整合，例如：一个企业新开发的 MIS 内部管理系统经常会与财务系统进行整合。这个时候除了考虑本身项目的功能点个数外，还要考虑系统整合或数据迁移部分的工作量新开发项目的功能点计算公式：

$$FP=(UFP+CFP)\times VAF$$

对于功能增强的项目功能点估算比较复杂，在功能点增强前需要计算有哪些是新增加的功能，有哪些是被修改的功能，还有哪些是属于数据迁移或系统整合的功能。然后计算新系统技术复杂度的调整因子“VAFA”，并在此基础上计算系统功能点的数量。当然此类项目也会去掉一些原有功能，那么在原有系统的技术复杂度基础上重新计算功能点的调

整因子“VAFB”，再计算所去掉的一些原有功能贡献的功能点数量。功能增强类型项目功能点计算公式：

$$FP=[(ADD+CHGA+CFP)\times VAF A]+(DEL\times VAF B)$$

对现有软件进行功能点估算时可以直接使用原始的计算公式。现货软件产品功能点计算公式：

$$FP=UFP\times VAF$$

下面我们用一个具体的范例进行详细说明。

假如某员工管理系统是一个新开发的软件项目。如表 10-11 所示的数据类型功能点个数为 19 个（7+7+5=19）；人机交互类型功能点个数为 30 个（4+4+4+3+3+3+3+3+3=30）。因此本项目未调整的功能点个数 $UFP=30+19=49$ 个。

假如该系统还需要与财务系统进行整合，假设 CFP 类型功能点的数量大约为 11 个。那么利用相应计算公式即可得出本软件项目的功能点个数为：

$$FP=(UFP+CFP)\times VAF=(49+11)\times 1.10=66 \text{ 个}$$

10.2.3 建立时间进度计划

建立时间进度计划的目的是为了确保项目按时完成所需要的各项活动。被批准的项目计划就成为了项目的进度基准，它是项目关系人对项目任务安排、时间安排达成共识的基础。编制项目进度计划的步骤如图 10-18 所示。

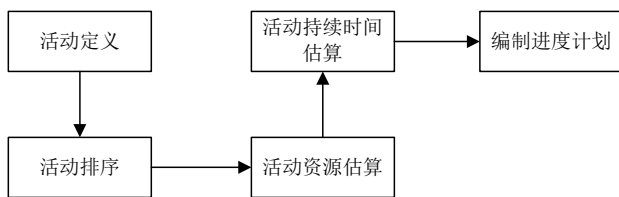


图 10-18 编制项目进度计划的步骤

活动的定义是将 WBS 底层工作包做进一步的拆分。WBS 的分解是面向工作产品的，而活动的拆分是将工作产品拆分为若干个任务，这些任务就是为了完成该工作产品所必需的行动。活动定义的主要成果就是任务的清单，它记录了项目的所有任务，任务清单中所有任务的产出物应该等于 WBS 底层所有工作包之和。

活动的排序是指识别与记载任务之间的逻辑关系。任务与任务之间有着以下 4 种逻辑关系：

- ① 完成～开始：这是最常见的一种逻辑关系，某个任务完成后下一个任务才会开始。

② 开始～完成：这是最少见的一种逻辑关系，常出现在“接力”类型的活动中。例如：某个哨兵站岗，除非有人来接替他，否则他必须一直站下去。也就是说哨兵是否下班取决于下一个哨兵的工作是否开始，这就是一个典型的“开始～完成”的关系。

③ 完成～完成：这种逻辑关系常出现在有并发关系的任务之间，例如：开发人员要完成单元测试和代码，只有这两项任务都完成了，他的工作才算完成。

④ 开始～开始：这种逻辑关系常出现在有并发关系的任务之间，例如：《软件需求说明书》完成后，软件设计人员和软件测试人员就可以同时开始“软件设计工作”和“编写系统测试用例”的工作。

在活动的排序中还要考虑以下 3 种任务与任务之间的依存关系：

① 强制性依赖是指所从事的工作中固有的依存关系，又称为硬逻辑。

② 可选择的依赖是项目管理团队所规定的依赖关系，通常是一些经验或最佳实践，也称为优先选用逻辑关系、软逻辑。

③ 外部依赖是指项目任务与项目外部活动之间的依赖关系。例如：设备是否到位。

活动资源估算和对活动持续时间进行估算都是针对任务清单中每个任务进行的判断，此时估算的效果会比之前利用历史数据对项目进行整体估算更加准确。所以对项目整体进行的判断被称为“估算”，针对每个任务进行单独判断并采用自下而上的方式累计得到的结果被称为“预算”，“预算”比“估算”更加准确。

编制进度计划的目的是按照任务间的关系将任务进行逻辑排列，并依据每个任务持续的时间可以推算出项目结束的日期。建立项目进度计划的完整流程如图 10-19 所示。

1. 使用单代号网络图法制订项目进度计划

单代号网络图法又称为优先顺序图法，它是一种使用方格表示任务，使用箭线表示任务之间逻辑关系的一种进度网络图绘制的方法，它的英文简称是 AON (Activity on Node)。箭线表示的逻辑关系有以下 4 种：

- 完成～开始
- 开始～完成
- 完成～完成
- 开始～开始

单代号网络图绘制简便，逻辑关系明确，没有虚活动，是近年来国内外常用的一种制订进度计划的技术，像微软的 Microsoft Project 就是采用单代号网络图法来绘制进度计划的。它的任务图例表现方式如图 10-20 所示。

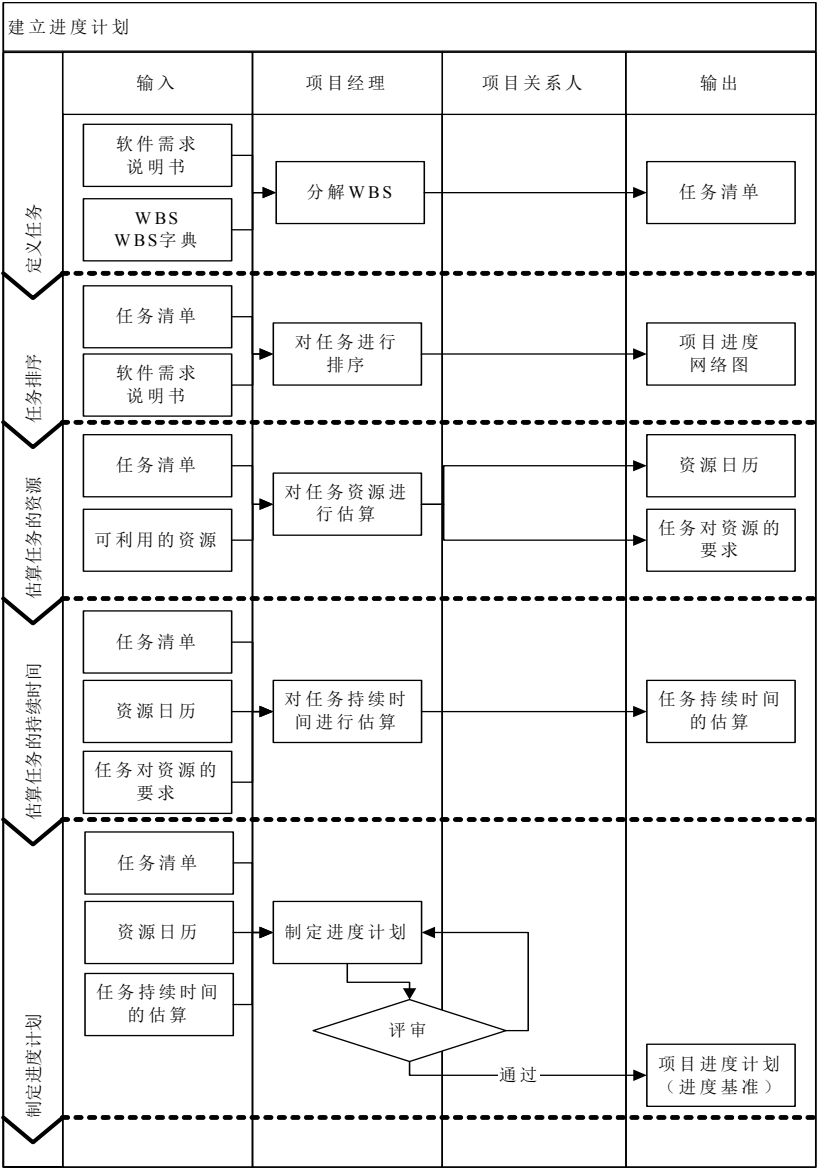


图 10-19 建立项目进度计划的流程

在绘制单代号网络图时需要注意以下事项：

- 在单代号网络图中每个任务必须有唯一的编号或命名
- 在绘制时禁止出现循环的回路

- 严禁出现双向箭头或无箭头的连线
- 严禁出现没有箭尾节点的箭线和没有箭头节点的箭线
- 单代号网络图中只应有一个起点和一个终点
- 箭线不能交叉

最早开始时间 ES	持续时间 D	最早完成时间 EF
任务名称		
最晚开始时间 LS		最晚完成时间 LF

图 10-20 单代号网络图例

单代号网络图绘制步骤如图 10-21 所示：

- STEP 01
- 先对任务清单上的每个任务估算持续时间（D）。
- STEP 02
- 确定起点的最早开始时间（ES）。
- STEP 03
- 从起点开始，采用正推法计算前置任务的最早完成时间(EF)=最早开始时间(ES)+持续时间(D)-1。
- STEP 04
- 从起点开始，计算后置任务最早开始的时间（ES）。
- STEP 05
- 确定终点的最晚完成时间（LF）。
- STEP 06
- 逆推法计算每个任务的最晚开始时间(LS)=最晚完成时间(LF)-持续时间(D)+1。
- STEP 07
- 从终点开始计算最晚完成时间（LF），顺着箭线的反方向逐个进行计算。

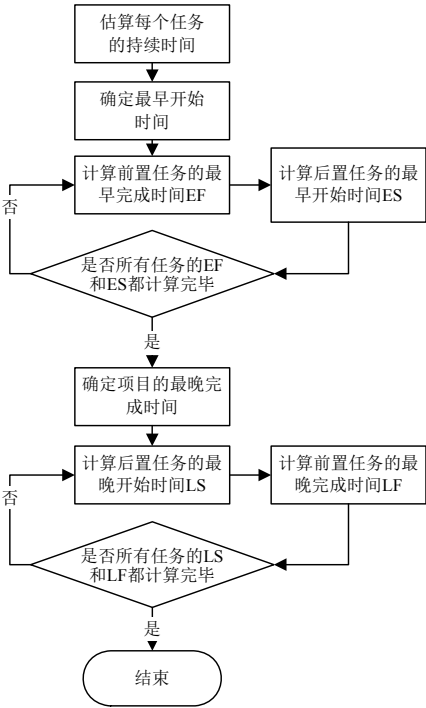


图 10-21 单代号网络图绘制步骤

下面我们用一个具体的范例进行详细说明。

某软件项目的任务清单如表 10-36 所示,客户给出的最早开工时间是 2009 年 2 月 2 日,周六和周日是休息日。

表 10-36 范例的任务清单

任务编号	任务名称	持续时间	逻辑关系
1	任务 1	3	起点, 任务 1 完成后任务 2 和任务 3 才能开始
2	任务 2	4	任务 2 与任务 4 是开始~开始的逻辑关系
3	任务 3	3	持续时间是 3 天, 与任务 5 是开始~开始的逻辑关系
4	任务 4	2	任务 2 完成后任务 4 才能开始
5	任务 5	5	任务 3 完成后任务 5 才能开始
6	任务 6	4	终点, 任务 4 和任务 5 都完成后, 任务 6 才能开始

STEP 01 按照逻辑关系绘制网络图。

按照客户给出的最早开始时间, 以及各个任务的属性和逻辑关系, 如图 10-22 所示, 画出单代号网络图, 并在图中标出每个任务的持续时间和项目的最早开始时间。

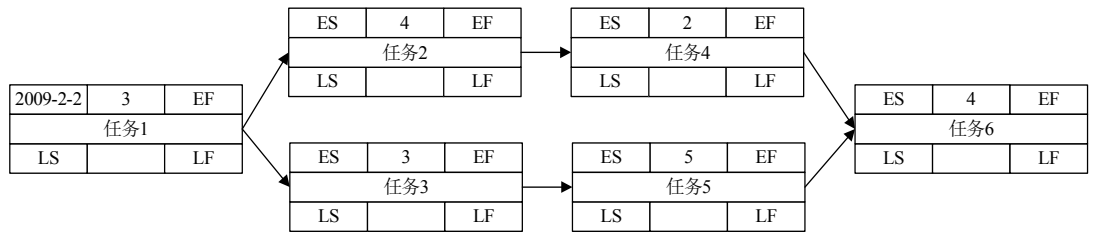


图 10-22 单代号网络图第 1 步

STEP 02 正推法计算最早完成时间 (EF)。

从起点开始依次计算每个任务的最早完成时间。

在完成一个任务 EF 的计算后就要按照 STEP 03 的要求完成它的所有后置任务的最早开始时间 (ES), 然后才能计算后置任务的 EF。最早完成时间 (EF) 的计算公式如下所示:

$$EF = \text{最早开始时间}(ES) + \text{持续时间}(D) - 1$$

STEP 03 计算最早开始时间 (ES)。

按照箭线的顺序, 当前置任务的最早完成时间 (EF) 计算完毕后, 就可以计算该任务的最早开始时间 (ES), 它的计算公式如下所示:

$$ES = \text{前置任务的最早完成时间}(EF) + 1$$



注意

如果该任务与它之前的任务存在多个“完成~开始”的逻辑关系，那么 EF 应该取它前置任务中最晚的一个 EF（最早完成日期）。例如：任务 6 的前置任务是任务 4 和任务 5，那么任务 6 的最早开始时间就是任务 5 的最早完成时间+1。

最后判断所有任务的最早开始时间（ES）和最早完成时间（EF）是否都已经计算完毕，如果没有，则从 **STEP 02** 开始继续计算直到所有的 ES 和 EF 都计算完成。终点任务的最早完成时间与最晚完成时间相同，完成后的网络图如 10-23 所示。

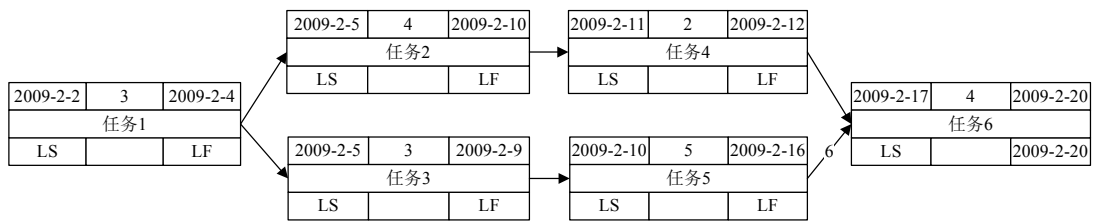


图 10-23 单代号网络图 **STEP 02** 与 **STEP 03**

STEP 04 逆推法计算最晚开始时间（LS）。

后置任务的最晚开始(LS)=最晚完成时间(LF)-持续时间(D)+1

从终点开始，当后置任务的最晚开始时间计算出来后，就要紧接着计算它前置任务的最晚完成时间，计算的方法参见 **STEP 05**。

STEP 05 计算最晚完成时间(LF)。

后置任务的最晚开始时间计算出来后，就要立即开始计算其前置任务的最晚完成时间，它的计算公式如下：

LF=后置任务的最晚开始时间(LS)-1



注意

如果该任务与它之前的任务存在多个“完成~开始”的逻辑关系，那么 LF 应该取它后置任务中最早的一个最晚开始时间。例如：任务 1 的后置任务是任务 2 和任务 3，那么任务 1 的最晚完成时间就是任务 3 的最晚开始时间-1。

最后判断所有任务的最晚开始时间和最晚完成时间是否都已经计算完毕，如果没有，则从 **STEP 04** 开始继续计算直到所有的 LS 和 LF 都计算完成。起点任务的最早开始时间与最晚开始时间相同，完成后的网络图如 10-24 所示。

2. 使用关键路径法对项目进行管理

关键路径法（Critical Path Method）是由雷明顿-兰德公司（Remington-Rand）的 JE 克里（JE Kelly）和杜邦公司的 MR 沃尔克（MR Walker）在 1957 年提出的。它经常与单代

号网络图一起使用，在学习关键路径前先了解 2 个术语：

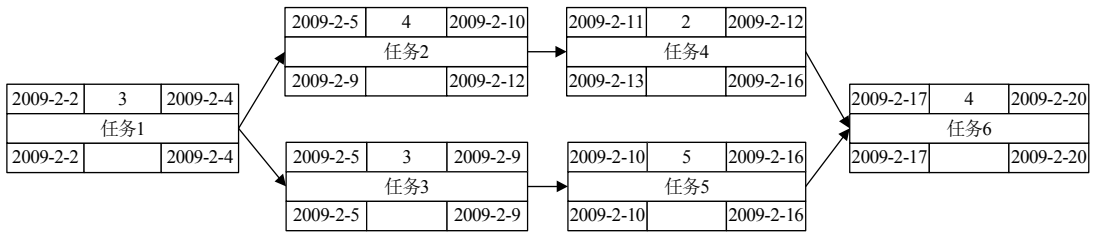


图 10-24 单代号网络图 STEP 04 与 STEP 05

■ 总时差

一个任务最早完成时间与最晚完成时间的差，称为总时差。总时差是不影响整个工期的可用来机动的时间，是所有任务共享的。它的计算公式如下所示：

某任务的总时差=最晚完成时间(LF)-最早完成时间(EF)

■ 自由时差

自由时差是单个活动的可以自由机动的时间。如果某个任务延期但也不会影响后置任务最早开始时间的时差，通常在路径汇合处才会有自由时差。它的计算公式如下所示：

某任务自由时差 = 后置任务的最早开始时间(ES)-本任务的最早完成时间(EF)

关键路径法是沿着网络图进行正向和反向分析，它具有以下特点：

- 总时差为 0 的路径就是关键路径
- 关键路径的持续时间最长
- 关键路径终点任务的完成时间就是项目的结束时间
- 关键路径上所有任务持续时间的和就是项目的周期
- 其中任何一个活动的延迟都会导致整个项目完工时间的延迟
- 压缩关键路径上的时间可以缩短项目的周期，而非关键路径则不可以
- 网络图中某个任务持续时间的改变可能会导致关键路径发生变化
- 如果网络图中存在多于一条的关键路径，那么项目存在风险
- 关键路径上的任务称为关键任务，关键任务具有以下特点：
 - 总时差为 0
 - 最早开始时间等于最晚开始时间，最早完成时间等于最晚完成时间
 - 自由时差为 0

下面我们用一个具体的范例进行详细说明。

以图 10-24 所示，计算单代号网络图中每个任务的总时差和自由时差，结果如表 10-37 所示。

表 10-37 计算范例的总时差和自由时差

任务名称	总时差=LF-EF	自由时差=后置 ES- EF
任务 1	0	0
任务 2	1	0
任务 3	0	0
任务 4	1	2
任务 5	0	0
任务 6	0	0

该范例的关键路径如图 10-25 所示。

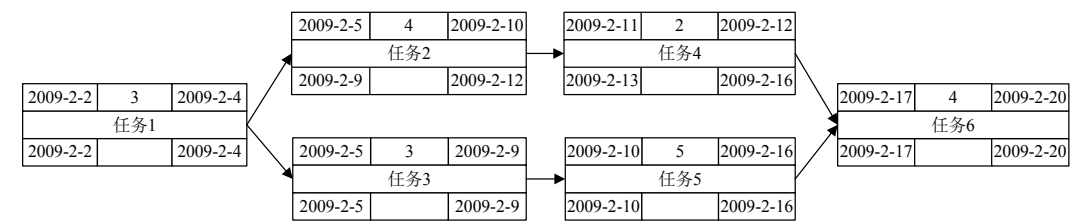


图 10-25 范例的关键路径

通常一个进度计划完成后总是不能满足项目的约束或领导的期望，因此可以采用以下方式对进度计划进行优化。

- 时间优化法
 - 采用新的技术、工具的方法来缩短关键路径上任务的周期
 - 采用快速跟进的方法将关键路径上的某些关键任务并行,但这样会增加项目的风险
 - 对关键任务进行加班,但这样会增加项目成本，有时也会造成关键路径的改变
- 资源优化法
 - 优先在关键路径上安排关键资源
 - 充分利用非关键路径的总时差和自由时差，将资源调整到关键路径上

为了方便对单代号网络图和关键路径法进行记忆，特对其核心内容进行了以下总结。

早开早完正向推，晚开晚完逆向算；
总差只在本身计，自差只存交汇时；
总差为 0 自差 0，终点总差同自差；
路径汇总算早开，迟的早完加一天；
路径分支算晚完，早的晚开减一天；

3. 使用 PERT 计划评审技术来制订项目进度计划

PERT（Program Evaluation and Review Technique，计划评审技术）最早是由美国海军

在研制北极星导弹时发展起来。

PERT 与单代号网络图也有相似的地方，它也需要对任务进行逻辑排序，并且也需要计算关键路径。

PERT 与单代号网络图法不同之处在于单代号网络图对任务持续时间的估算采用的是 1 个数值的方法。而 PERT 对任务持续时间采用了“乐观、正常和悲观”3 个时间点一同估算的方法。通过 PERT 除了可以计算出项目的完工时间和关键路径，还假设了任务持续时间和项目完成时间是随机的并且遵循着某种概率分布。

如图 10-26 所示，这是一个项目完工的正态分布图，横坐标代表了项目完工的期望工期；纵坐标代表了发生的可能性，曲线内的面积代表了项目完工的概率分布。

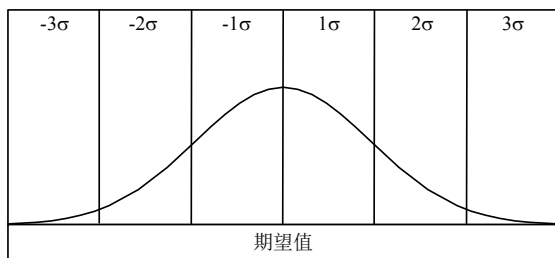


图 10-26 项目完工概率

由图 10-26 可以看出：

- 当项目经理完成一个项目进度计划时，在正态分布图中该项目按照该计划时间完成的概率只有 50%。
- 以期望值为中心， $\pm 1\sigma$ 内的累积概率分布之和为 68%。
- 以期望值为中心， $\pm 2\sigma$ 内的累积概率分布之和为 95%。
- 以期望值为中心， $\pm 3\sigma$ 内的累积概率分布之和为 99%。

PERT 的计算公式如下所示：

$$\text{某个任务期望完成时间 } t_i = \frac{\text{乐观值} + 4 \times \text{正常值} + \text{悲观值}}{6}$$

$$\text{项目完工的期望值 } T = \sum \text{各个任务期望完成的时间}$$

$$\text{某个任务的方差 } \sigma_i^2 = \left(\frac{\text{悲观值} - \text{乐观值}}{6} \right)^2$$

$$\text{项目的方差 } \sigma^2 = \sum \text{各个任务的方差}$$

$$\text{项目的标准差 } \sigma = \sqrt{\text{项目的方差}}$$

$$\text{任务的标准差 } \sigma_i = \frac{\text{悲观值} - \text{乐观值}}{6}$$

下面我们用一个具体的范例进行详细说明。

某软件项目的任务清单如表 10-38 所示,客户给出的最早开工时间是 2009 年 2 月 2 日,周六和周日是休息日。

表 10-38 PERT 范例的任务清单

任务编号	任务名称	乐观估算	正常估算	悲观估算	逻辑关系
1	任务 1	4	5	12	起点, 任务 1 完成后任务 2 和任务 3 才能开始
2	任务 2	1	8	9	任务 2 与任务 3 是开始~开始的逻辑关系
3	任务 3	8	10	12	
4	任务 4	4	5	7	任务 2 完成后任务 4 才能开始
5	任务 5	6	8	10	任务 3 和任务 4 都完成后, 任务 5 才开始

STEP 01 计算项目期望值并绘出网络图。

依据客户需求和 PERT 计算公式,对范例任务清单中的任务期望值进行计算,结果如表 10-39 所示,网络图如图 10-27 所示。

表 10-39 任务的期望值

任务编号	任务名称	期望值 (天)
1	任务 1	$(4+4\times 5+12)/6=6$
2	任务 2	$(1+4\times 8+9)/6=7$
3	任务 3	$(8+4\times 10+12)/6=10$
4	任务 4	$(3+4\times 5+13)/6=6$
5	任务 5	$(6+4\times 8+10)/6=8$

项目期望的工期是: 37 天

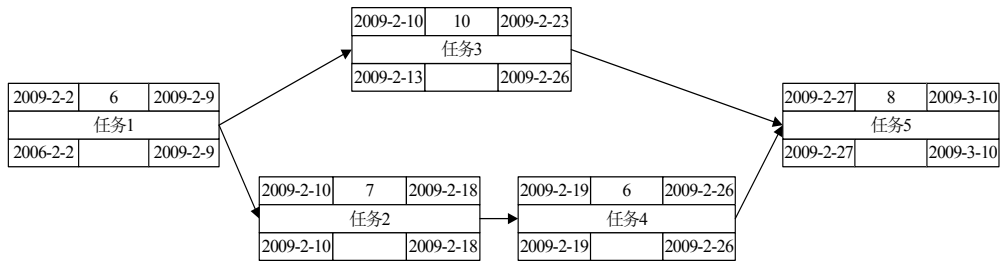


图 10-27 PERT 范例网络图

STEP 02 计算每个任务的标准差。

按照客户的任务清单即可算出每个任务的方差,从而计算出项目的方差和标准差,如表 10-40 所示。

表 10-40 任务的标准差

任务编号	任务名称	任务的方差
1	任务 1	$((12-4)/6)^2=1.7777$
2	任务 2	$((9-1)/6)^2=1.7777$
3	任务 3	$((12-8)/6)^2=0.4444$
4	任务 4	$((13-3)/6)^2=2.7777$
5	任务 5	$((10-6)/6)^2=0.4444$
项目的方差：7.22		
项目的标准差：2.68		

STEP 03 画出项目正态分布图。

根据表 10-40 算出的项目标准差和表 10-39 算出的项目工期，可以画出该项目的正态分布，如图 10-28 所示。

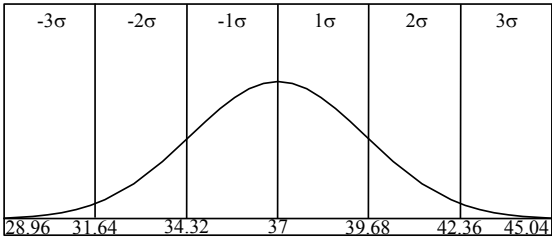


图 10-28 项目正态分布图

项目管理人员从图 10-28 中可以得到以下信息：

- 项目 37 天内完成的概率是 50%。
- 项目在 ±1 σ 范围内：34.32～39.68 天完成的概率是 68%。
- 项目在 ±2 σ 范围内：31.64～42.36 天完成的概率是 95%。
- 项目在 ±3 σ 范围内：28.96～45.04 天完成的概率是 99%。
- 如果客户要求 在 34 天内完成项目，那么概率大约是 50%-(68%/2)=16%。

10.2.4 建立项目费用预算

在制订项目计划第一步对项目进行整体估算时就已经完成对项目费用的估算，项目费用的估算是对项目所需资源成本的大致估计。项目费用的预算是将单个任务或工作包的费用进行汇总，以建立衡量项目总体绩效情况费用基准。它是在项目范围基准 WBS、进度计划中的任务清单，以及任务对资源要求的基础上进行的详细计算，其流程如图 10-29 所示。

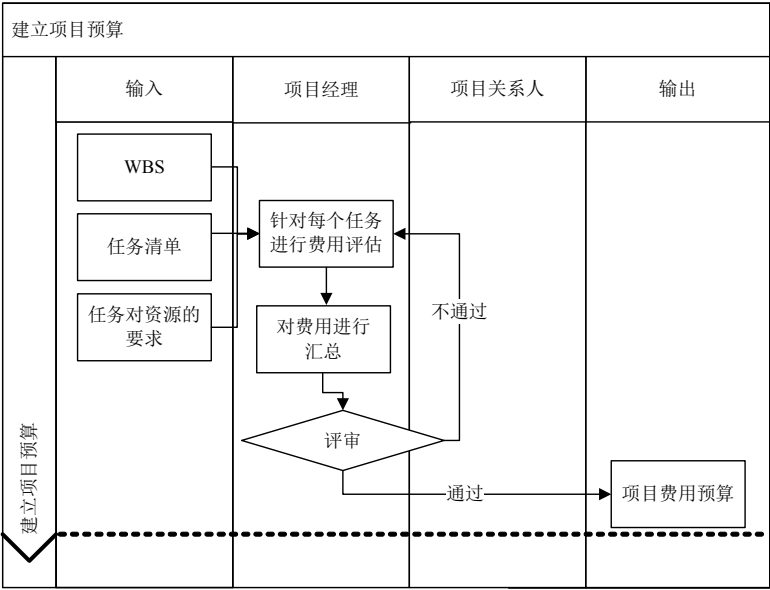


图 10-29 建立费用预算流程

在项目进行的过程中，成本的实际花费累计起来会呈现出一条“S”型的曲线，如图 10-30 所示，这也是对项目费用进行度量时常用的指示器。

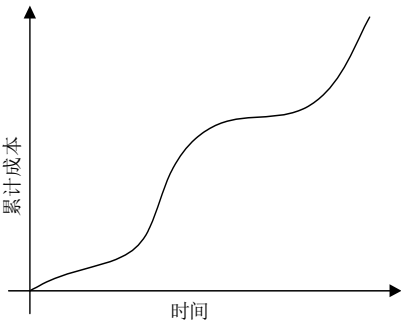


图 10-30 累计成本 S 型曲线

1. 利用净现值对项目进行决策

净现值（Net Present Value）是因投资所产生的对未来现金流的折现值与投资成本之间的差值。项目管理人员可以利用净现值来对项目进行决策，净现值越大，投资的收益就越好。

净现值的原理是假设项目估计的收益在年末肯定可以实现，然后将项目初始的投资按照一定折现率进行计算，当净现值大于零时，除了偿还本息外还会有剩余；当净现值小于

零时，该项目收益不足以偿还本息。净现值的计算公式如下所示：

项目的净现值(NPV)= $\sum_{t=0}^n ((CI - CO)_t / (1 + i)^t)$

其中，*i* 是折现率，CI 是现金流入量，CO 是现金流出量，(CI-CO)*t* 是第 *t* 年的利润。

● 范例 1

某软件公司计划开发一个产品，假设折现率为 10%，项目的利润如表 10-41 所示，那么项目 2010 年当年的利润净现值是多少？

表 10-41 某项目 2009~2012 年利润估算表

年份	2009	2010	2011	2012
利润（万元）	100	110	140	180

2010 年的 NPV=110/(1+10%)²=90.91 万元

● 范例 2

某公司预计投入 200 万元研发一个产品，预计项目在 2012 年以前每年的利润如表 10-42 所示，那么该项目是否值得投资？

表 10-42 某项目 2009~2012 年投资及利润估算表

年份	2009（初始投资）	2010	2011	2012
利润（万元）	200	50	80	100

NPV=50×（1+10%）+80×（1+10%）²+100×（1+10%）³-200=284.9-200=84.9 万
该项目预计在 2012 年可以累计盈利 84.9 万，因此该项目是可行的。

2. 利用动态投资回收期对项目进行决策

当项目的估算或预算都完成时，项目管理人员可以利用动态投资回收期的方法对项目的可行性进行分析。

动态投资回收期是指在考虑了货币时间价值的情况下，以项目的净现金流抵偿项目初始投资所需要的全部时间，也就是从项目投资开始计算，截止到累计折现现金流等于 0 的时间。它的计算公式如下所示：

动态投资回收期=累计净现值开始出现正值的年份-1+截止到上一年累计净现值的绝对值/当年净现值

下面我们用一个具体的范例进行详细说明。

某软件公司 2009 年计划投入 1000 万人民币开发一套电子商务的产品，预计从 2010 年起每年实现产品销售收入 1600 万，每年各种成本花费预计 1000 万。现金流和净现值如表 10-43 所示，那么该软件产品的回收期是几年？

表 10-43 某项目的财务报表

年 份	2009	2010	2011	2012	2013
投资（万元）	1000	—	—	—	—
成本（万元）	—	1000	1000	1000	1000
收入（万元）	—	1600	1600	1600	1600
净现值	-950	430	400	380	340

因为-950+430+400+380>0，即累计的净现值在第 4 年（2012 年）的时候出现了正值，那么将第 4 年（2012）作为当前年，从项目投资开始一直到第 4 年的上一年（2011 年）净现值的绝对值是：

$$|-950+430+400|=120$$

根据上面的公式，本项目动态投资回收期=(4-1)+120/380=3.32 年

10.2.5 计划项目的其他内容

项目计划的核心内容是项目的进度、成本和资源，项目的其他部分也需要被计划。

1. 项目的风险

项目的风险来自于项目的不确定性，项目的风险会对项目带来积极或消极的影响，而且项目的风险会随着项目的进展而发生变化，对项目风险及其管理进行计划是非常重要的，详细内容可以参见本书第 8 章的内容。

2. 项目的数据管理

项目的所有数据可以分为电子数据和纸质的数据，电子数据会由配置管理工具对它进行控制，具体内容详见本书第 6 章的内容。对项目的纸质文档同样要建立相应的管理制度，因为纸质的文档通常都更为重要，例如项目的合同。对项目纸质文档的管理可以借鉴图书管理制订，要明确哪些角色可以查看哪些纸质文档，这些文档如何借阅以及如何归还。

3. 项目所需技能的管理

每个项目根据它的需求和设计都会对项目组成员的技能提出要求，这些要求必须被细化并明确，因为这些需求与项目组已掌握的技能进行对比后就可以得出项目团队的培训需求。

4. 项目团队培训需求的管理

培训对于软件技术人员的自我提高非常重要，也是软件企业未来发展的源动力。通过项目所需技能与项目团队已掌握的技能进行对比，就可以得到本次项目培训的需求。这个

工作通常与项目所需技能的管理一同进行规划。

5. 计划关键人员的参与

在制订 WBS 时就已经对每个 WBS 节点所需要的人力资源进行了识别。计划关键人员的参与主要是协调项目进度日历与人力资源日历之间的冲突，确保项目所需要的资源可以按时投入到项目中，必要时可以寻求高层领导的支持。

10.2.6 建立项目计划的基准

通过以上几节的内容，项目管理人员可以确定项目的范围，完成项目进度和成本的预算，以及对项目其他内容的计划，但这些内容只能称其为计划，而不是项目的基准。

这些计划在完成后必须与项目相关关系人一同进行评审，意见达成一致并获取到相关人员的承诺，最后将这些计划作为项目的基线进行保存和发布，这样才能将项目计划转变成项目的基准，其流程如图 10-31 所示。

10.3 软件计划的常见问题及案例分析

项目计划是项目实施的基准，是项目质量的依据，以下通过案例分析的形式来讲述项目计划制订过程中的一些常见问题。

10.3.1 假设是项目计划的根本条件

【案例】

某软件公司项目经理小刘正在准备制订项目计划，该项目需要招聘 1 名高级程序员。项目开始 5 周后招聘工作一直没有进展，项目第一个里程碑已经延后了 3 天。公司高层经理发现了项目进度的偏差，于是开始质疑项目经理的管理能力，小刘对此有口难辩。

【分析】

在咨询顾问对小刘的项目计划进行审查时，发现项目经理的 WBS 是正确的，进度计划的安排是合理的，而且项目团队成员都按时、保质地完成了每天的任务，只是项目计划中缺少了对该项目的“假设”。

项目的“假设”是项目计划存在的根本条件，它记录了项目计划的不确定性，是项目风险的来源。

在小刘的项目中应该假设“该项目所需要的高级程序员会按时到位”，那么当出现以上问题时就可以向公司高层进行解释，以证明自己的管理能力和项目计划没有问题。

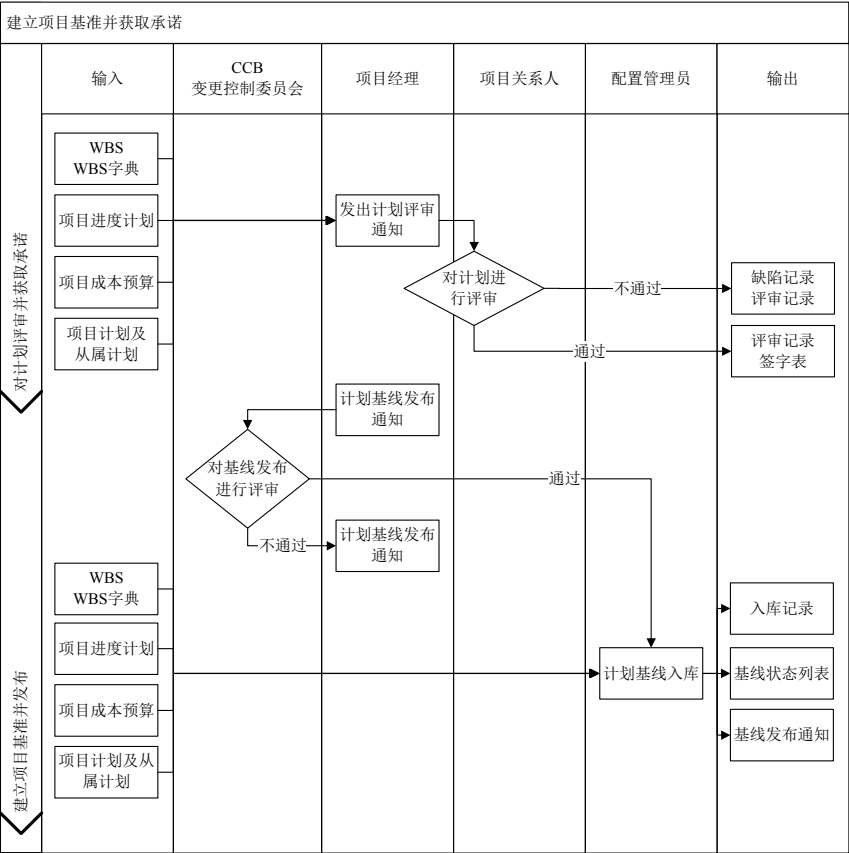


图 10-31 建立项目基准并获取承诺

10.3.2 关键路径的计算之案例 1

【案例】

如图 10-32 所示，计算该项目的关键路径。

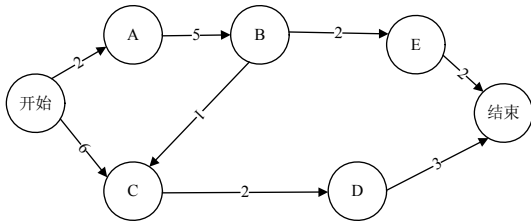


图 10-32 某项目的关键路径

【分析】

本项目有 3 条路径，它们的时间长度分别如下：

- 开始→A→B→E→结束：11 天
- 开始→C→D→结束：11 天
- 开始→A→B→C→D→结束：13 天

根据关键路径的定义，“开始→A→B→C→D→结束”为本项目的关键路径。

10.3.3 关键路径的计算之案例 2

【案例】

某项目的网络图如图 10-33 所示，现在客户的需求发生了变更，需要增加一个 6 天的任务 Z 在 A 和 B 之间，那么新的项目计划工期是多久？

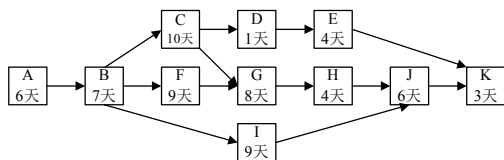


图 10-33 某项目网络图

【分析】

首先计算该项目的关键路径，然后判断新增加的任务是否会导致关键路径发生变化。

本项目在变更前关键路径是 A→B→C→G→H→J→K，工期为 44 天。当项目发生变更后，新的任务也是增加在关键路径上，因此变更后的关键路径是 A→Z→B→C→G→H→J→K，工期为 50 天。

10.4 小结

商场如战场，软件项目更是如此，项目的敌人通常都是无形的风险和隐患，是项目团队与自己意志力、耐力和执行力的对抗。要想管理好一个项目，一份合理、可行的计划是必不可少的。中国古代的大兵法家孙子就讲过这样的话：“夫未战而庙算胜者，得算多也，未战而庙算不胜者，得算少也。多算胜，少算不胜，而况于无算乎！吾以此观之，胜负见矣”。希望广大项目管理人员和质量管理人员以此共勉。

10.5 思考题

1. IFPUG 国际标准功能点估算法中将功能点分为了哪几类？
2. 项目的范围与产品的范围有什么不同？
3. 任务与任务之间的逻辑关系有几种？哪种是最常见的？

11

第 11 章

软件质量管理的监督手段——项目监控

在软件质量管理体系中有两种监控的手段和方法，一种是项目级的监控，它主要是由项目经理完成的，当然高层管理人员也可以进行参与；另外一种是第三方的独立监控，或者可以称之为“审计”，这项工作主要由直接向公司高层负责的软件质量保证人员来完成。本章将主要讲述项目管理级别所从事的项目监控。

通过对软件项目实施跟踪和监控，可以为各级管理人员提供足够的可视性。项目监控的目的是在项目计划发布后，通过定期检查项目计划中各种参数从而客观地了解项目的进展情况，并且在项目的进展情况与项目计划有较大偏差时，管理层及客户能够及时地做出调整，并采取有效措施，以使项目回到正常的轨道。

项目监控的方法非常简单，但关键的一点就是要将对项目跟踪和监控的内容和过程文档化，它是监控各项活动、沟通协调状态和采取纠正措施的基础。项目团队成员和其他项目关系人定期向项目经理提交周报或月报，项目经理对其内容进行汇总，然后定期召开项目例会或里程碑会议。项目经理通过对项目进展中的各项实际数据进行收集和度量，然后通过项目周报或月报，以及项目度量报告来客观反映项目的真实进展情况，及时发现各种偏差并采取纠正行动，对即将发生的风险进行预防。最后项目经理将这些报告分发给公司的各级管理人员和项目的关系人，如果客户要求，还可以将其呈交给客户，目的是使大家都能够及时地了解项目的真实进展状况。这里需要提醒广大项目管理人员，项目监控的对象是项目计划中的各种参数或属性，而不是每个项目团队成员。

11.1 软件监控管理流程及最佳实践

项目计划和项目监控主要是由项目经理进行管理，有人更形象地比喻项目计划和项目监控是项目经理的左右手。因此，项目计划和监控对项目的成败具有非常重要的意义。项目计划与项目监控之间的关系是密不可分、相辅相成的，如果没有项目计划，那么就谈不上项目监控，如果没有项目监控，则项目的计划便不可能落到实处。因此，项目计划是项目跟踪和监控的基础，项目计划中的各项参数执行的情况是项目监控评判的指标，它包含工作产品、工作任务、成本、工作量、进度等属性，而工作产品与工作任务的属性又可分为项目规模、复杂度等。软件项目监控的流程如图 11-1 所示。

11.1.1 监控项目的主要参数

软件项目监控主要是对项目的进度、成本、工作成果的质量进行的监控。

1. 对项目进度的监控

对项目进度监控的主要目的是评估项目实际进展情况是否与计划发生了偏差，如果实际进度发生了大幅提前，这并不一定就是好事，也许是因为漏掉了某些功能没有实现。如果实际进度发生延迟，那就更应该引起警惕，查找原因并及时采取纠正措施。通常进度延迟可以采用赶工、快速跟进、变更原始计划或缩小项目范围等方式使项目实际情况与计划相吻合。

在实际工作中对项目进度进行监控时，主要关注与项目进度计划中每个任务的实际开始时间、实际完成时间和实际花费的工作量，以及项目里程碑的具体日期。项目管理人员通过以下步骤即可完成对项目进度的监控：

STEP 01 通过对里程碑时间的计划值和实际值之间的对比，就可以知道项目是否出现阶段性的进度偏差。

STEP 02 当发生阶段性进度偏差时，可以通过对该阶段内每个任务实际开始时间、实际完成时间、计划开始时间和计划完成时间的对比找到进度偏差发生的原因。

STEP 03 及时制订措施来纠正偏差，常见的方法有赶工或快速跟进。但是赶工会增加项目成本，快速跟进是调整某些任务之间的关系，使它尽量并行前进，这样会给项目带来风险。

STEP 04 定期检查纠正措施是否发挥了应有的效果。

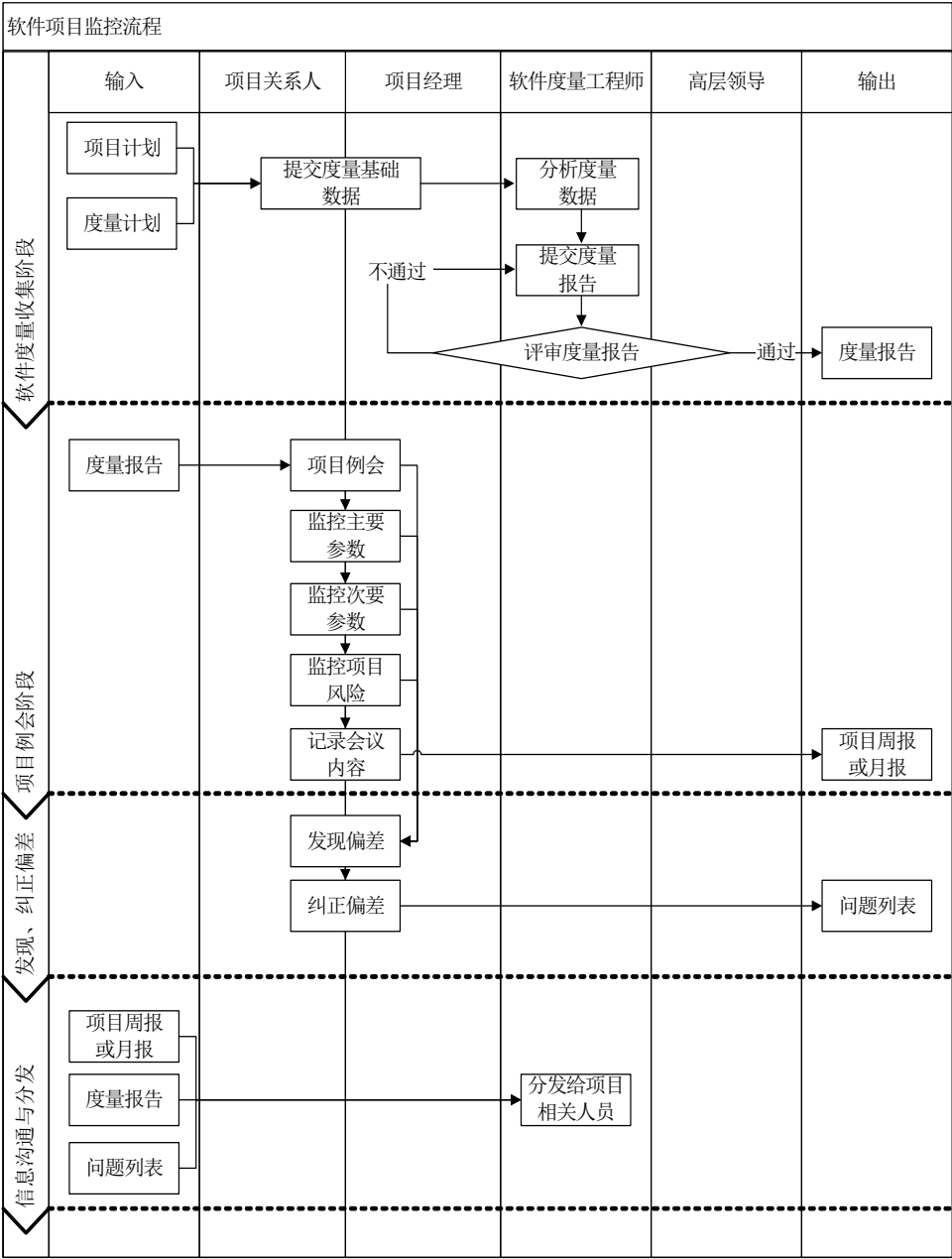


图 11-1 软件项目监控流程

2. 对项目成本的监控

对项目成本监控的目的是将项目实际花费控制在项目预算内。同样项目开支的大幅缩减也并不一定就是好事，也许是因为漏掉了某些功能没有实现。当项目开支出现大幅赤字时，就应该及时查找原因并采取纠正措施。通常项目成本的超支与项目范围的增加和进度的延迟有关，项目范围的增加肯定会导致项目需要投入更多的资源，项目进度的延迟通常采用赶工的方式来纠正进度偏差，但是赶工会造成项目成本的额外开支，因为项目组需要支付加班费、加班餐费和加班交通费。项目管理人员通过以下步骤即可完成对项目成本的监控：

- STEP 01** 以里程碑为标记统计该阶段成本花费的计划值和实际值，并将它们进行对比，就可以得到项目是否出现阶段性的成本偏差。
- STEP 02** 当发生阶段性成本偏差时，可以通过对该阶段内每个任务实际开支和计划开支进行对比找到成本偏差发生的原因。
- STEP 03** 及时制订措施来纠正偏差。
- STEP 04** 定期检查纠正措施是否发挥了应有的效果。

3. 对项目质量的监控

对项目质量监控的目的是将项目的质量控制在项目预期内。如果一个软件公司经过一段时间的软件度量管理，就会积累下大量的度量数据，通过对不同类型项目的分析，可以得到每种类型项目各个阶段缺陷出现的频率，因此对项目质量的跟踪和监控就可以通过控制图来完成。如图 11-2 所示，某软件公司在编码开发阶段出现的缺陷个数通常在 180~360 个之间，因此该公司某项目的产品质量处于可控范围内。

当项目质量出现失控时，就应该及时开展质量回溯的工作，查找原因并采取纠正措施。通常软件质量出现问题的常见和根本原因都是项目的周期短，或者需求变更频繁等。项目管理人员通过以下步骤即可完成对项目质量的监控：

- STEP 01** 以里程碑为标记统计该阶段缺陷发生的频率，也可以将缺陷按照种类或严重程度级别的级别划分后再进行统计。

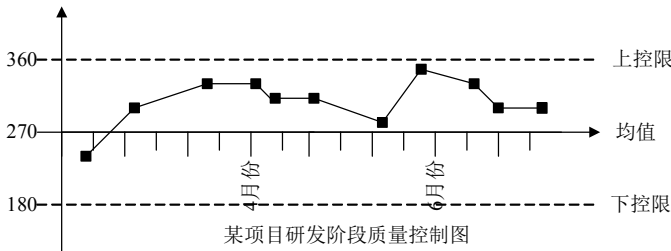


图 11-2 质量跟踪控制图

STEP 02 参考公司级别的历史度量数据中有关产品质量部分的度量目标，找出控制图的上控限、下控限和均值。

STEP 03 定期将项目实际的质量情况在质量跟踪控制图中标记，以判断当前项目是否发生了质量的偏差。

STEP 04 如果发生偏差就应该及时制订措施来对其进行纠正。

STEP 05 定期检查纠正措施是否发挥了应有的效果。

11.1.2 监控项目的次要参数

软件项目除了成本、进度和质量外，还有很多次要的参数需要被监控。这些次要的参数是相对于时间、成本和质量而言的，但是这些次要参数对软件项目来说都是十分重要的，如果一个项目管理人员放松了对这些次要参数的监控，那么软件项目肯定会发生风险，严重的甚至会发生失控。在日常工作中软件项目管理人员必须对项目的工作产品或任务的属性进行监控，并对项目资源的使用情况、项目关系人知识技能的情况、项目的承诺及履行的情况、项目数据管理的情况、关键人员的参与情况进行跟踪。

1. 监控工作产品或工作任务的属性

对项目工作产品或工作任务属性的监控主要体现在工作产品或工作任务的规模或复杂度会发生变化。对项目规模进行跟踪通常是为了应对软件需求的变更，在必要时可以作为向客户方提出延长项目周期或增加项目经费的依据。对复杂度进行跟踪可以及时发现项目的风险，防止在项目计划时由于对项目了解程度不足而低估了某些任务或工作产品的难易程度。项目管理人员通过以下步骤即可完成对项目工作产品或工作任务属性的监控：

STEP 01 在每次项目发生变更时对项目的规模进行估算，记录每次变更的大小。当项目规模累计增加或减少一定比例时，就要向公司提出项目重大变更的申请，以便公司及时了解项目的状况，并向客户方提出合理的、适当的补偿。项目规模的变化往往是由于需求调研时没有掌握客户方的隐性需求造成的。

STEP 02 定期对项目工作产品或工作任务的复杂度进行跟踪，可以及时避免该工作产品或任务的所有人因为技术能力达不到要求，而出现未能按期并保质地完成该项工作的风险。技术复杂度的变化往往是由于一些需求的变化导致的，还有一些时候是因为设计工作不够细化造成的。

STEP 03 发现偏差后及时制订措施进行纠正。

STEP 04 定期检查纠正措施是否发挥了应有的效果。

STEP 05 对偏差的原因进行总结，避免在项目中重复发生。

2. 监控资源的提供与使用

软件项目的资源可以分为硬件资源和人力资源两大类。硬件资源就是大家常见的如主机和外围设备、网络、软件测试用的环境、软件开发或测试用的工具等资源。软件行业是一种脑力密集型的行业，人就必然成为该行业中必不可少的一种资源，例如：在 Microsoft Project 中人就作为一种资源进行管理。一个项目所需要的硬件资源如果数量或质量上达不到计划中的要求，很可能会导致某些任务不能按时开展，甚至会影响项目阶段性目标的实现。如果项目所需要的人力资源无法到位，或者所要求的资源在数量上出现偏差，那么该项目同样面临工期延误的风险。例如：某软件项目需要 5 个开发人员，但公司最终只能提供 4 个开发人员，那么项目经理唯一可以确保项目质量的方法就是延长项目的周期。

在软件项目中对项目硬件资源和人力资源的跟踪和监控是确保项目可以按期交付的重要环节，项目管理人员通过以下步骤对它进行监控：

STEP 01 定期对项目资源的提供情况进行跟踪和监控，确保公司或供应商可以及时提供项目必要的资源。项目经理可以通过项目资源计划或项目进度计划中定义的关键资源参与项目的时间为最终期限，提前一段时间对它提供的情况进行跟踪。

STEP 02 当资源到位后，就要按照项目进度计划中资源的分派情况进行跟踪，确保所需资源的合理使用，避免资源的浪费。

STEP 03 当发现所需资源不能按期到位时，就要及时进行风险识别，按照风险的预防和应对措施采取相应的行动。

STEP 04 定期检查纠正措施是否发挥了应有的效果。

3. 监控项目人员的知识与技能

软件项目的资源分为硬件资源和人力资源。硬件产品都有自己的性能参数，例如一台电脑性能的好坏可以通过 CPU 的频率、内存的大小和硬盘大小等参数进行衡量。人力资源同样有着自身衡量的标准，例如：工作年限、掌握了哪些技能、技能掌握的熟练程度等。软件公司都在强调“以人为本”，由此可见人力资源在软件行业中的重要性，因此对项目成员的知识与技能进行监控，可以确保项目按照既定的技术解决方案进行实现，项目管理人员通过以下步骤对它进行监控：

STEP 01 在项目计划中有对项目所需技能的分析、项目团队组织架构图、项目职责分派矩阵和项目团队成员介绍的内容和信息。

STEP 02 通过项目团队成员介绍与这些信息进行对比，可以发现项目所需技能与现有团队成员掌握的技能之间出现某些偏差。

STEP 03 当发现偏差时就应该及时采取纠正措施。对于项目人员知识和技能上的偏差可以通过尽早开展相应培训，或更换相应资源进行补救。

STEP 04 定期对项目人员参与培训的效果进行跟踪。

4. 监控承诺事项

当今的社会是建立在诚信基础上的，承诺是承认和诺言的体现，它的表现形式可以是口头上的也可以是书面上的。承诺不仅在软件项目中，更是在每个人的一生中都非常重要并带有某种深刻含义的行为，如果一个人对他自己的承诺都当作儿戏，那么他必将被周围的朋友和这个社会所抛弃。在软件项目中，每个项目关系人对项目行使权利、承担义务都离不开承诺，如果项目不能获得某些关键人员的承诺，或者承诺不能得到兑现，那么项目将会出现风险。项目管理人员通过以下步骤对承诺进行监控：

STEP 01 获得承诺的途径就是召开评审会议，让项目关键人员对评审的内容进行确认。

STEP 02 跟踪和监控承诺的最好方法就是将承诺文档化，通过对评审报告的签字确认来行使每个团队成员自己的话语权。

STEP 03 项目经理要定期回顾每个确认过程中关键人员是否给出了承诺，如果某个确认未能获得他们的承诺，那么项目就应该延后其他工作，先使相关人员的意见达成一致，获得承诺后再开展其他工作，否则就会产生项目不确定性，项目的风险也随之而来。例如：对软件需求的确认时，客户方不能给出该需求是否是他们所需要的承诺，在这种情况下继续开展项目的后续工作将会给项目带来非常大的风险。

STEP 04 当发现承诺的事项出现偏差时，首先应该将其识别成为一种风险，及时制订预防或应对的措施。

STEP 05 执行风险的预防或应对措施，正确获得相关人员的承诺。对无法获得承诺的情况应该及时通知公司高层领导，并采取措施将其影响降为最低。

5. 监控数据管理

软件项目的数据分为电子数据和纸质文档数据两大类，电子数据的跟踪和监控可以通过软件配置工具来实现，这里所讲的对数据管理的监控是专门针对纸质文档的跟踪和监控。一个项目的纸质文档有很多，特别是一些重要的承诺都是通过纸质文档来实现的，如何确保这些纸质文档不被没必要的人所查看，确保这些纸质文档被借出后都能如期归还，就应该制订一套类似于图书借阅制度的管理条款。项目管理人员通过以下步骤对承诺进行监控：

STEP 01 查看纸质文档借阅记录中的借阅人是否符合项目配置管理计划中所分配的权限。

STEP 02 检查纸质文档借阅记录中借阅人是否按计划归还相应文件。

STEP 03 如果发生偏差就需要及时与借阅人进行沟通，确保纸质文档的安全。

6. 监控关键人员的参与

在软件项目中关键人员是否能够按照计划如期参与到项目中，这是项目成功的关键。

如果发生关键人员无法按时参与项目的活动，通常会导致项目延期或出现产品质量问题。例如：需求评审需要客户、需求调研人员、项目经理和相关项目关系人的参与，如果该需求调研人员此时正在外地出差，那么该评审任务就只能延迟。如果需求评审时客户无法参与确认，那么该项目所提交的产品很可能无法满足客户的要求。项目管理人员通过以下步骤对关键人员的参与情况进行监控：

- STEP 01** 按照项目进度计划、评审会议的签字情况定期进行检查，从而可以发现偏差。
- STEP 02** 对发现的偏差进行记录，并识别其是否会成为风险。
- STEP 03** 如果是风险就需要制订预防或应对措施，并按照措施及时进行纠正。
- STEP 04** 定期回顾纠正措施的效果是否达到预期。

7. 监控项目的风险

项目的风险体现了一种因果的关系，它来源于项目的不确定性，项目风险随着项目的进展也会发生动态变化，因此对项目风险的跟踪是项目监控的重要组成部分，项目管理人员通过以下步骤对项目的风险进行监控，风险监控示意图如图 11-3 所示。

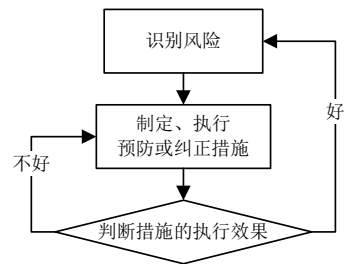


图 11-3 风险监控示意图

STEP 01 定期对风险进行重新识别，新识别出来的风险要及时制订风险的预防和应对措施，并且要明确这些措施的责任人。此外还要根据现阶段项目的情况对风险的可能性、严重性、阈值、优先级别等属性进行定义。

STEP 02 定期判断风险在当前阶段的状态。

STEP 03 定期判断风险在当前阶段发生的可能性和严重性。

重性。

STEP 04 定期判断风险的优先级别。

STEP 05 定期对风险预防或应对措施的执行效果进行评判，以判断是否需要对这些措施进行修正。

STEP 06 定期判断风险预防或应对措施执行后，是否会引发其他新的风险或者还有残留的风险。

11.1.3 监控项目的方法

软件监控的方法可以分为定期监控、阶段性监控和事件触发性监控三种。

① 定期监控是指项目组可以通过定期召开周例会、月例会的方式来对项目的各种参数进行跟踪和监控。

② 阶段性监控是以项目生命周期各阶段的里程碑为标记，通过里程碑的评审会议来对

项目的各种参数进行跟踪和监控。如果项目的里程碑会议与项目周例会或月例会的时间间隔较短，可以将项目周例会或月例会合并到项目里程碑会议中一起召开。

③ 事件触发性监控是指当项目发生重大变更时都需要对项目的某些参数进行跟踪和监控，例如当项目发生需求变更时，都需要对项目的规模变化情况进行跟踪和监控。

软件项目监控可以由项目经理、高层经理共同参与，但项目监控的主要责任人是项目经理。在一个软件公司里项目管理工作不仅仅是项目经理的工作，部分高层管理人员也要做其他层面上的项目管理工作，因此邀请高层管理人员参与项目会议是十分重要的，因为高层管理人员可以借此了解项目的情况，为项目经理提供资源和支持。

软件项目监控主要是以项目计划为依据，通过对项目参数的计划值与实际值之间的对比找到偏差，项目计划的主要参数项目成本与项目进度之间有着非常紧密的关联和影响关系，通常项目的延期都会导致项目成本的增加，项目成本的增加通常是为了确保项目不会延期。但项目成本和项目进度的测量单位并不统一，一个是货币的单位人民币，一个是时间的单位天、周或月。如果能有一种方法将项目成本与项目进度进行统一，那么将会降低项目监控的工作量，提高项目监控的准确性。

1996 年美国国防部创建了 EVMS（挣值管理系统），1998 年美国国防部和国家标准学会将“挣值管理系统”颁布为行业标准，同时美国航空航天局（NASA）、国税局（IRS）、联邦调查局（FBI）等机构开始使用这套系统，不久加拿大、英国、澳大利亚等国家也相继把“挣值管理系统”作为政府和工业界的标准。

挣值（Earned Value）的核心就是比较项目实施与计划期望之间成本与时间的差异，然后根据这些差异再对项目剩余的任务进行预测和调整。挣值分析的优点就是能够同时判断项目的成本和进度的执行情况，以预算和实际费用来衡量项目的进度，也就是说将项目成本和项目进度统一使用货币单位进行测量。在对公司以往的项目进行分析时，有些项目比计划提前完成，其中有的是因为提高了生产的质量从而避免返工而实现的，而有的只是因为客户缩小了项目范围而实现的；管理人员会发现如果只是单纯使用计划的时间进度与实际的项目进度做比较，或者只是通过对项目成本的预算和实际花费的经费进行比较，都不能全面反映项目本身的绩效，因为没有可比性，只有在完成相同的工作前提下对时间、成本的差异对比才是合理的。

就像在平时娱乐打牌时，牌的好坏直接影响了最终的结果，但不能完全反应玩家的真实能力。但桥牌却不太一样，它要求牌好的一方应该得到更多的积分，这样就给比赛提供了一个公平的环境。通过挣值来表示实际完成的任务所对应的预算成本，这就在计划和实际之间建立了一个桥梁，总得来说挣值是从以下几方面来分析问题的，EV 挣值示意图如图 11-4 所示。

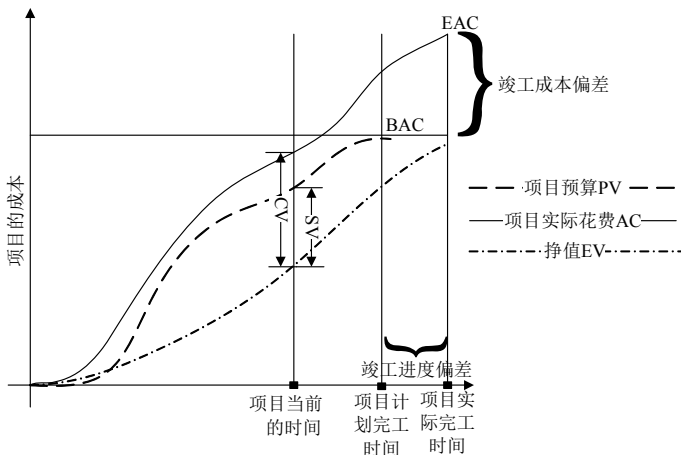


图 11-4 挣值示意图

① 在完成同样计划任务的情况下比较预算成本和实际花费之间的成本差异。换句话说，不管项目实际花费了多少，只比较计划中需要完成任务的价值。

② 充分考虑项目中任务的时间、资源、成本等多方面因素，统一使用成本指标来对每个项目任务进行衡量。

③ 在花费同样成本的情况下，将计划应该完成的任务与实际完成的任务进行比较，从而得到进度差异。

挣值分析中的 4 个关键指标如下所示：

- 计划值（Planned Value）简称 PV，在规定的时间内计划完成工作所需要的预算成本。
- 实际成本（Actual Cost）简称 AC，在规定时间内已完成工作的实际成本。
- 挣值（Earned Value）简称 EV，在规定时间内完成工作所对应的预算成本。
- 项目总预算（Budget at Completion）简称 BAC，项目计划中的总预算。

挣值分析中的两个偏差：成本偏差和进度偏差，它们都是以货币单位进行计算的。

$$\text{成本偏差(Cost Variance, CV)} = \text{EV} - \text{AC}$$

① 当 $CV > 0$ 时，表示实际花费少于预算，即项目成本有结余，如图 11-5 所示。

② 当 $CV = 0$ 时，表示项目完全按照计划执行。

③ 当 $CV < 0$ 时，表示项目执行情况不理想，实际的花费超过预算，如图 11-6 所示。

$$\text{进度偏差(Schedule Variance, SV)} = \text{EV} - \text{PV}$$

① 当 $SV > 0$ 时，表示进度提前，如图 11-7 所示。

② 当 $SV = 0$ 时，表示进度按期完成。

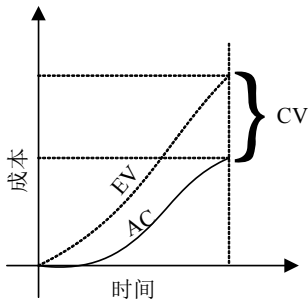


图 11-5 项目成本控制良好

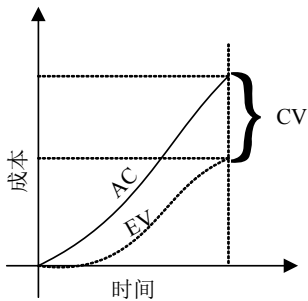


图 11-6 项目成本失控

③ 当 $SV < 0$ 时，表示进度延期，如图 11-8 所示。

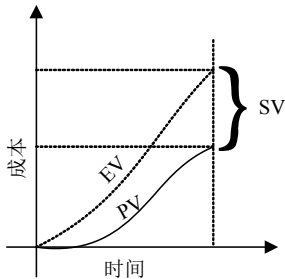


图 11-7 项目进度控制良好

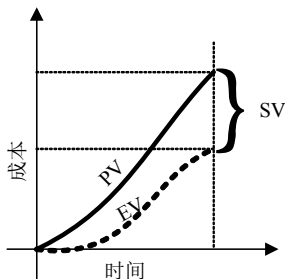


图 11-8 项目进度失控

挣值分析中的两个绩效指标为成本绩效指数与进度绩效指数。

在企业发展的过程中时常会发生这样的现象，在年底进行回顾的时候发现营业额比上一年有较大增长，但净利润却出现下滑，这是什么问题呢？其实就是项目的成本绩效不好造成的。挣值风险提供了两个工作绩效的指标供各级管理人员进行分析：CPI 是指每花费 1 元钱所产出的工作价值是多少；SPI 是指实际进度完成了计划进度的百分比。

成本绩效指数 $\text{Cost Performance Index(CPI)} = \text{EV} / \text{AC}$

- ① 当 $\text{CPI} > 1$ 时，表示实际费用低于预算费用，项目过程中的投资回报率（ROI）较为理想。
- ② 当 $\text{CPI} = 1$ 时，表示实际费用与预算费用正好吻合，项目过程中的投资回报率（ROI）符合预期。

③ 当 $\text{CPI} < 1$ 时，表示实际费用超出预算费用，项目过程中的投资回报率（ROI）较差。

进度绩效指数 $\text{Schedule Performance Index(SPI)} = \text{EV} / \text{PV}$

- ① 当 $\text{SPI} > 1$ 时，表示进度提前，即实际进度比计划快。
- ② 当 $\text{SPI} = 1$ 时，表示实际进度等于计划进度。
- ③ 当 $\text{SPI} < 1$ 时，表示进度延误，即实际进度比计划慢。

挣值分析中有两个完成指数：成本完成指数与任务完成指数。

任务完成指数(PC)=EV/BAC，任务完成指数代表完成总工作量的百分比。

成本完成指数(PS)=AC/BAC，成本完成指数代表已经消耗的成本占项目总预算的百分比。

挣值分析的一个重要功能就是根据项目当前实际进展情况来预测将来项目的发展趋势。项目管理人员可以通过项目完工预算（Estimate at Complete，EAC）来预测将来项目完成时所花费的总成本，其计算公式如表 11-1 所示。

表 11-1 EAC 项目完工预算

公 式	适用情况
$EAC=AC+(BAC-EV)/CPI$ 实际支出+经过 CPI 修正后的剩余项目预算	目前的偏差被视为典型的偏差，而且在今后会发生类似的情况
$EAC=AC+(BAC-EV)$ 实际支出+剩余工作的原始预算	目前的偏差被视为非典型的偏差，而且在今后不会发生类似的情况
$EAC=AC+ETC$ 实际支出+剩余工作的重新估算	<ul style="list-style-type: none">• 项目在过去的实施情况中表明原来所作的项目估算彻底过时，需要重新估算；• 项目变化较大，原来项目估算的环境和条件对现在来说都不适用

项目管理人员可以通过项目完工尚需的预算（Estimate To Complete，ETC）来预测将来项目完工时还需要投入多少成本，它的计算公式是 $ETC=EAC-AC$ 。

11.2 软件监控管理常见问题及案例分析

11.2.1 项目参数之间存在相互的影响和依赖

【案例】

某软件公司刚刚结束项目管理的培训，项目经理小李即将成为某企业网站项目的项目经理，由于该项目规模较大，公司高层领导特别提醒小李平时要注意对项目进行监控。项目监控时需要对很多项目的参数进行跟踪，如何更好地理解这些参数的含义是令小李头痛的事情。

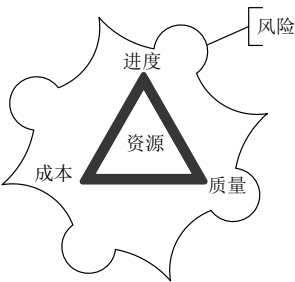


图 11-9 项目参数间关系示意图

【分析】

这天项目经理小李找到本次培训的讲师，希望可以找到一个更加简单、利于理解的方法来理解项目参数间的关系。张老师告诉小李，如图 11-9 所示：项目的进度、成本和质量是项目的三元素；项目是需要资源来完成的，“监控资源的提供与使用”重点关注的是资源的数量；“监控

项目人员的知识与技能”重点关注的是资源的能力；“监控承诺事项”、“监控关键人员的参与”、“监控数据管理”重点关注的是资源的行为；“监控工作产品或工作任务的属性”重点关注的是资源所承担的工作。也就是说除了项目三元素外就是项目的资源，那么四者在一起就会产生风险，因此还需要“监控项目的风险”。小李听后犹如醍醐灌顶，虽然项目的参数非常多，但是参数间有着如此的种种联系，这样再对它进行记忆就简单很多了。

11.2.2 挣值法在软件项目中的应用

【案例 1】

某公司人力资源管理系统需要增加一个功能，小王担任了该项目小组的组长。这次功能增强部分将会添加 10 个小的功能，经过估算需要两个软件工程师做 10 天，预计花费 10000 元，平均一个小功能 1000 元。项目进行到第 6 天时，实际完成了 4 个功能的工作量，项目实际支出了 5000 元。请问项目当前的绩效情况如何？

【分析】

项目总预算 $BAC=10000$ 元，即 $AC=5000$ 元。

项目预计 10 天完工，平均每天花费 $=10000/10=1000$ 元。项目进展到第 6 天时， $PV=6 \times 1000=6000$ 元。

本项目需要实现 10 个功能，平均每个功能 $=10000/10=1000$ 元。项目进展到第 6 天，实际完成了 4 个功能， $EV=4 \times 1000=4000$ 元。

$$CV=EV-AC=4000-5000=-1000$$

$$SV=EV-PV=4000-6000=-2000$$

$$CPI=EV/AC=4000/5000=0.8$$

$$SPI=EV/PV=4000/6000=0.67$$

假设当前的偏差是典型的，即未来还会发生的偏差，那么完工时的总估算 $(EAC)=AC+(BAC-EV)/CPI=5000+(10000-4000)/0.8=13750$ 元。

完工尚需的预算 $(ETC)=EAC-AC=13750-5000=8750$ 元。

本项目截止到第 6 天，成本方面超支了 1000 元，进度方面还有 2000 元的工作没有按期完成，以此状况进展下去，项目预计还需要支出 8750 元，项目的绩效如图 11-10 所示。

【案例 2】

某软件公司电子商务平台将进行第二期升级改造项目，项目经理小莫开始着手研究合同条款、确定项目范围、进行 WBS 分解、任务定义和排序、编写进度计划和资源计划、进行成本的估算和预算等，经过多次反复修改，项目计划终于得到公司的批准。该项目由 11 个任务组成，总工期为 120 天，预算花费 32 万元。其任务分解如图 11-11 所示。

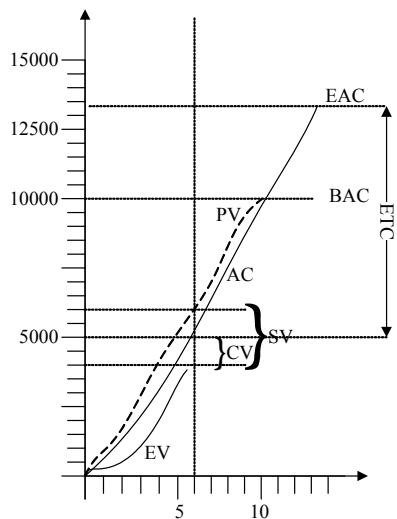


图 11-10 案例 1 项目的绩效情况

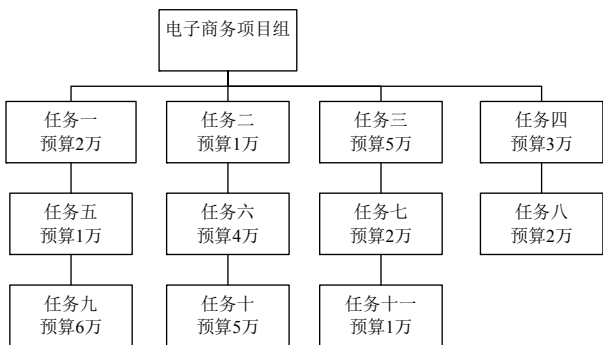


图 11-11 项目 WBS 分解图

当项目进展到第 2 周时，公司高层领导希望了解该项目的绩效情况，因此项目经理小莫做了一个进度跟踪甘特图，如图 11-12 所示。按计划两周应该完成任务一、任务二、任务三、任务五、任务六和任务八。但实际工作了两周后，除了任务八外其他计划内的任务均已完成，项目实际支出 13 万元。因此，小莫提交的报告中对项目绩效进行了如下总结：“项目进度有所落后，但项目成本控制良好”。

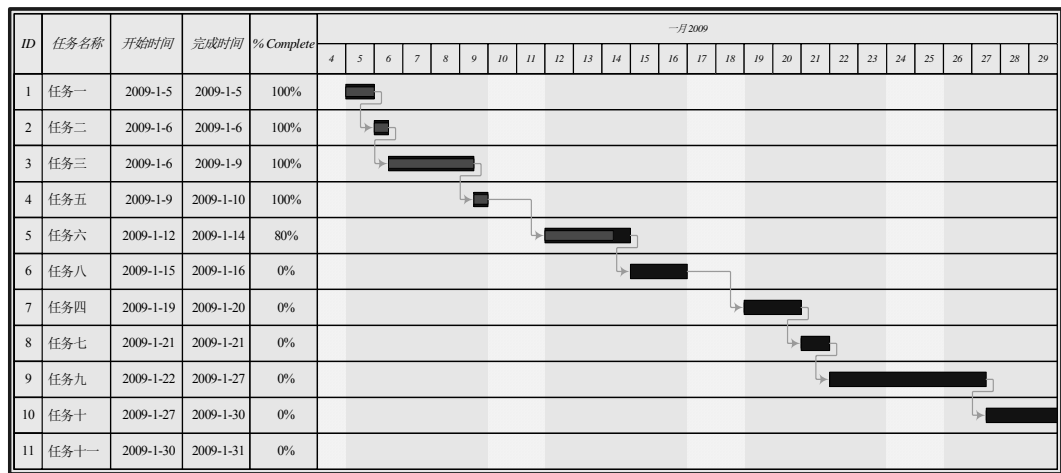


图 11-12 项目跟踪甘特图

那么项目经理小莫做的总结报告内容是否正确呢？

【分析】

挣值分析法是判断项目绩效的最好方法。

本项目当前进展到第 2 周，项目实际花费（AC）为 13 万元。

按照计划应该完成任务一、任务二、任务三、任务五、任务六和任务八，因此 PV 为 15 万。

但项目进展到第 2 周时，如图 11-12 所示，从项目组提交上来的进展报告中进行分析，任务一、任务二、任务三、任务五都已经全部完成，任务六只完成了 80%，因此 EV 为 12.2 万。

进度偏差(SV)=EV-PV=12.2-15=-2.8（万）

费用偏差(CV)=EV-AC=12.2-13=-0.8（万）

进度绩效指数(SPI)=EV/PV=12.2/15=0.81

费用绩效指数(CPI)=EV/AC=12.2/13=0.93

完工时的总估算(EAC)=AC+(BAC-EV)/CPI=13+(32-12.2)/0.93=35.27（万）

也就是说项目当前成本超支了 0.8 万元，进度方面还落后 2.8 万元的工作量没有完成，以此发展下去，项目总成本将超支 3.27 万元。

11.3 小结

软件项目监控是以项目计划为基准，通过各种会议和度量数据来对项目计划中的各个参数进行持续不断的跟踪和监督的过程，当发现某些参数的计划值与实际值之间出现偏差时，就需要尽快制订纠正的措施，并且将它作为工作任务安排给相关责任人，直到将偏差纠正为止。项目的监控过程对整个项目起到了直接监督的作用，是确保项目的产品满足客户需求的重要手段。

11.4 思考题

1. 项目监控的主要参数和次要参数都有哪些？
2. 项目监控的方法有几种？分别是什么？
3. 在挣值分析中 CV 和 SV 分别代表什么？
4. 在挣值分析中 CPI 和 SPI 分别代表什么？

12

第 12 章

软件质量管理的根源——需求工程

1995 年在美国开始的一项针对软件企业项目成功率的调查中,有 33% 的项目没有完工,剩下完工的项目中又有 51% 的项目没有交付¹²。通过对其原因进行深入分析后发现由于软件需求调研的失败,以及软件需求变动的频繁所造成原因占了将近一半。因此,软件项目需求调研的正确性和稳定性是项目成功的关键,如果有一个良好的需求调研过程和结果,那么可以毫不夸张地说软件项目成功了一半。

在软件行业发展的初期由于规模不大、复杂度也较低,软件研发主要关注于软件算法的优劣。后来随着软件规模的不断增大,对软件图形化界面友好性的要求也越来越高,最早的软件生命周期“瀑布式”模型就诞生了,软件需求也就成为了它的一个阶段。但随着对软件需求重要性认识的逐渐增加,软件需求已经从一个“阶段”被提升到“工程”的层面,并成为了软件工程中最核心、最复杂的过程之一,其原因主要集中在以下几个方面:

- ① 软件需求是判断软件质量是否满足客户需要的依据。
- ② 软件需求是软件项目范围的具体体现,范围的变化将直接导致项目成本和进度的变化。
- ③ 软件需求调研的内容没有范围可言,所有与项目直接相关或间接相关的人员都可能会涉及访谈,所有与软件产品直接相关或间接相关的知识、行业背景、客户的客户信息,甚至客户日常工作习惯等都需要了解得清清楚楚。
- ④ 软件研发人员所熟知的是自身软件行业的知识和信息,但所需要开发的软件产品却

涉及另外一个行业，这正是所谓的“隔行如隔山”。

⑤ 软件的需求就是客户方对自身行业所总结的最佳实践，客户方对这些总结的最佳实践是否具有条理性、系统性是很关键的。

⑥ 软件需求调研的过程可能会涉及客户方的各级管理人员和从事各种具体工作的操作人员、行业的专家、软件开发和测试人员等。因此，软件项目经理和需求调研人员要有很强的协调能力、组织能力和沟通能力。

⑦ 软件需求调研人员最好具备客户方的行业背景、扎实的软件开发和测试基础，以及良好的沟通和协调能力。但像这种样样具备且又愿意从事软件需求调研工作的人在国内还很少。

12.1 软件需求工程概述

软件需求工程的核心就是软件需求，按照 IEEE 提供的定义软件需求就是：

- 解决用户的问题或达到项目目标所需要的条件和能力。
- 软件系统或组件要满足合同、标准、规范或其他正式文档所需要具备的条件和能力。
- 反映以上两点所描述的条件和能力的文档说明。

在较早的软件工程中，“软件需求调研阶段”通常要完成以下工作：

- 调研客户的需求。
- 在《软件需求说明书》中描述软件的功能和性能需要，以及接口的信息。
- 将客户业务功能转化为计算机模型，并编写《系统需求说明书》。

在当今的软件工程中，“软件需求工程”是系统工程和软件工程的交汇，通过一系列的活动来明确客户需要，以在软件产品与软件质量、软件产品与客户之间架起一座桥梁。软件需求工程被分为软件需求开发和软件需求管理两个大的部分，它们分别要完成以下工作：

1. 软件需求开发

- 调研客户的需求
- 开发客户需求
- 开发组件需求
- 识别接口信息
- 建立操作场景的描述
- 建立功能需求的描述
- 分析客户需求
- 确认客户需求

软件需求开发的目的在于产出并分析业务、产品、功能和非功能性的需求。业务和软

件产品的需求主要汇总在《软件需求说明书》中，功能和非功能的具体描述将体现在《系统规格说明书》中。

业务的需求包括行业知识、客户的背景、客户的组织架构等方面的信息，是客户对系统、产品的高层次的目标要求。

产品需求描述了客户使用该软件产品所需要完成的任务、操作和功能。

功能的需求是指软件开发人员所必须实现的某个软件产品的功能，以使客户能够完成他们的目标，产品能够满足客户的需求。

非功能性的需求由以下几个方面组成：

- 性能需求：软件产品的技术性能指标，例如系统吞吐量、响应时间、最大并发数等。
- 环境需求：软件产品运行时所处环境的要求，例如硬件方面：主机 CPU 的个数、内存和硬盘的大小、网络带宽的大小等；软件方面：操作系统、数据库等方面。
- 可靠性需求：是软件产品在运行过程中不发生故障的时长、概率和应对措施。例如：系统是否可以支持 7×24 小时在线、是否对软件产品的数据进行自动化备份等。
- 安全性要求：描述该软件系统从硬件到软件都需要采取哪些安全及保密措施。例如：需要什么样类型的防火墙，用户的权限管理需要有哪些功能等。
- 界面友好性需求：通常符合客户日常工作操作习惯的软件，其界面友好性最强。
- 项目约束：对软件开发过程中时间、成本和其他各种资源的要求。

2. 软件需求管理

- 获取项目关系人对需求的理解和承诺
- 跟踪客户需求的变更
- 通过需求双向跟踪矩阵对需求进行管理，并发现需求与工作产品之间的不一致项
- 对需求管理过程中所发现的不一致项进行跟踪，直到最终关闭

软件需求管理的目的在于管理项目产品和产品组件的需求，并识别这些需求与项目计划和工作产品间的差异。软件需求管理的对象是项目收到或产生的所有技术性和非技术性需求。软件需求管理也要控制需求的变更，并记录需求变更的理由，以维护产品需求与所有产品和产品组件需求之间的双向追溯性。

12.2 软件需求开发的流程及最佳实践

软件需求的来源不止是客户提出的需求，有时组织或项目组也会提出对客户需求的补充。另外在软件设计或制订技术解决方案时，软件设计人员也会提出自己对软件产品需求的看法和建议，因此软件需求调研的对象不一定只是客户。

软件研发过程中的所有工作产品之间都有着相互间的先后顺序，可以说从原始需求起一直到系统测试用例都可以用一条“线”将其贯穿起来，在这条“线”上的每个工作产品发生变更都会带来一系列的连锁反应，如何做好变更的管理，如何利用工作产品间的相互关系发现它们与软件需求间的差异，这是软件需求管理的重点。

本节将通过一系列软件需求工程的最佳实践来指导软件需求的开发和管理的工作，为软件研发的整体质量打下坚实的基础。

12.2.1 需求调研的方法

软件的需求在项目启动之初对于项目组就像“雾里看花”那样既对未来充满幻想又让人捉摸不透。开发客户需求的过程就是收集关键人员的需要、期望、约束以及接口，并转换成客户需求。在开发客户需求时需要关注以下3个层次的内容：

- ① 业务目标、行业背景等信息。
- ② 要明晰每个用户具体工作的流程、所从事的工作内容、工作的环境、规章制度、组织架构、权限分配等信息。
- ③ 用户对于每个具体功能的使用习惯、该功能操作的具体内容和流程，以及非功能性的需求。

1. 需求调研的方法

软件的需求分为显性需求和隐性需求。显性需求是客户方正在使用的各种纸质文档、国家或行业的标准和规范、企业或部门运营的流程、组织架构、行业背景等已经文档化的信息。隐性的需求是关键人员头脑中的最佳实践、口口相传的行业经验、企业文化和各种潜规则等非文档化的或者需求调研人员不问就无人提及的信息。因此，软件需求开发的重点是隐性需求，客户需求调研的技术和方法如下：

- 显性需求调研方法

- 从客户方获取文档资料

客户正在使用的纸质文档、流程、规范和标准都是软件需求的重要组成部分，应该在需求调研时尽可能多地收集这些信息。

- 访谈和问卷调查

有些时候客户方的工作会比较忙，因此可以通过集中访谈或问卷调查的方式来对客户进行调研并索取纸质文档资料。

- 隐性需求调研方法

- 技术展示

客户往往都是不了解软件开发的，因此通过对软件项目组所掌握的开发技术进行展示，

可以启发客户他们所需要的软件产品大概是什么样子的。

➤ 操作剧本及场景描述

在市场上有各种各样的软件产品在销售，项目组通过借鉴同行业类似的产品功能和特点，加以对该客户行业背景的学习，就可以在需求调研前准备一些场景的描述或操作剧本，以便在需求调研时解释项目产品的功能和特点。

➤ 现场观摩和任务分析

去客户现场观摩其工作人员的具体操作，并通过任务的分配方式来了解企业运作过程中所使用的工作流程。

➤ 原型法

很多时候由于客户对软件产品缺乏概念，从而无法提出更深层次的隐性需求，因此需求项目组先开发一些产品原型给客户演示，从而对其思维进行启发。

➤ 脑力激荡

集体的力量是无比强大的，可以通过召集关键人员共同讨论，激发思维并不断碰撞，这往往是需求调研发掘隐性需求的常用方法。

➤ 更多的用户体验

原型法是在产品开发前进行的，用户体验是产品开发后进行的。更多的用户体验是为了让客户方有更多机会了解软件产品的功能和特点是否满足其需求，并通过体验的过程提出隐性的需求，以便项目组不断地完善。采用极限式开发模型的项目就是很好的例子。

● 通用的需求调研方法——5W1H 法

该方法非常简单，在业内的应用也非常广泛，效果也非常好。它是在需求调研的过程中向客户问 6 个问题，这 6 个问题中有 5 个问题的英文首字母是 W，1 个问题的英文首字母是 H，因此称其为 5W1H 软件需求调研法。

① Why（为什么）：首先做任何事情都要明确其目的和方向，客户提出的需求也应该有其目的性和方向性。这个目的就是客户为什么要提出这些需求，这个方向就是所开发的软件产品对他们有什么帮助。如果要开发一个财务系统的软件，可是客户提供的需求却是如何让企业的环保达标，那么这样的需求就与本项目背道而驰了。

② What（做什么）：既然在第一个问题 Why 中客户已经阐明了他的目的和方向，那么接下来就是要明确在这个方向上做什么，或者是实现哪些功能。通过这种方式就可以将一个大的系统逐级分解成小的模块、子模块或某些具体的功能。

③ Who（谁来做）：在 What 的逐级分解下已经有了那么多的功能需要实现，那么这些功能在企业中又是由哪些人来使用呢？通过 Who 可以明确最终用户的角色分类，对模拟客户场景是很有帮助的。

④ When（什么时候）：既然已经明确了有哪些功能以及有哪些人会使用这些功能，接下来就要清楚什么时候这些人才会使用这些功能。

⑤ Where（在哪里）：由 What 分解出来的功能应该在哪里由什么人在什么时候使用。例如：有些功能的使用是在公司上班时财务人员用的，有些功能是在外地出差时由业务人员在手机上使用的。

⑥ How（如何做）：以上 5 个 W 已经构成了一个场景的框架，如何对这个框架结构进行填充和关联，这就是 How 所需要描绘的内容。

5W1H 需求调研的方法是将需求调研的步骤结构化，它是一个从大到小逐渐细化的过程，该方法与其他任何一种需求调研的方法共同配合使用时都会产生更好的效果。

2. 如何解决需求调研过程中的矛盾

开发客户需求中有一点非常重要，就是关键人员对需求的约束。例如：在软件开发中客户提出的操作方式往往会因为软件开发技术的约束而只能用其他方式代替；有时某个部门提出的需求又会与其他部门的需求相矛盾，这些都会导致软件需求调研中矛盾的发生。如果这些矛盾造成了激化，那么必然会影响项目的发展及最终的验收，解决矛盾的唯一方法就是沟通。在做好沟通的同时希望可以通过以下方式辅助需求人员开展工作。

- 增强业务背景知识的学习，特别是对专业术语的理解和使用。只有学会使用客户方的语言表达方式才能更好地与客户进行交流。
- 要清楚了解客户的真实目的。例如：有时客户要求开发一个产品，项目组就会将重点集中于产品的各项功能中，而忽略了客户在调研中讲过的一句话：“将数据迁移后，原有的系统将不再使用”。用户已经暗示了该项目的一个重要前提是新系统必须要与原有系统顺利对接，这样才不会影响客户方的正常工作。如果项目组只注重产品的功能而忽略了与原系统的数据接口，那么这个项目风险将非常大。
- 需求一定要文档化。这样便于甲乙双方对项目的目标达成一致，而且也是将众多无序的需求进行梳理并将其条理化的过程。在这个过程中还可以发现很多未知的需求。
- 好的需求文档要图文并茂。能够用图形表述的内容一定要配合用例图、流程图、状态图、界面原型等图形。因为有些内容用文字描述起来会比较复杂，如果因为某些错别字或语法错误造成误解就得不偿失了。何况客户总是很忙或对需求调研缺乏耐心，通过图文并茂的方式可以加快其对产品需求的理解。
- “兼听则明”是双方合作成功的基础，项目组要多听取客户的建议，因为软件产品毕竟是客户使用的。客户也要在需求调研中多听取项目组的反馈，软件需求调研的过程有时也是对客户现有工作流程和习惯的一次过程改进。

- 对于软件需求调研人员来说，例如“界面友好”、“操作简单”这样的词语是不能使用的，因为这些过于笼统，应该提出自己的建议来让客户参考或选择。
- 是否变更往往是项目组与客户之间的主要矛盾，在文档化需求的基础上对需求变更进行估算，并告知客户所产生的影响，是化解矛盾的唯一方法。
- 软件需求调研人员要学会“反讲”，“反讲”的过程就是需求调研人员用自己的语言将客户所提供的需求信息重新讲给客户听。因为“反讲”在沟通中是确保信息发出者与信息接受者之间对需求理解保持一致的最好方法。
- 软件需求调研人员要有耐心，虽然客户的工作很忙，有时客户还认为需求调研会给他们带来额外的工作量，在需求开发和需求规格化的过程中总会遇到一些非常细致的问题需要客户解答。需求调研人员千万不要自己做主，一定要把每个问题问得清清楚楚，也千万不要觉得自己的工作会给客户带来麻烦，因为软件产品最终的使用者就是他们，需求调研人员现在所做的工作是在帮助客户今后可以更好地应用该软件产品来提高他们的工作效率。

3. 需求调研的流程和计划

在需求调研时应该制订一份《需求调研计划》，以明确参与需求调研的甲乙双方人员名单和责任、需求调研阶段需要交付哪些工作成果、需求调研的时间安排、需求调研的方式以及如何对需求调研的结果进行确认。

《需求调研计划》是甲乙双方开展合作的依据，因此制订完毕后必须得到客户方的认可。在制订《需求调研计划》时对关键人员的分析是至关重要的，如果遗漏了某个关键人员，那么需求调研的结果一定不完整。软件需求调研阶段的流程如图 12-1 所示。

12.2.2 软件需求分析的概述

需求调研时所获得的大量需求往往是不系统、不完整的，甚至有些需求是错误的、矛盾的或者是不必要的，只有通过需求分析的步骤发现问题并重新调研，才能将软件需求进行梳理，使其贯穿起来并具有条理性和完整性，最终将其规格化。如图 12-2 所示，需求分析是一个承上启下的环节。

软件需求分析的过程也是将需求逐渐细化的过程，最先能够确定并规格化的通常是业务和产品的需求，这些都是对软件产品需求整体的描述，往往记录在《软件需求说明书》中。对这些整体需求继续细化，就得到对各个产品组件的需求，这些需求也会被记录在《系统规格说明书》中。

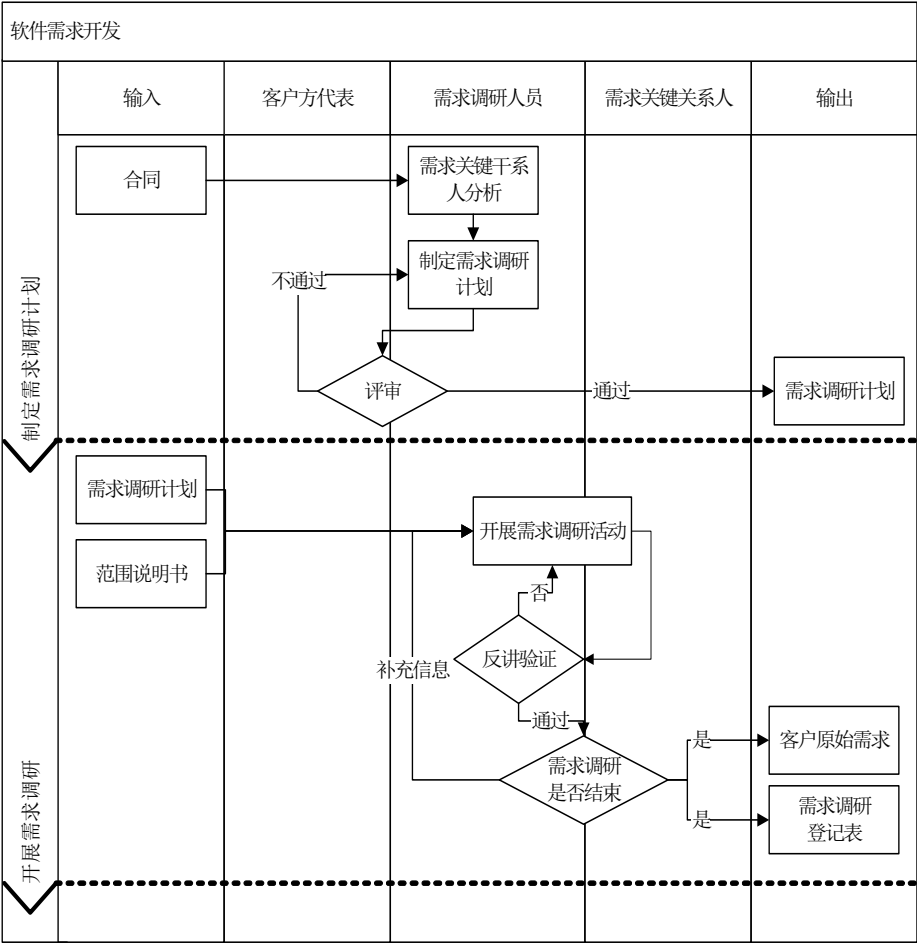


图 12-1 需求调研流程图

1. 需求分析的原理

软件需求分析的过程是将客户业务逻辑转化为规格化产品需求的桥梁，如图 12-3 所示，需求调研的过程就是对客户现有的软件系统、业务流程、纸质文档等进行提取的过程，经过汇总后将生成客户的业务模型。

对用户的业务模型进行抽象和升华的过程就是将用户物理的、纸质的业务模型转化为软件产品业务逻辑模型。该模型被规格化到《软件需求说明书》中，也可以使用 UseCase 图等进行表述。

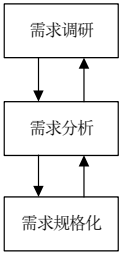


图 12-2 需求分析与调研、规格化关系示意图

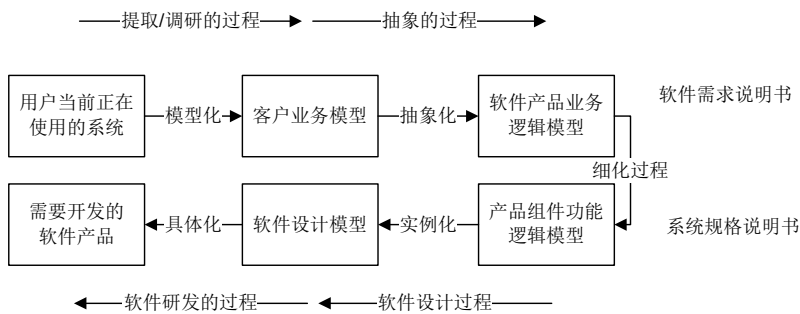


图 12-3 软件需求转换过程示意图

将软件业务逻辑模型进行细化，就生成了产品组件功能逻辑模型，该模型被规格化到《系统规格说明书》中，也可以使用组件图、活动图等进行表述。

软件设计人员依据《系统规格说明书》的内容将其实例成各个类或接口，最后由软件开发人员开发出最终的软件产品。

需求分析的大体步骤如图 12-4 所示。

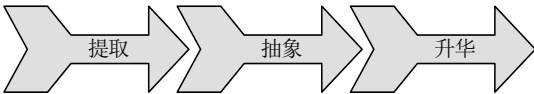


图 12-4 需求分析步骤

STEP 01 “提取”，它有两个方面的含义，一方面需求调研本身就是从客户的业务中提取出软件产品并帮助客户所实现功能的过程。另一方面是对采用“5W1H”法进行调研时对“**How**”所做工作内容的提取，软件需求人员在询问某个功能应该如何实现的同时还应该对该功能提取足够多的真实数据以供分析使用。

STEP 02 “抽象”，通过客户的原始需求以及该功能足够多的真实数据，需求调研人员就可以对它进行分析、归纳和抽象。其归纳和抽象的过程可以依据“5W1H”法中的 4 个“**W**”进行，例如：**Who** 是对角色和对象的分类，在系统中不同的角色会抽象出不同的权限模型；**Where** 是对操作媒介和环境的抽象，有些功能是在电脑上用的，有些功能是在手机上用的；在对 **When** 进行抽象时可以发现某个时间段有大量用户在使用同一个功能，因此就可以提出系统中某个功能的使用频率和性能要求。**What** 是对软件的功能进行抽象，以便在开发过程中提炼重用的组件。

STEP 03 “升华”，其目的有两个方面，第一个方面是借助软件自动化的实现方式对客户现有流程和管理进行优化，以此来提升企业的管理能力。另外一个方面是对汇总、抽象后的需求进行自我升华，以提出更为细化、深入的需求，例如：某个功能在上班前 2 小

时有大量用户使用，因此该功能的页面响应时间应该小于3秒钟。

2. 需求分析的方法——原型法

软件需求分析的方法有很多，例如：面向对象法、面向数据法、结构化法、原型法等，以下将为大家重点介绍原型分析法。

大多数的客户对于软件的需求来说都是模糊的、笼统的，或者有些客户心里非常清楚自己想要的是什么，但就是不知道如何说清楚。其实这些情况都很正常，就像过年了想要买件新衣服，只知道要的是衣服，它的功能是可以穿，最好是可以穿得好看。如果他不亲自去商场看看并试试所选中的样式，想必没有人可以帮他买到符合他要求的衣服。

原型法最早使用在机械制造业中，例如汽车厂商要研发一款新车，首先是进行市场调查，看看老百姓需要什么样的功能，然后设计图纸，接下来就是生产出样车，经过各方面对样车的试用和检验再对样车进行修改，此过程会反复多次直到满足各方面的要求后才会批量生产。

在软件需求分析中也借鉴了此原理，经过需求调研和分析后，软件的原型是产品早期一个可运行的版本，它反映了当时需求调研、分析的成果和效果。原型完成后就会给客户试用，客户会对原型进行评价并提出新的需求，其原理如图12-5所示。

有些原型可能只是一些界面的雏形加上文字的描述，有些原型则是经过设计和编码后可以运行的程序，因此原型法又分为原型的抛弃法和原型的演化法。如图12-6所示，以原型的演化法为例，经过多轮的迭代过程，原型的功能得到逐步完善，最终演化成交付给客户的产品，这种方法的好处是投入在原型开发上的时间和成本都没有白费。

原型法克服了传统软件分析过程中“纸上谈兵”、“闭门造车”的弊端，项目组与客户之间有了更多的交流机会，通过对客户进行原型的演示或客户通过亲自试用，可以更快获取客户隐性的需求，因此能够把握客户需求的要点。

12.2.3 软件需求规格化

通过以上对需求分析原理和方法的介绍，软件需求调研人员在需求阶段会生成两份文档，分别是《软件需求说明书》和《系统规格说明书》，这两份文档合称软件需求文档。软件需求调研人员在开发客户需求的过程中千万不要忘记对软件产品接口信息进行调研和分析，在此过程中所得到的接口信息大多是有关产品外部接口的描述，这些信息同样要记录在软件需求文档中。

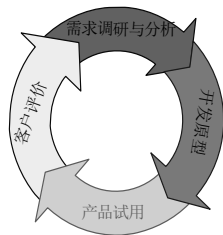


图 12-5 原型式开发模型示意图

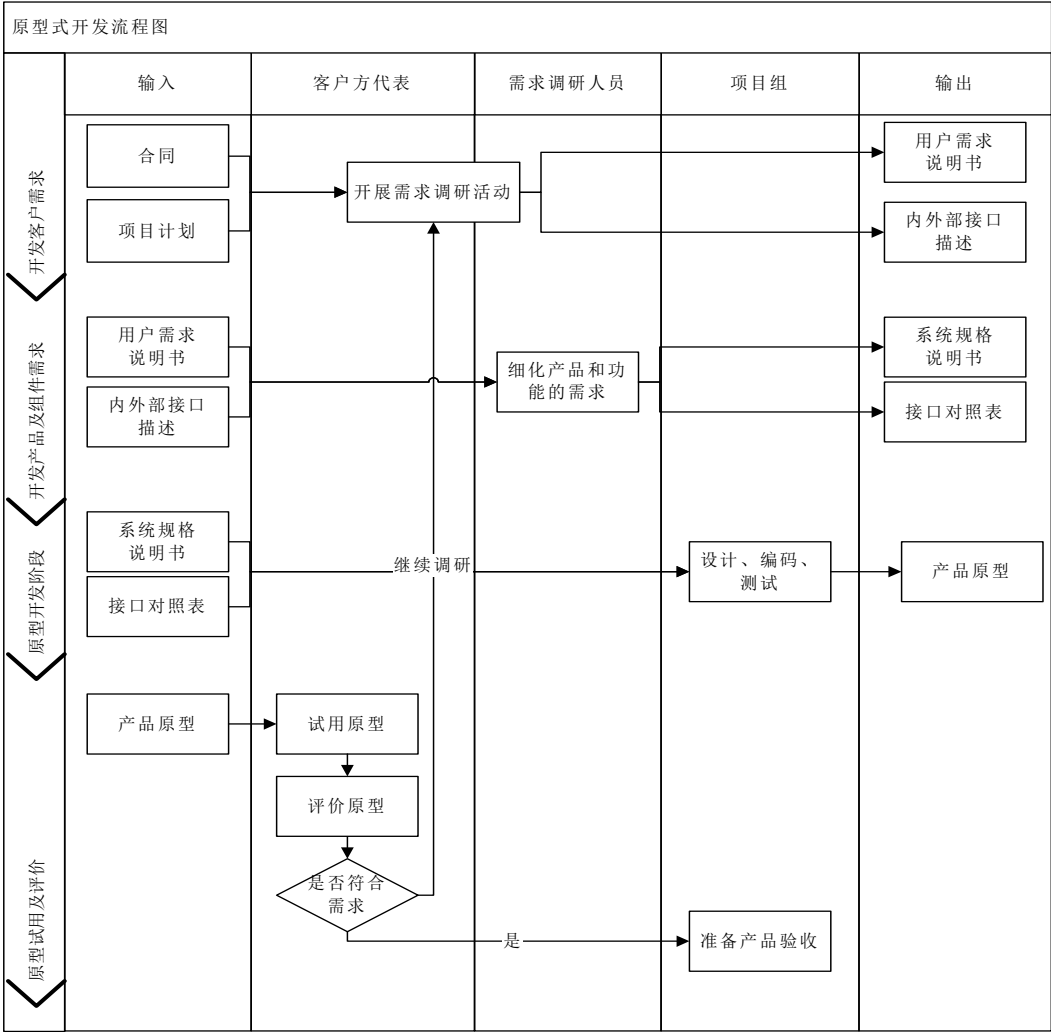


图 12-6 原型式开发流程图

1. 软件需求文档编写标准

合格的软件需求文档应该具备以下特点：

① 无歧义。在说明书中的每一个需求都只有一种解释。对于最终交付给客户的产品中的每一个特性都可以找到唯一的术语与其对应。如果某一个术语在某一特殊的段落中有使用过，那么该术语的每种含义必须作出解释并注明适用的场合。由于软件需求文档通常是使用自然语言编写的，因此在起草时必须注意消除其歧义性。

② 完整性。如果一份软件需求文档能满足以下要求，则说明其具有完整性。

- 要包含全部有意义的需求，无论是功能的、性能的还是设计约束或内外部接口信息的。
- 对所有可能出现的正常或非法操作的响应予以定义。
- 要符合软件需求文档的使用规定，不能随意删减章节，如果某些章节不适用，应该予以注释并说明。
- 软件需求文档中的参数要描述其范围和使用时的限定值、各种计算结果的精确度、系统负载的能力、响应时间和超时时间。
- 在软件需求文档中不能出现“待定”一词或相近含义的描述。

③ 可验证性。在软件需求文档中的每一条需求都必须可以找到对其验证的方法，并对它进行验证。

④ 一致性。在软件需求文档中的所有需求之间的描述不能出现矛盾；需求的描述与它对应的图形、列表中的内容必须一致；所有的需求必须与其他文档的描述一致。

⑤ 唯一标示。在软件需求文档中的每一条需求都要有唯一的标示。例如：SRS-1.1、SRS-2.3.5。

⑥ 可追溯性。如果软件需求文档中的需求是条例清晰的，那么在编写或修改该文档时，应该可以方便地引用每一个需求，并且可以通过需求的编号、术语等进行追溯。

⑦ 在软件需求文档中不能出现有关软件设计的内容，以及有关项目时间、成本、资源等方面的约束。

2. 《软件需求说明书》

该文档至少要包含以下内容：

- ① 目的：描述编写《软件需求说明书》的目的以及预期的读者。
- ② 范围：描述软件产品会提供什么功能，以及不提供哪些功能。
- ③ 缩略语、术语：通过对术语或缩略语的定义，可以指导读者对该文档的阅读。
- ④ 参考资料：罗列制订《软件需求说明书》时所参考的各种文档的名称、版本号、作者的名称、发布的时间等信息。
- ⑤ 项目背景描述：对项目背景的描述，有助于读者对本文档的理解。
- ⑥ 产品描述：当该产品是一个较大的系统时，应该对它各个组成部分的功能、接口进行描述。如果该产品是另外一个产品的组成部分，也要详细描述其功能和接口。
- ⑦ 产品功能：软件产品所要完成功能的一个概述，可以通过功能列表或方框图的方式进行描述，以便让读者可以快速了解软件产品的功能构成。
- ⑧ 设计性约束：对设计人员制订技术解决方案时的一种约束。例如：客户要求使用某个供应商的服务器、客户要求在某操作系统下运行程序。

⑨ 假设：在编写本文档时假设正确的事项或依据都要在此注明。该假设可能是导致需求变更的主要原因。

⑩ 具体需求的描述：要对每一条需求的输入、操作过程和输出结果进行描述，并且要针对每一条需求按照不同角色、不同使用方式来实例化一个操作场景。

⑪ 外部接口的描述：它包括了以下 4 种接口的描述

- 用户接口：主要是指用户所使用的 UI 界面。
- 硬件接口：如果软件产品需要与硬件设备进行交互，则要说明是什么样的设备以及如何支持这种设备。
- 软件接口：如果该软件与其他软件产品有交互的需要，则要说明其他软件产品使用的是哪种数据库、操作系统、版本等信息。
- 通信接口：特殊情况下要指定产品的哪些功能采用了何种通信的方式，例如：FTP、E-mail、UDP、TCP 等。

在《软件需求说明书》中对具体需求进行描述时要按照不同角色、不同使用方式建立实例化的操作场景。操作场景包括两方面内容，一方面是操作场景所处的环境，另一方面是指一系列操作或事件顺序发生的景象，操作或事件不同顺序的组合可能会产生不同的结果。

任何操作都会有其存在的环境作为约束，例如同样的查询订单和供应商资料时，业务员在公司使用电脑和在外地使用手机进行查询的方式是不同的。多个角色会使用同一组操作时，需要按照不同角色分别实例化操作场景。

建立操作场景对《软件需求说明书》来说是核心内容之一，操作场景的建立过程也是需求分析的过程，在此过程中往往还会发现许多新的或未明确的需求。

在对《软件需求说明书》进行验证时，通过对操作场景的演示也是一种非常好的检查方法。

3. 《系统规格说明书》

《系统规格说明书》是在《软件需求说明书》的基础上细化而成的，该文档所描述的需求也称为“产品与产品组件需求”。例如：在《软件需求说明书》中描述一个收发 E-mail 的功能时侧重的是收发邮件的过程和逻辑，而不会关心 E-mail 应该具备哪些属性，而《系统规格说明书》就要将 E-mail 细化到“收件人列表、邮件主题、发件人地址、邮件正文、邮件的附件”等信息。它们之间的差别如表 12-1 所示。

表 12-1 软件需求文档的内容差异

《软件需求说明书》	《系统规格说明书》
以客户的视角编写非技术类文档	以软件开发人员的视角编写的技术类文档
采用客户的业务术语进行描述	采用计算机术语进行描述

续表

《软件需求说明书》	《系统规格说明书》
侧重描述实现某个软件需求所需要的流程和逻辑	将功能细化为多个组件，侧重产品组件内部的流程、逻辑和功能
通过场景的实例来描述客户的业务	通过功能定义来描述产品组件的功能

《系统规格说明书》与《软件需求说明书》在格式上的最大区别在于“具体需求的描述”部分，《系统规格说明书》采用以下功能定义的方式来描述每一个产品组件的功能，如表 12-2 和表 12-3 所示。

表 12-2 产品组件功能定义汇总表

功能类别	功能名称、标识符	描 述
功能 A	产品组件 A.1	
	产品组件 A.2	
	产品组件 A.2.1	
	...	
功能 B	产品组件 B.1	
	产品组件 B.1.1	
	产品组件 B.2	
	产品组件 B.2.1	
	...	
功能 C	产品组件 C.1	
	产品组件 C.2	
	...	

表 12-3 A.1 功能描述

产品组件名称	
功能描述	
优先级	
输入	
操作序列	
输出	
补充说明	

功能定义也称为“功能分析”，如表 12-3 所示，它描述产品组件预期要实现的功能、产品组件内部的业务逻辑，以及各个产品组件间的操作顺序、输入、输出和其他说明如何

使用该产品组件的信息。

在建立功能定义时通常会先建立一个功能的汇总表，如表 12-2 所示。其目的是让读者对本产品的所有组件及其之间的关系有大体的了解，有时也可以通过方框图或 UseCase 图来表示。

功能定义是《系统规格说明书》中的核心内容，在对产品组件功能和内部逻辑进行定义的过程中有时也会发现新的需求或未明确的需求。

在对《系统规格说明书》进行验证时，通过对每个产品组件功能定义的审查也是一种非常好的检查方法。

操作场景是将一组功能串联起来以实现客户的某种需求，它使用的是功能而不是产品组件，因为《软件需求说明书》主要是给客户看的，而且客户并不懂什么是产品的组件。《系统规格说明书》则是将《软件需求说明书》中的某个功能拆分为不同的产品组件，体现了功能与产品组件之间的关系，着重描述了组件内部的逻辑和功能，这为软件设计对产品组件的抽象和重用提供了基础。

4. 对需求进行优先级排序

客户提出的需求通常会很多，有些是软件产品必须完成的功能，有些是要根据项目实际情况进行酌情处理的，有些根本就是与项目无关的。因此在需求规格化的时候必须对软件产品的需求进行优先级排序。软件的需求可以分为以下 3 种级别：

① 期望的需求：是软件产品必不可少的需求，即“雪中送炭”的需求，如果不做，那么软件产品将无法满足客户的需要，客户将无法使用该产品。

② 普通的需求：在项目成本、时间都允许的情况下应该实现的需求，这些需求往往能够让客户使用起来更为方便，可能是某些期望需求在特定情况下的替代或另外一种表现形式。

③ 兴奋的需求：往往是客户的一些额外要求，不做是不会影响客户使用的，做了就会起到“锦上添花”的效果。在对需求进行成本、时间和风险权衡时，这种需求通常会被筛选掉。此类需求有范围“镀金”的嫌疑。

5. 分析需求以取得平衡

软件需求文档起草完毕并不代表需求规格化工作的完成，需求规格化的最后一个步骤是要对需求进行分析，以在需求的规模、项目的成本、时间、风险等方面进行权衡，如果发现需求的内容超出了项目成本、时间所能够承受的范围或者某个需求给项目带来较大风险，那么就应该参考需求的优先级顺序并进行取舍。对需求、成本、时间和风险进行权衡时可以通过表 12-4 完成。

表 12-4 需求权衡列表

需求名称	功能规模 (功能点个数)	商业 风险	技术 风险	是否 取舍	取舍原因
功能 A1	50	无	无	舍弃	该功能规模过大，超出了本次项目的预算，并且暂时不做也不会影响项目的商业目标
功能 A2	10	无	低	保留	
功能 A2.1	20	无	中	保留	
功能 B1	40	无	低	保留	
功能 B2	30	无	高	舍弃	技术存在较高风险，建议下一期再进行开发
功能 C1	20	无	低	保留	
功能 C1.1	30	无	中	保留	
功能点总个数	200-50-30=150	成本、进度分析结果			以 2008 年年底组织级历史度量数据为依据，150 个功能点规模的项目比较符合本次预算和时间安排

6. 需求文档规格化的重要性

规格化客户需求的目的是为了把每次客户调研过程中所记录的客户原始需求进行梳理，使其具有条理化和可验证的能力，最终将多份需求调研的成果总结成一份《软件需求说明书》。分析和规格化客户需求的过程与需求调研的过程之间经常反复，因为在分析和规格化的过程中常常会发现一些未明确的信息需要重新进行调研。需求规格化的重要性如下：

- 为客户与软件项目组之间建立一个共同认可的目标，是项目一切行为的指导性原则。这个原则主要是为了让客户判断即将开发的软件产品是否符合他们的要求，或者如何修改现有的需求才能达到他们的要求。
- 软件需求文档化的重点是“过程”，过程的结果才是文档。软件开发是个思维缜密的过程，正是有了文档化的过程，才能将关键人员提出的需求条理化，才能在这个过程中发现隐含的需求。这就是所谓的“谋定而后动”，只有把握了这个规律才能在后续阶段减少返工，从而提高了开发的整体效率。
- 在项目启动时唯一能够估算项目规模的方式就是“功能点估算法”，功能点估算的依据就是详细的需求文档，只有估算出项目的规范才能对项目的成本、进度、资源等方面进行估算，因此没有文档化的需求也就没有准确、可信的项目计划。
- 软件开发过程中不时会伴随着软件验证的环节，单元测试的依据是《详细设计说明书》，集成测试的依据是《概要设计说明书》，系统测试的依据是《系统规格说明书》，用户验收的依据是《软件需求说明书》，如果没有了验证的依据，那么软件测试的工作和质量保证的过程都将无法开展，那么也就无法保证软件产品的质量。

- 越来越多的软件公司都希望有自己的产品，产品的功能和应用应该具有行业普遍适用性，最好的方法就是从众多软件项目中提取、融合共性的需求，如果项目都没有软件需求的文档，那么公司也就无法研发出普遍适用性的产品。

7. 软件需求开发过程小结

软件需求开发的流程如图 12-7 所示。需求开发的过程是由需求调研、需求分析和需求规格化 3 个部分组成的，这 3 个部分不是线性顺序排列的，而是一种“你中有我、我中有你”互相矫揉不可明确分隔的关系，其中需求分析起到承上启下的作用。

在需求规格化的过程将产生《软件需求说明书》和《系统规格说明书》，在规格化需求时经常会使用需求分析的手段。

《软件需求说明书》中记录了软件产品所能提供的功能，以及客户想要实现的功能。它是以客户的专业术语和视角对客户业务逻辑、工作流程、规章制度、组织架构、日常操作、各种表单等内容进行提取、抽象和升华的结果。

《系统规格说明书》是对《软件需求说明书》中功能进行细化的结果，它记录了每个需求功能所分解出来的各个产品组件，以及这些组件可以实现的功能、内部逻辑、组件与组件之间的关系等。它是以软件人员的视角和术语进行描述的，为后期软件设计中规划产品组件、重构产品组件提供了基础。

创建操作场景是规格化客户需求的重要手段，在《软件需求说明书》中通过操作场景的建立不但可以形成规格化的文档，而且还可以对需求进行分析，以发现更多遗漏的问题。

功能定义是规格化产品组件需求的重要手段，在《系统规格说明书》中通过对功能进行定义的方式来描述产品组件的需求，它比操作场景所关注的对象颗粒度更小、更细微，同样可以发现在需求调研中遗漏的内容。

待需求全部分析并规格化后，就要对全体需求进行优先级排序，以确定软件产品开发的重点，这样才能更加合理地制订项目计划。

最后还要根据合同对成本和时间的约束来估算软件需求文档中的内容是否可以完成，是否有些需求存在较高的风险。如果发现高风险的需求或者超出项目能力的情况，就需要根据需求的优先级进行权衡和取舍，从而保证软件项目的可行性。

12.2.4 需求验证及确认方法

当软件需求规格化后就需要对产出的工作产品进行验证和确认，以确保满足关键人员的需要、期望、约束及接口。

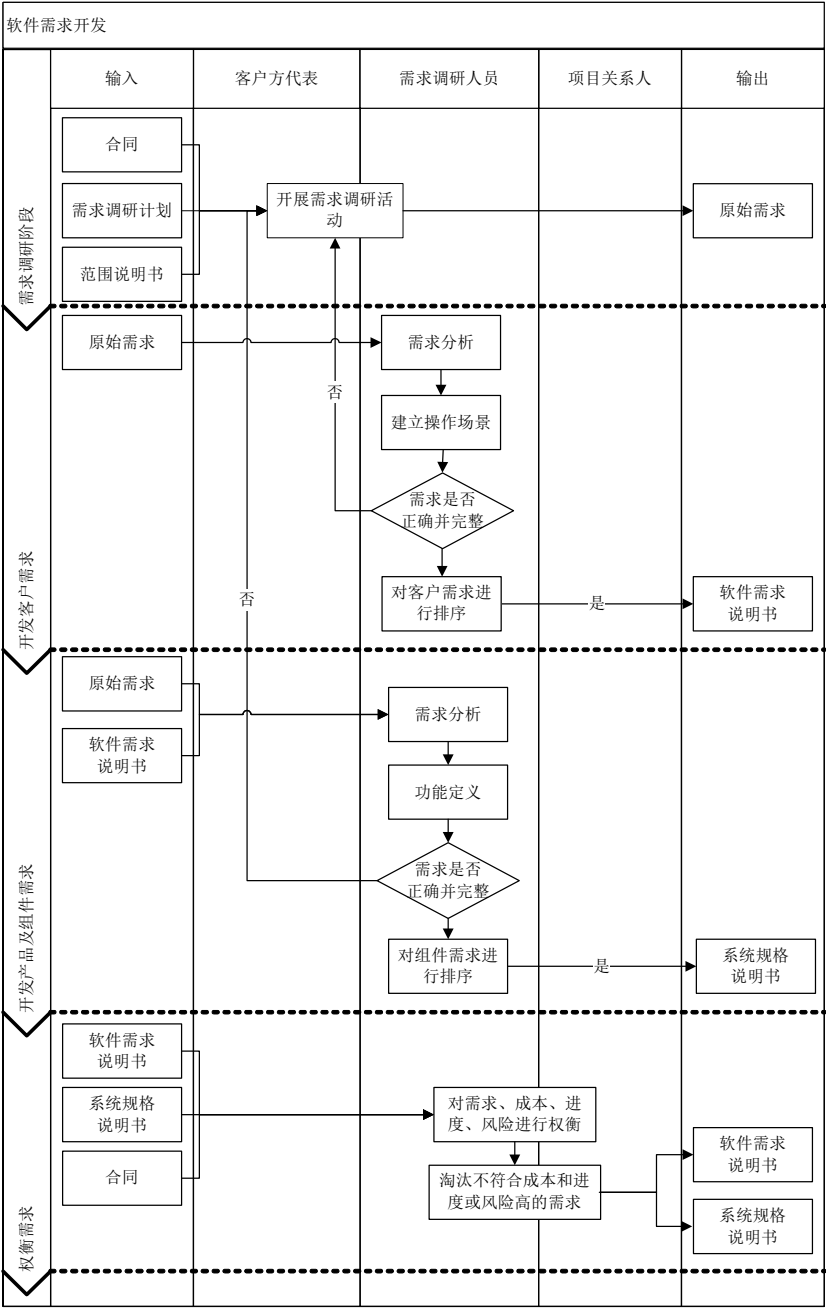


图 12-7 需求开发流程图

软件需求开发阶段工作产品有两个，分别是《软件需求说明书》和《系统规格说明书》，可以根据项目的具体情况对这两份文档采取分别验证及确认和一起验证及确认两种方式。如图 12-8 及图 12-9 所示，由于《系统规格说明书》是在《软件需求说明书》的基础上对需求进行细化的，因此建议将这两份文档进行分开验证和确认。

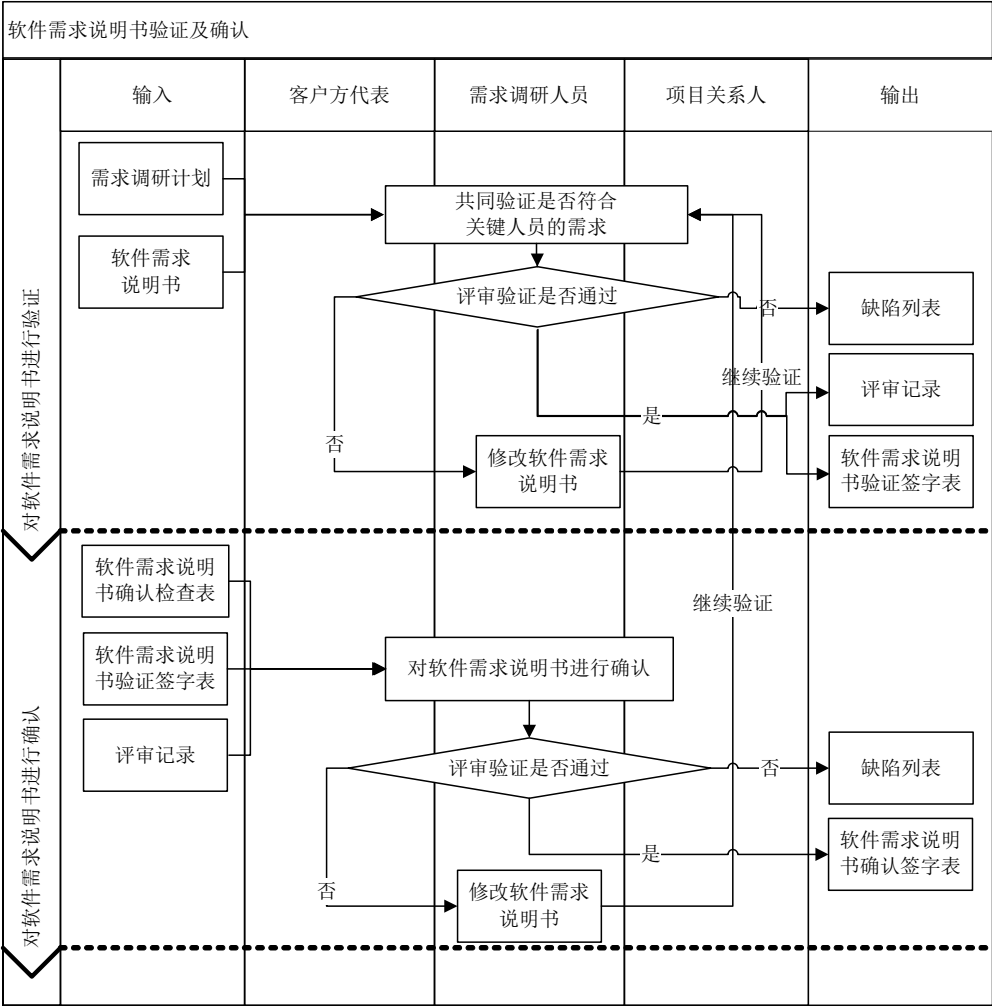
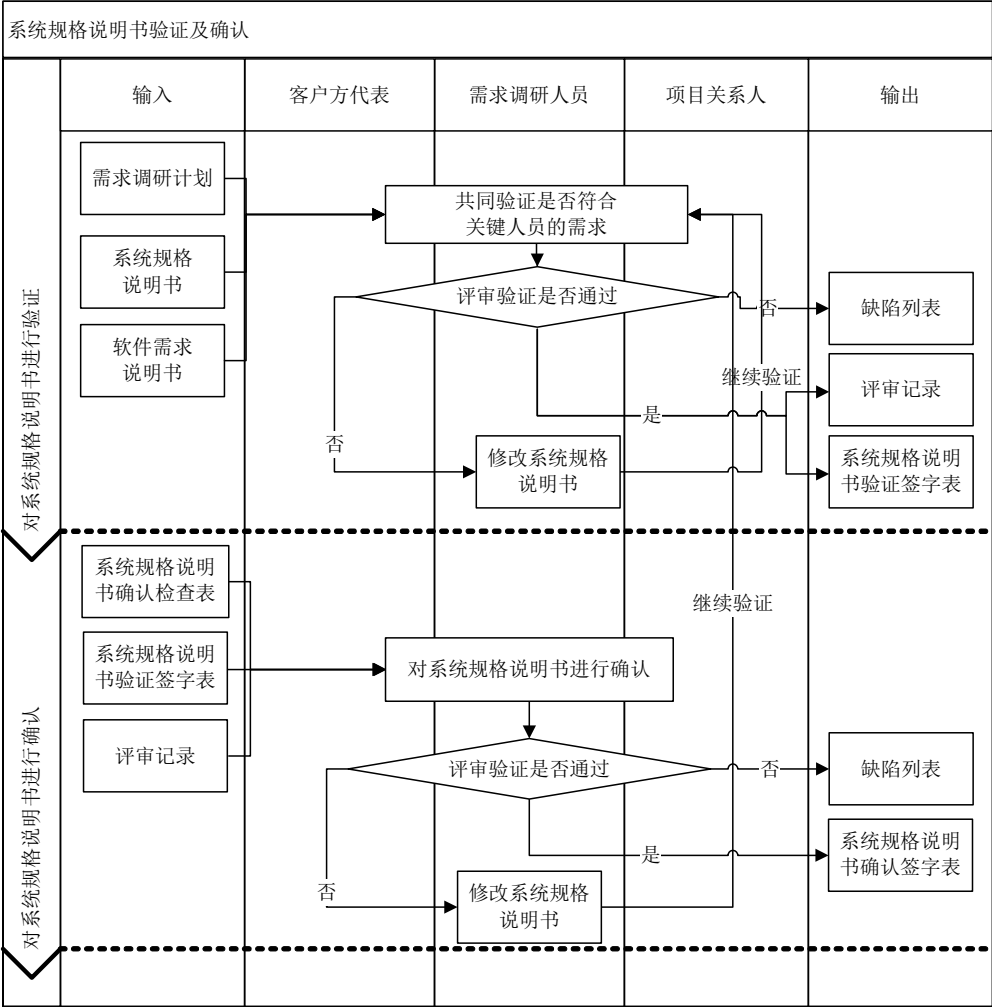


图 12-8 对软件需求说明书验证及确认的流程

在对《软件需求说明书》进行评审时重点要对操作场景的描述进行评审；在对《系统规格说明书》进行评审时要重点对功能定义的内容进行评审。



1. 验证方法 1

这两份工作产品都是文档，因此验证的方式均采用评审的方法进行检查。在评审时必须要有客户方代表、软件设计、开发和测试人员的参与。因为他们都是该工作产品的直接使用者。在对这两份文档评审时，除了评审其内容的正确性以外，还要对其编写的标准进行检查，其检查表的内容如表 12-5 所示。

表 12-5 《软件需求说明书》内容检查表

项目名称:		北极光		项目经理:		路人甲	
QA 名称:		路人乙		审计时间:		2009-3-8	
序 号	检 查 项			审计结果	本次审计是否适用	备 注	
1	《软件需求说明书》的内容是否达到无歧义的标准			√	√		
2	《软件需求说明书》的内容是否达到完整性的标准			√	√		
3	《软件需求说明书》的内容是否达到可验证性的标准			√	√		
4	《软件需求说明书》的内容是否达到一致性的标准			√	√		
5	《软件需求说明书》的内容是否达到可追溯性的标准			√	√		
6	《软件需求说明书》中的需求是否都有唯一的编号			√	√		
7	在《软件需求说明书》中不能出现有关软件设计的内容，以及有关项目时间、成本、资源等方面的约束。			√	√		
8	操作场景的描述是否覆盖了所有需求，描述的内容是否正确			√	√		

2. 验证方法 2

在对《软件需求说明书》进行验证时，如果项目组采用原型法进行开发并提供了软件产品的原型，那么就要搭建一个客户仿真环境来部署该原型，并让所有相关项目关系人一起来对原型的功能进行验证，其流程如图 12-10 所示。

3. 确认

对《软件需求说明书》和《系统规格说明书》进行确认时必须有客户、软件设计和测试人员的参与，缺一不可，因为他们都是这两份文档的直接使用者。确认的过程可以与其验证的过程同时进行，通过使用检查表的方式来完成确认的操作，其检查表如表 12-6 所示。

表 12-6 需求文档确认过程检查表

项目名称:		北极光		项目经理:		路人甲	
QA 名称:		路人乙		审计时间:		2009-3-8	
序 号	检 查 项			审计结果	本次审计是否适用	备 注	
1	《软件需求说明书》的内容是否通过了评审			√	√		
2	客户、软件设计和测试人员是否在《软件需求说明书》评审结果上签字			√	√		
3	《软件需求说明书》评审中所发现的缺陷均被修复			√	√		

续表

序 号	检 查 项	审 计 结 果	本 次 审 计 是否适用	备 注
4	《系统规格说明书》的内容是否通过了评审	√	√	
5	客户、软件设计和开发人员是否在《系统规格说明书》评审结果上签字	√	√	
6	《系统规格说明书》评审中所发现的缺陷均被修复	√	√	

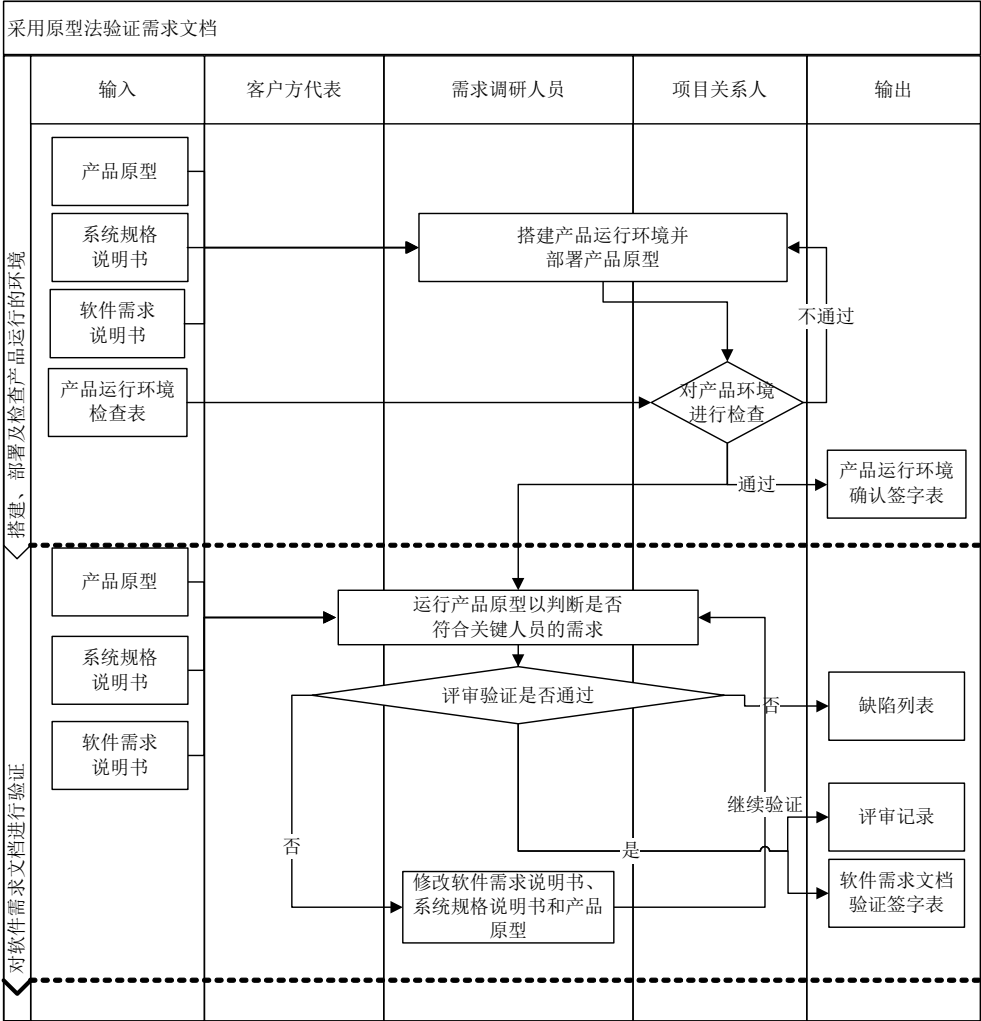


图 12-10 采用原型法验证软件需求文档

12.3 软件需求管理的流程及最佳实践

对软件需求进行管理的目的是为了控制软件需求的变更，维护需求与工作产品之间的关系，识别需求与工作产品之间的差异，对发现的差异进行纠正直到关闭。如图 12-11 所示，软件需求管理包含以下 5 个方面的内容。

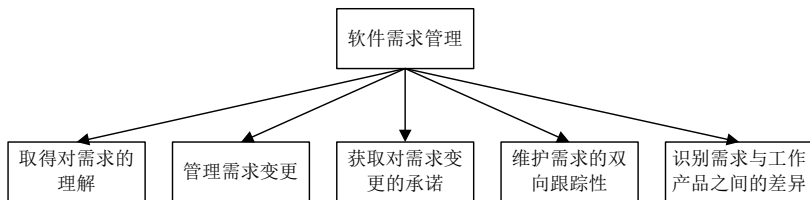


图 12-11 软件需求管理示意图

1. 取得对需求的理解

软件的需求是软件研发的基准，是项目关系人的共识，特别是项目团队成员更要理解需求，否则就会出现最终的软件产品不具备客户所要的功能。

通常软件需求文档得到确认后，在需求阶段里程碑前，项目组会依据《软件需求说明书》和《系统规格说明书》进行培训，为了保证培训的效果可以采用“反讲”的方法来检验每个项目成员是否都理解了需求。

2. 管理需求变更

软件需求调研时不可能 100% 获取到客户的需求，软件项目也不可能不出现变更，只要按照变更管理的流程对变更施加管理就可以减小对项目的影响。按照变更的影响可以将变更分为重大变更和日常变更两种。通常软件项目的重大变更会有以下特点，除此以外的都属于日常变更。软件需求变更管理的方法和流程可以参考本书第 6 章中的相关章节。

- ① 项目的规模增加或减少 10% 及 10% 以上
- ② 里程碑的延误 5 个及 5 个以上工作日
- ③ 项目组成员的变动 20% 及 20% 以上
- ④ 项目成本累计超支 10% 及 10% 以上
- ⑤ 项目进度累计延迟 10 个及 10 个以上工作日
- ⑥ 在某一阶段严重级别的缺陷达到 20 个及 20 个以上

3. 取得需求变更的承诺

当发生需求变更时，要对变更的影响进行评估，当需求变更得到批准后，还要取得相

关人员对本次变更的承诺。例如：需求调研人员承诺软件需求文档已经更新并达到客户的要求；项目团队成员承诺可以实现本次需求变更的内容。

对需求变更的承诺可以通过对需求变更的评审获得，评审的流程可以参考本书第 4 章的相关内容，获取承诺的方法可以参考本书第 3 章的内容。

4. 维护需求的双向跟踪性

如图 12-12 所示，软件的需求是根据原始需求得到的，产品组件的需求又是通过软件需求得到的，设计文档又是参考产品需求制订的，以此类推可以发现软件项目的很多工作产品之间都有着关联关系，不管是从前向后，还是从后向前都可以进行追踪，此种跟踪方式称为需求与工作产品之间的横向跟踪。

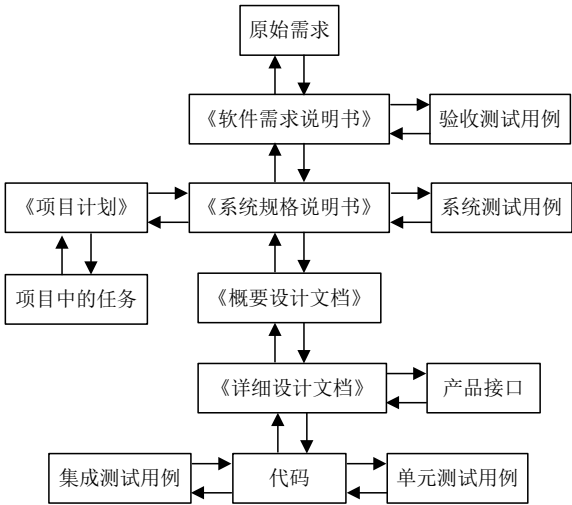


图 12-12 软件需求与工作产品间横向跟踪示意图

除了需要与工作产品之间的横向跟踪以外，很多工作产品本身还有纵向跟踪的关系，如图 12-13 所示，软件需求本身、代码本身也具有双向跟踪性。此种跟踪方式称为工作产品纵向跟踪。

维护工作产品之间的双向跟踪关系对评估软件变更的影响范围是十分有用的，如果项目组不清楚工作产品之间的关系，那么很可能在变更时遗漏某些工作产品，这就会给项目带来非常隐蔽而又致命的风险。

当软件项目组将各种工作产品之间的关联关系搞清楚之后，就可以填写到《需求跟踪矩阵》和《工作产品跟踪矩阵》中，通过跟踪矩阵来维护它们之间的关系。

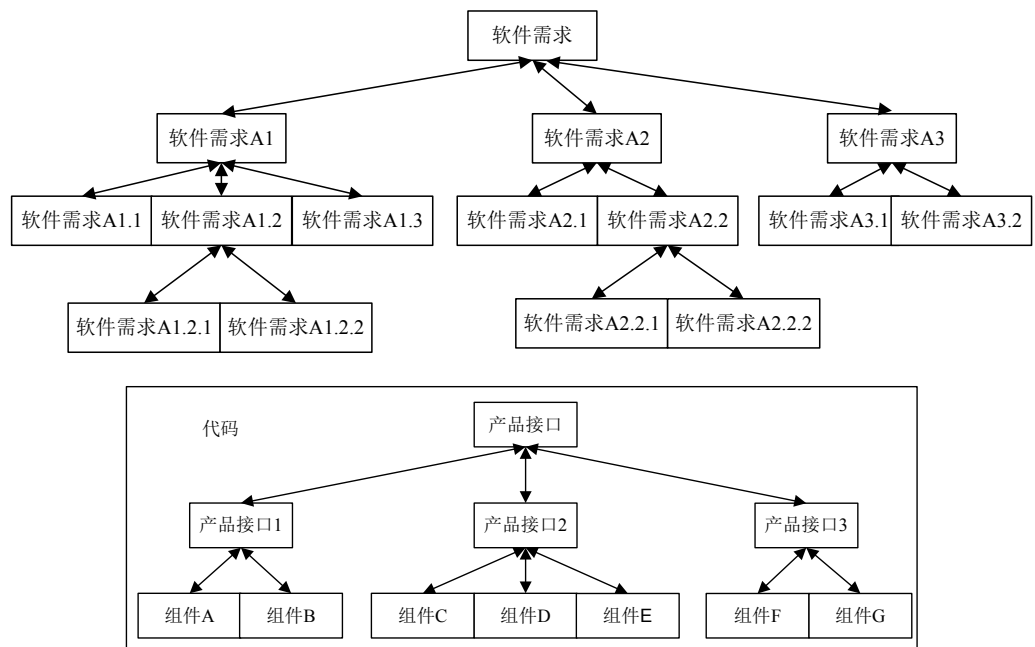


图 12-13 工作产品间的纵向跟踪示意图

不管是原始需求还是产品组件的需求，在对其规格化时都会为每一个需求分配一个唯一的标识符，例如原始需求可以用 OR1、OR1.1 来表示；软件需求可以用 SRS1、SRS1.1 表示；产品组件需求可以用 PR1、PR1.1 表示。对于设计文档中的功能点同样也会分配唯一的标识符，例如概要设计文档 SD1、SD1.1；详细设计文档 DD1、DD1.1。对于测试用例也是一样，例如系统测试用例 ST1、ST1.1；单元测试用例 UT1、UT1.1；集成测试用例 PT1、PT1.1。

如表 12-7 所示，项目组可以将各种工作产品中的这些唯一标示在填写《需求跟踪矩阵》对应的表格中，以此来建立并管理它们之间的关系。当项目组件需求 PR2.2 发生变更时，项目管理人员就可以很容易得知 SRS1、SD5、DD7 以及其他工作产品都会受到影响，这样对项目变更的估算就会更准确。

表 12-7 需求跟踪矩阵

原始需求	软件需求		组件需求		概要设计	详细设计	...
OR1	SRS1	SRS1.1	PR1	PR1.1	SD1	DD1	...
						DD2	...
						DD3	...

续表

原始需求	软件需求		组件需求		概要设计	详细设计	...
OR1	SRS1	SRS1.1	PR1	PR1.2	SD2	DD4	...
				PR1.3	SD3	DD5	...
							...
							...
			PR2	PR2.1	SD4	DD6	...
				PR2.2	SD5	DD7	...
							...
			PR3	PR3.1	SD6	DD8	...
				PR3.2		DD9	...
	SRS2	SRS22.1	PR4	PR4.1	SD7	DD10	...
OR2	SRS3	SRS3.1	PR5	PR5.1	SD8	DD11	...
			PR6	PR6.1	SD9	DD12	...
				PR6.2		DD13	...
				PR6.3		DD14	...
		SRS3.2	PR7	PR7.1	SD10	DD15	...
						DD16	...
			PR8	PR8.1	SD11	DD17	...
					SD12	DD18	...
						DD19	...
					SD13	DD20	...
			PR9	PR9.1			...
							...
		SRS3.3	PR10	PR10.1	SD15	DD23	...
			PR11	PR11.1	SD16	DD24	...
	SRS4	SRS4.1	PR12	PR12.1	SD17	DD25	...
				PR12.2		DD26	...
				PR12.3		DD27	...
				PR12.4	SD18	DD28	...

5. 利用跟踪矩阵识别差异

利用双向跟踪矩阵中记录的需求与工作产品以及工作产品自身间的追踪关系，就可以

发现完整性的差异，如表 12-7 所示，软件设计人员遗漏了组件需求 PR9.1 的设计。因此利用“跟踪矩阵”对工作产品进行验证也是一种非常好的方法。

12.4 软件需求工程常见问题及案例分析

在日常工作中会发生各种各样的有关软件需求工程方面的问题，下面将通过几个案例来找到问题的本质并寻求解决问题的方案。

12.4.1 对需求关键关系人分析的重要性

【案例】

一家外贸公司的老总找到当地一家著名软件公司的 CEO，希望可以为我国内贸易部研发一套单证系统。于是 CEO 将此任务交给了项目经理小苗，这是小苗第一次独立担当项目经理。

由于国内贸易部从经理到职员一共才 5 个人，因此小苗打算采用集体访谈的方式，大家共同讨论单证系统的需求。在访谈过程中大家都踊跃发言，纷纷提出自己的想法和建议，并将他们日常所使用的单证样本都提供给了小苗。小苗对此次访谈的效果也非常满意，他向公司保证两个月一定可以交付一个满意的产品给客户。

两个月后单证系统交付给客户试用，可是外贸公司却迟迟不肯验收，并指出单证系统还缺少很多报表导致不能满足他们的需求。

这到底是哪里出了差错呢？

【分析】

其实小苗忽略了对需求关键关系人的分析工作就直接开展对客户的调研和访谈，这种风险非常隐蔽，不到项目交付是看不出来的，而且对项目的危害又非常严重，就像小苗这个项目那样很可能会导致项目的失败。

小苗的失误在于忽略了外贸公司的老总也是需求的关键关系人，毕竟这个项目是老总发起的，他想要一套单证系统肯定是有原因的。在这个项目中，外贸公司的老总希望通过单证系统可以统计公司的业务量和金额，并且希望可以分析贸易产品的利润空间。因此这不是一套简单的应用系统，而是包含数据挖掘和分析功能的管理系统。

因此小苗对需求关键关系人分析的失误导致了该项目偏离了客户的需求。

12.4.2 调研时需求关键关系人不能被代替

【案例】

某软件公司承接了一个为当地外贸集团研发一套内部管理系统的项目，项目经理小乔兼任该项目的需求调研和分析工作。今天小乔按照制订好的需求调研计划去对该集团储运部业务员小李进行访谈，可是到达外贸公司后才发现小李突然外出办事。小乔只好去找储运部的经理老王确认小李什么时候才能回来。

老王说：“我们的工作都很忙，不可能因为你的到来而影响我们的正常工作。你有什么事情就问我吧，我给你 30 分钟的时间，你赶快问。”老王将储运部的工作流程等大体情况向小乔进行了介绍，并提供了一些纸质的文档模板。

小乔回到公司后非常高兴，因为他今天不但完成了计划中对储运部经理的访谈，而且还间接完成了对储运部职员小李的调研，达到了“事半功倍”的效果，这下可以不用再去看客户的脸色了。

可是这个事情被公司副总老张知道了，老张觉得这次调研只算完成了一半，还需要再去储运部找小李进行访谈。项目经理小乔对此十分不解。

【分析】

公司副总老张向小乔对今天所发生的事情进行了如下分析。

在制订需求调研计划时访谈人员名单是经过反复讨论并充分听取了甲乙双方的意见才制订下来的，是具有代表性的。在需求调研时关键人员之间的建议和想法有时会出现矛盾或约束，因此不能用一个人的观点来代替另一个人的观点。虽然储运部的老王是小李的领导，但在需求调研时他们都是被访谈的人员，只是同一类用户中选出的代表，对需求调研活动来说并不会考虑他们之间的行政级别，因此对业务员小李的访谈并没有完成。

12.4.3 需求文档规范化与 XP 极限式开发的理念是否矛盾

【案例】

某软件公司是一家 CMMI 3 级的企业，项目经理小何是一个技术高手，他与许多软件技术人员一样都对 XP 极限开发模型非常欣赏，因此他的项目不管规模大小统统采用 XP 极限式开发。

公司副总老张对此情况十分不解，通过一次与小何的谈话才得知事情的原委。由于 XP 极限式开发强调的是“以人为核心，以代码代替文档”的思想，并且强调拥抱变化。因此小何觉得 XP 极限式开发是一种先进的开发模型，它优于过程烦琐并让人感觉过程臃肿的 CMMI 模型。

那么 XP 极限式开发与 CMMI 之间是否真的有矛盾呢？哪些地方又是矛盾的根源呢？

【分析】

CMMI 强调的是规范化的文档，当然软件需求文档也包含在内。XP 极限式开发强调的是以代码的方式来代替文档，软件项目中不需要有太多的文档。由此可见对于规范化的文档方面是有矛盾的，那矛盾的根源是什么呢？

XP 极限式开发是针对软件项目实施时的一种生命周期模型，CMMI 是对软件企业成熟度进行评估的一种标准，而不是对软件项目进行评估的标准。所以，CMMI 关注的是软件企业，而 XP 极限式开发关注的是软件企业中的某个项目，这也就是矛盾的根源。

采用 XP 极限式开发某个软件项目可能会成功，可以给企业挣到更多的钱，但是软件项目给企业带来的金钱效益却远远多于知识、技能和经验的贡献，因为 XP 极限式开发提倡的是以人为核心，以代码代替文档的理念。如果项目的核心成员离职了，那么谁能保证在短短一个月的工作交接过程中可以掌握他对项目的所有知识和经验呢？软件企业的流动率一直都不低，这走的是一个人，如果多走几个那么软件企业又该怎么办呢？

文档规格化的过程是将头脑中的知识编码化，也就是将隐性的知识显性化。这样做的唯一目的就是支持软件企业的长久发展，为了该目的也势必会增加项目组的成本和工作量，这也就是矛盾的核心。

12.4.4 利用需求跟踪矩阵来应对项目变更

【案例】

某软件公司项目经理小朱第一次承担大规模的项目，他最近有些苦恼，项目发生的两次变更将原来井井有条的生活搞得一团糟，现在有种“头痛医头脚痛医脚”的感觉。到底应该如何管理好项目变更呢？

【分析】

在项目变更管理中会规定项目变更的流程，以及哪些是重大变更，哪些是日常变更，这些都属于变更管理的规范，而不是变更管理的技术。

利用《需求跟踪矩阵》对需求和工作产品进行横向和纵向的跟踪是控制软件项目变更最好的技术方法。在每次对变更影响进行估算时，可以通过《需求跟踪矩阵》准确知道变更影响的范围；《需求跟踪矩阵》需要被定期更新，在对变更后的效果进行验证和确认时，通过更新后的《需求跟踪矩阵》还可以知道哪些工作产品被修复，这样就确保了变更的完整性。

只有学会利用《需求跟踪矩阵》对变更进行管理，才能将软件项目管理得有条不紊。

12.5 小结

软件需求过程分为需求开发和需求管理两大部分，需求开发是需求调研、需求分析、需求规格化和验证、确认需求的通称。需求管理讲的是如何应对需求的变更，如何利用工作产品的双向跟踪性来管理好项目的变更。

只要通过良好的需求开发过程来保证需求的正确性，通过需求管理的过程来保证需求的完整性，这样的软件项目就已经成功了一半，这样的软件项目所开发出来的产品质量才会满足客户的需要。

12.6 思考题

1. 需求工程分为哪两大部分？
2. 需求调研、需求分析与需求规格化之间是怎样的关系？
3. 需求管理由哪些部分组成？
4. 通过什么来记录和维护需求与工作产品之间双向跟踪的关系？

13

第 13 章

软件质量管理的群体决议机制—— 决策分析

当“5·12”汶川大地震发生后，全国人民都自发地行动起来贡献自己的力量，就像温家宝总理说的那样：“一个很小的问题，乘以 13 亿，都会变成一个大问题；一个很大的总量，除以 13 亿，都会变成一个小数目”。可见当今社会群体的力量不可小视。在日常生活中人们总会遇到各种各样的机会或者需要利用有限的时间、资源来做更重要的事情，俗话说得好：“把钢用到刀刃上”。当人们面对众多的选择时往往会眼花缭乱，那么应该如何进行判断和选择呢？从大的方面来看，国家定期召开全国性的“人民代表大会”和“政治协商会议”，这就是利用群体的力量和智慧来对国家重要的事情进行决策；从小的方面来看，当一个高中生在填写高考志愿时，往往一家人都会坐下来开个家庭会议进行讨论。利用群体的思维和力量，以会议协商的形式来保证未来发展一帆风顺，这是小到个人，大到国家都在使用的一种方式。

在软件研发过程中也有很多时候需要通过群体决议机制来确定项目或产品未来发展方向，例如：对软件技术解决方案的选择、对硬件设备厂商的选择等。

但在现实生活中也经常会发生这样的情况：“真理有时掌握在少数人的手中”，也就是说群体性决策的结果不一定是正确的。那么如何通过群体决策保证软件项目朝着正确的

方向发展呢？这就需要依靠本章所要讲述的一套科学的决策分析方法来实现。

13.1 软件决策分析概述

软件决策分析的目的在于依据已经建立的评估准则对各种已经识别出来的候选解决方案进行选择 and 决策。软件的决策分析过程是一个正式的评估过程，在软件生命周期的任一阶段都有可能发生，因此一个软件项目决策的目标要尽早识别，以便安排相应的时间和资源。

软件决策分析的过程是个结构化的方法，它包含了 6 个环节：建立决策分析指南、建立决策的准则、选择评估的方法、识别候选解决方案的提案、评估候选解决方案的提案、选择解决方案。其总体流程如图 13-1 所示。

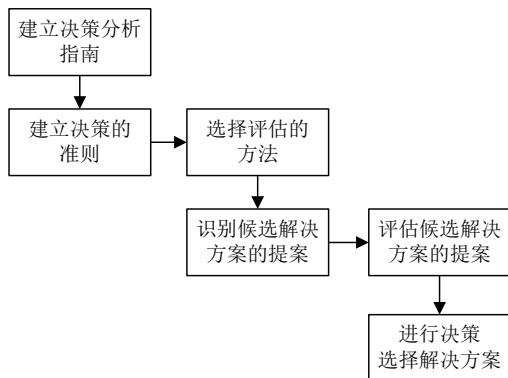


图 13-1 软件决策分析总体流程

软件决策分析的对象不只是针对技术类的解决方案，对于其他非技术类的方案选择也同样适用。通过以上结构化的方法就可以减少软件决策分析过程中的主观性，并且可以通过这样的方法来选择一个符合不同项目关系人多样性需求的、支持项目未来发展的解决方案。

总的来说，软件决策的过程有些类似招投标的过程。首先要制订招投标的规范，也就是软件决策分析的指南。然后制订招标书中的各项需求以及评标的方式，这就是决策分析中的决策准则与决策方法。决策的准则要尽早制订并进行发布，以便方案制订人可以及时评估自己是否有能力完成该解决方案。如果方案的内容与决策准则谬之千里，那么就浪费了大家的时间。接下来就会收到来自各方的标书，在决策分析过程中也就是解决方案的“提案”，决策小组就需要对众多的提案进行识别、评估和筛选，最终选定入围的解决方案。最后就是正式评标和开标的过程，决策小组会根据决策的准则和方法对候选方案进行选择，以确定最终的解决方案，并且会将方案选择和拒绝的理由告知相关人员。

13.2 软件决策分析流程及最佳实践

如图 13-1 所示，本节将对软件决策分析的 6 个最佳实践进行逐一讲解。

1. 建立决策分析指南

决策分析是一个正式的评估过程，它需要花费很多人力、物力和精力，并不是每个决策都重要到需要正式的评估。若在软件项目中没有明确的定义，那么在遇到问题时首先就会在重要与不重要之间进行无休止的讨论。当问题涉及中、高风险，或者问题会影响项目目标时，通常建议使用正式的决策分析过程。项目管理人员可以通过以下标准对决策的对象进行判断，在软件项目中常会发生的决策如表 13-1 所示。

- 当某个问题与中、高优先级的风险有直接关系
- 当某个问题会导致项目延迟一定时间或比例
- 当某个问题会影响实现该项目目标的能力
- 当决策的投资回报率合理时

表 13-1 软件项目中常见的决策

决策对象分类	决策分析的对象
产品需求类	对项目范围变更的决策 对需求优先级的决策
项目计划类	对项目可行性研究的决策 是否要开展单元测试的决策 是否要开展集成测试的决策 对供应商选择的决策 对项目计划变更的决策 对项目过程定义的决策
软件设计类	技术解决方案的决策 产品设计方案决策 产品集成方案的决策 Make or Buy，即自己开发还是购买现成产品的决策 重用组件还是自己开发的决策 对软件设计变更的决策
软件开发类	软件开发环境的决策 软件开发工具选择的决策
软件测试类	软件测试环境的决策 软件测试工具选择的决策 用户验收测试环境的决策

软件决策分析指南是确定何时使用正式决策的过程来解决非计划中的问题。决策分析指南通常要在项目计划阶段完成，可以是一份单独的文档，也可以作为项目计划中的一个章节，其制订流程如图 13-2 所示。

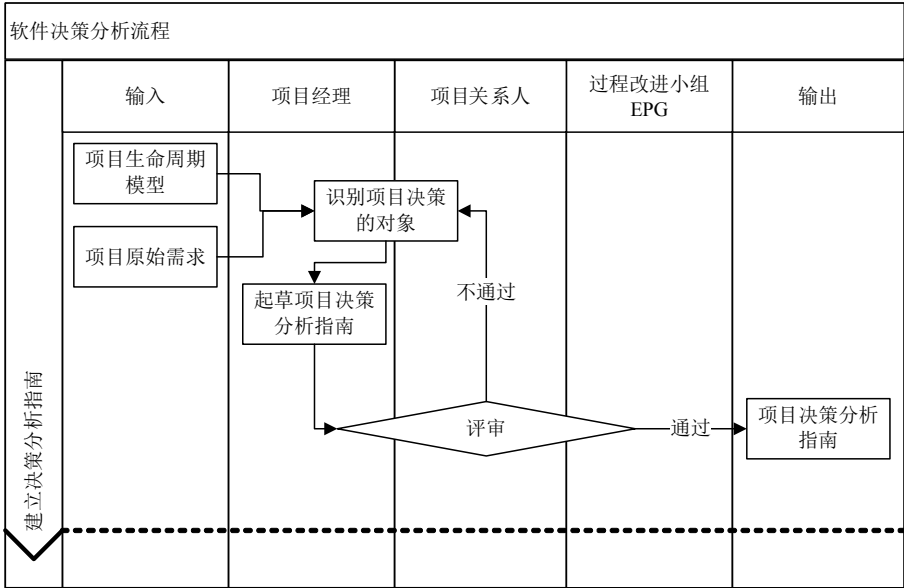


图 13-2 建立决策分析指南的流程

2. 建立评估准则

决策分析的准则可以是量化的，也可以是非量化的。量化的准则是使用“权重”来反映不同准则之间的重要性。非量化的准则可以使用“高、中、低”、“大、中、小”的方式进行较为主观的等级分类。评估准则是评估候选解决方案的基础，通过对评估准则的排序，排列在前的准则对评估的影响相对较高。

如果决策的准则太多，还可以为其设置过滤条件，例如：“必须可以在.NET 平台上运行”或者“必须选择 Dell 的服务器”。在准则定义时还需要记录该准则制订的原因，以及对该准则分类的原因，这样可以避免在后续决策过程中出现决策人员对决策准则合理性的猜疑。在制订决策分析准则时可以从以下 4 个方面进行思考，制订决策准则的流程如图 13-3 所示。

- ① 技术约束：项目组掌握的技术与实现软件产品所需要具备的技术之间往往存在差异。
- ② 环境冲击：外界环境的变化对解决方案的选择有很大的影响。
- ③ 风险：风险是软件决策分析中必不可少的准则。

④ 项目成本：公司在进行某些决策时往往会受到成本的约束。

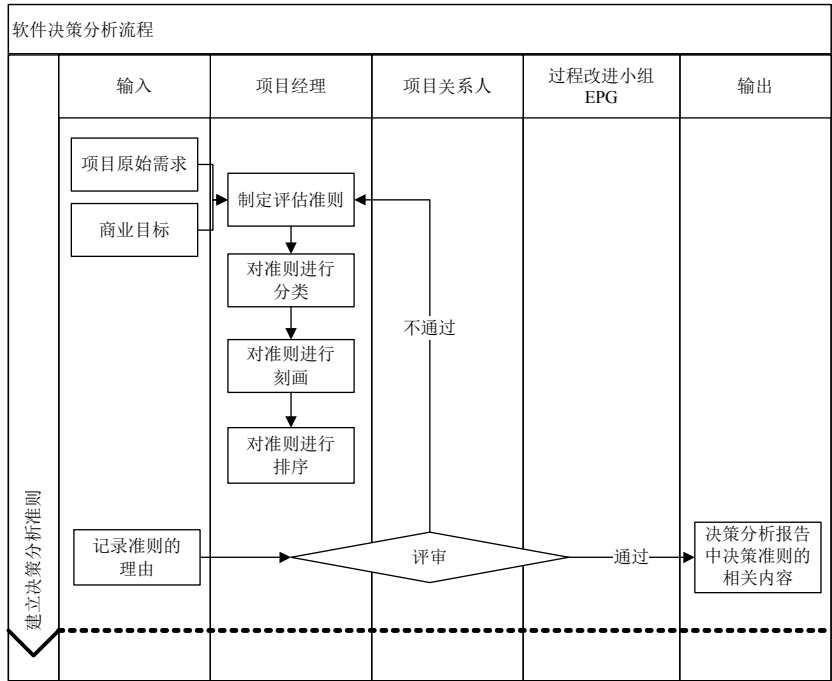


图 13-3 决策准则制订的流程

例如当对某个软件产品架构候选方案制订决策准则时，可以从以上 4 个方面进行思考，其内容如表 13-2 所示。

表 13-2 对某产品架构候选方案进行决策的准则

序 号	准则内容	缘 由	条 件	权 重
1	是否可以满足公司内所有项目架构的需求	内部环境	必选条件	20%
2	该架构研发的周期是否可以符合市场需要	外部环境	必选条件	20%
3	研发该架构所需投入的成本	成本因素	必选条件	11%
4	该架构具有分布式的功能	技术约束	必选条件	10%
5	该架构设计理念的前瞻性	技术约束	必选条件	10%
6	实现该架构本身的技术风险	风险	必选条件	8%
7	该架构整合工作流的难易程度	技术约束	必选条件	8%
8	通用性、扩展性和重用性	技术约束	必选条件	8%
9	开发人员对此架构的易用性	内部环境	可选条件	5%

3. 选择决策的方法

选择一个恰当的决策方法将会节省决策的成本和时间，降低决策的风险，增强决策的效果。典型的决策方法如下：

- 模拟：通过决策树的模拟方法进行决策。
- 调查：通过对当事人或相关人员和环境调查的结果进行决策。
- 基于经验：聘请专家进行决策。
- 基于原型进行推断：一般用于技术解决方案的决策。
- 测试：对技术解决方案进行测试，例如：性能测试等。
- 加权打分法：通过筛选和加权打分的方法进行决策。
- 一票否决法：针对高质量或高风险的问题进行决策时使用。

4. 识别候选解决方案的提案

对于一般的软件项目来说，在进行决策分析时能有两套解决方案供选择就可以了，但是对于大型项目来说，解决方案的提案可能会有很多选择，从中识别出合理的候选方案将是一个非常关键的环节。

让不同技能和背景的关键人员参与头脑风暴的讨论，有助于识别和解决各种假设、约束与偏见。头脑风暴也可能会激发更有创意方案的产生或将多份提案进行融合，形成候选解决方案。候选解决方案必须是一份详细的、具有可操作性和执行性的方案。

5. 评估候选解决方案的提案

使用已经建立的决策准则和方法，对候选解决方案的提案进行评估。评估分为分析、讨论和评审三个环节，有时反复的分析和讨论也是必要的，最终形成候选解决方案，其流程如图 13-4 所示。

有时也会在对候选方案的分析和讨论时发现提前制订的决策准则存在一定的不准确性。因此，在评估方案时也应该鼓励调整决策的准则和相应的假设条件。

6. 进行决策并选择解决方案

依据决策的准则和方法，对从众多提案中筛选出来的候选解决方案进行决策，最终选定一个方案。一个方案的最终选定可能包括多次的评估，有时新技术的产生会在评估时改变候选解决方案，或者供应商的价格发生了改变。

在进行正式决策时，有时会在评估的信息不足时进行决策，这就必然会产生风险，因此在软件决策分析的过程中不但会产生一份《决策分析报告》来记录方案选择或拒绝的理由，而且还会输出相应的风险到项目风险列表中，其流程如图 13-5 所示。

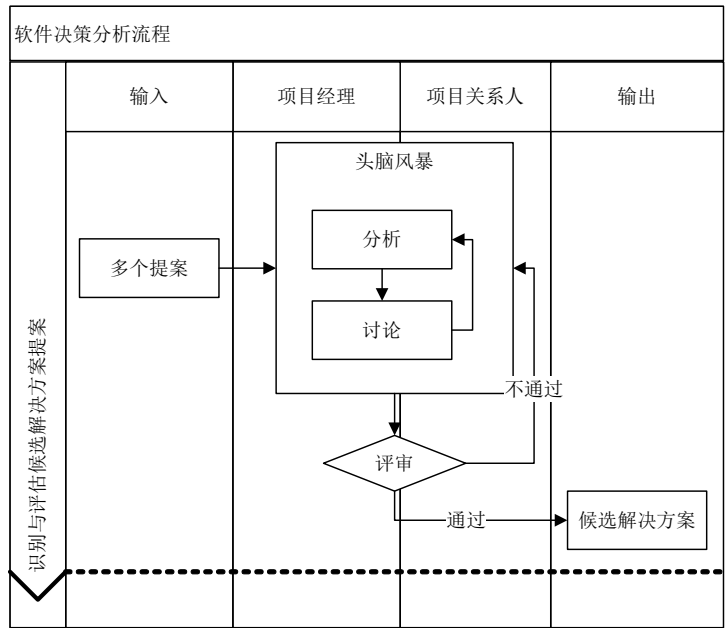


图 13-4 识别与评估候选解决方案提案

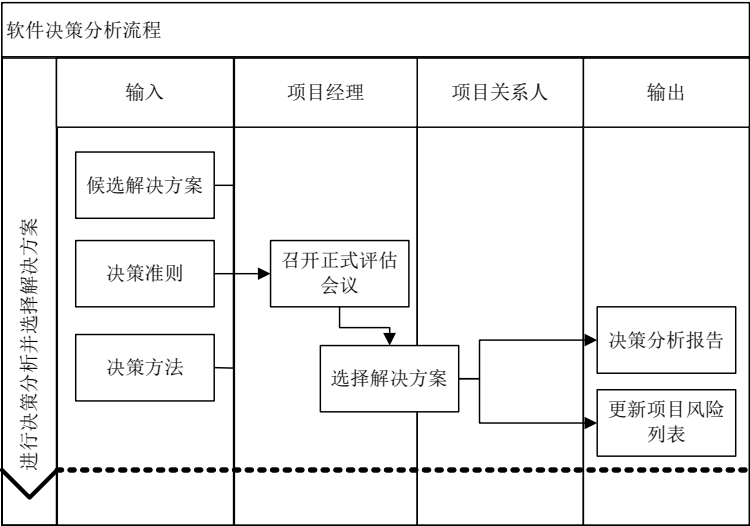


图 13-5 决策分析并选择解决方案

13.3 软件决策分析常见问题及案例分析

软件的决策分析关系到项目未来的发展，以下将通过两个案例来对软件决策的具体过程进行讲解和分析。

13.3.1 决策树的使用方法

【案例】

某软件公司打算开发新一代的互联网产品，有两个项目经理都非常适合作为这个项目的负责人。于是公司决定让他们按照项目需求分别制订项目计划，然后通过决策分析的方式从中选择一人来负责此项目。

由于该项目是研发公司自有的产品，因此与其他项目的不同之处在于项目时间上没有过于苛刻的要求，公司唯一的期望就是在合理的成本下尽快实现该产品的所有功能，并确保产品的质量。

本次决策主要的决策准则就是项目成本，公司希望项目经理对项目成本的可能性也进行估算，并且采用决策树模拟的方式进行评估。

【分析】

决策树（Decision Tree）一般都是自上而下来生成的。每个事件都可能引出两个或多个事件，并导致不同的结果，把这种决策分支画成图形就像树一样，故称为决策树。决策树是对候选方案进行选择的一条简捷的途径，它不仅可以帮助人们理解问题，还可以帮助人们解决问题。

决策树分析法是应用最广的归纳推理算法之一，它是一种逼近离散值目标函数的方法。

决策树分析法的优点如下：

- 可以生成用于理解的规则
- 计算量相对来说不是很大
- 可以处理连续和离散属性字段
- 决策树可以清晰地显示哪些字段比较重要

决策树分析法的缺点如下：

- 对连续性的字段比较难预测
- 当类别太多时，错误可能会增加得比较快
- 一般只是根据一个属性来分类

● 对全局性判断不足

如图 13-6 所示，项目经理小李制订的项目计划中有 60%可能会花费 80 万元，30%的可能会花费 85 万元，最坏情况发生的概率大约是 10%，这种情况下项目可能花费 100 万元。项目经理小璞制订的项目计划中有 70%的情况会花费 85 万元，30%的情况会花费 90 万元。

根据决策树的分析方法，两个项目经理项目成本的期望值如下：

小李制订的项目计划成本期望值=80×60%+85×30%+100×10%=48+25.5+10=83.5 万元

小璞制订的项目计划成本期望值=85×70%+90×30%=59.5+27=86.5 万元

依据决策准则的规定，项目花费最少的获胜，因此项目经理小李获得了此项目。

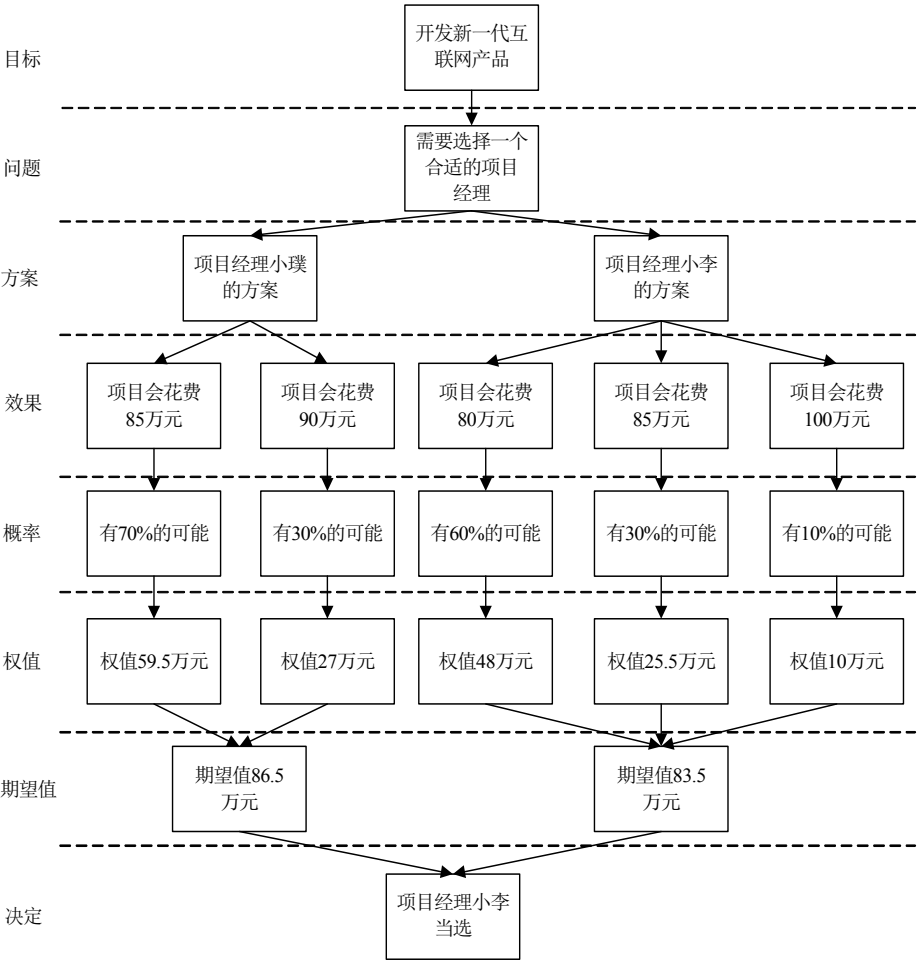


图 13-6 决策树分析案例

13.3.2 加权打分的决策方法

【案例】

某软件公司物流项目正处于概要设计阶段，公司的 3 名架构师将提出 3 套不同的方案供项目经理小艾进行选择，为了可以客观地对这些方案进行评估，小艾决定使用加权打分法对项目进行决策。

【分析】

加权打分法是进行决策分析时常用的一种方法。这种方法的具体做法是：首先列举进行决策时应该考虑的重要准则，其次按照过滤条件进行第一步筛选。然后为每条准则分别打分，其分值和权重值相乘得出该准则的积分，最后将全部准则的积分加起来得出一个方案的总分。对每个候选方案都采用同样的方法来打分，最终通过每个方案得分的高低来评价其好坏。

加权打分法的优点如下：

- 打分容易
- 核算简单
- 便于反馈

加权打分法的缺点如下：

- 适用范围较小
- 采用本方法时，需要根据具体的决策对象设计不同的决策准则

项目经理小艾首先将项目的需求发给了公司技术委员会的成员，然后寻求他们的帮助制订出 9 条决策的准则，其中第 9 条准则“开发人员是否容易理解此架构”为可选条件，其他 8 条决策准则都是必须符合的。在制订完决策准则后，小艾将其分别发给了这 3 名架构师。

一个星期过去了，3 名架构师分别提交了他们的方案给项目经理，小艾召集技术委员会的成员开会对其进行决策。如表 13-3 所示，决策小组通过决策准则的属性对候选方案进行过滤，结果架构师小胡的方案过于复杂，使它不能满足决策准则 2“该架构研发是否可以在 2009 年 5 月 1 日前完成”而被首轮淘汰。

表 13-3 对候选方案进行过滤

	准则 1	准则 2	准则 3	准则 4	准则 5	准则 6	准则 7	准则 8	准则 9
决策 准则 的 内 容	是否可以 满足公司 内所有项 目架构的 需求	该架构研 发是否可 以在 2009 年 5 月 1 日前完成	研发该架构所 需投入的成本 必须小于 50 万， 但不得低于 40 万，越少越好	该架构 具有分 布式的 功能	该架构 设计理 念的前 瞻性	实现该架 构本身 的技术风险 高低	该架构是 否可以整 合公司的 工作流产 品	通用性、 扩展性和 重用性的 能力	开发人员 是否容易 理解此架 构

续表

	准则 1	准则 2	准则 3	准则 4	准则 5	准则 6	准则 7	准则 8	准则 9
属性	必选	必选	必选	必选	必选	必选	必选	必选	可选
小李的方案	√	√	√	√	√	√	√	√	√
小胡的方案	√	×	√	√	√	√	√	√	√
小赵的方案	√	√	√	√	√	√	√	√	
筛选的结果:	由于小胡的架构设计方案不能满足“准则 2”的要求，因此被淘汰。其他人的设计方案将通过加权打分进行评判								

接下来决策小组针对每条决策的准则对小李和小赵的方案进行打分，最后将每项的得分乘以权重，然后累计求和得出每套方案的总分，如表 13-4 所示。

表 13-4 对过滤后的方案进行加权打分

序 号	准则内容	权 重	小李方案得分	小赵方案得分
准则 1	是否可以满足公司内所有项目架构的需求	20%	10	10
准则 2	该架构研发是否可以在 2009 年 5 月 1 日前完成	20%	10	10
准则 3	研发该架构所需投入的成本必须小于 50 万，但不得低于 40 万，越少越好	11%	8	6
准则 4	该架构具有分布式的功能	10%	10	10
准则 5	该架构设计理念的前瞻性	10%	6	8
准则 6	实现该架构本身的技术风险高低	8%	8	7
准则 7	该架构是否可以整合公司的工作流产品	8%	10	10
准则 8	通用性、扩展性和重用性	8%	7	6
准则 9	开发人员是否容易理解此架构	5%	3	0
*每条准则最多可以给 10 分，最少可以给 0 分				

小李方案得分=10×20%+10×20%+8×11%+10×10%+6×10%+8×8%+10×10%+7×8%+3×5%=2+2+0.88+1+0.6+0.64+1+0.56+0.15=8.83

小赵方案得分=10×20%+10×20%+6×11%+10×10%+8×10%+7×8%+10×10%+6×8%+0×5%=2+2+0.66+1+0.8+0.56+1+0.48+0=8.5

经过加权打分的计算，小李的方案最终以 8.83 分获选。

13.4 小结

软件决策分析是通过评审会的方式选择一份合适的解决方案，决策分析的过程中会制订决策的准则和方法。软件确认过程和同行评审一样可以采用评审会的方式，也同样需要制订相关确认或验证的准则，而且同行评审和软件确认过程也可以与决策分析使用相同的决策方法。它们之间的区别是：决策分析的结果是项目在后续过程中会按照该解决方案进行实施，是为未来发展进行规划；软件确认过程的结果是活动相应的承诺并且建立信任的机制；同行评审的结果是对工作产品有效性、正确性的验证。

13.5 思考题

1. 决策分析都包含了哪些步骤？
2. 决策分析常用的方法有哪些？
3. 决策分析与同行评审和软件确认过程有哪些相似的地方？
4. 决策分析与同行评审和软件确认过程有哪些不同的地方？

软件质量管理的构建机制——产品集成

软件研发过程中产生的各种文档、代码、组件、脚本等工作成果都称为工作产品，当对组件、模块或代码进行组装并形成最终交付给客户的可执行、可应用的软件产品的过程称为软件产品集成过程。简单地讲就是从工作产品到产品的组装过程。

软件产品构建机制分为两类，一类是编码开发阶段最后进行的一道工序，称为软件产品集成，如图 14-1 所示；另一类是编码开发阶段中的一种渐进性的、重复性的、螺旋式的、持续性的过程，称为持续式集成。在此重复过程中不断重复组装产品或组件，对组装后的工作产品进行评估，然后再次组装，在该过程中软件产品或产品组件的功能也在逐渐递增。

软件产品集成的关键是对产品或产品组件内部和外部接口的管理，以确保接口与接口间、接口与组件间的兼容性。

对软件质量体系中的缺陷和风险，追溯其根源很多都是因为软件产品集成的工作没有做好，或是进行软件产品集成的时机没有把握好而导致的。例如：某软件项目到了客户验收阶段才发现软件产品与客户现有系统的接口不匹配，造成客户的某些数据无法导入到系统中；某软件项目开发的模块单独测试都没有问题，当在进行组装后有些数据流转的时候却发生错误。读者通过本章的

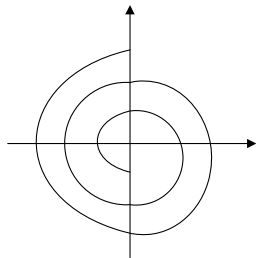


图 14-1 持续性集成示意图

学习可以掌握软件产品集成的要点，将从源头来提高软件产品的整体质量，并可避免大量的返工以及高风险的发生。

14.1 软件产品集成管理概述

软件产品集成的目的就是组合产品的模块或组件，使其形成更复杂或最终可交付的产品。在产品集成期间要确保集成后的产品符合用户和设计的需要。在理解此定义时要注意以下 3 点：

- ① 产品集成出来的对象不一定是最终的产品，也可能是项目过程中某一个中间的组件。
- ② 如果集成出来的是更为复杂的组件，那么要符合《概要设计》的要求。
- ③ 如果集成出来的是最终交付的产品，那么要符合《软件需求说明书》和《系统规格说明书》的要求。

软件开发中的很多流程和其他传统行业的生产过程和方法在原理上都具有非常多的相似性，软件产品集成的过程可以归纳为一只笔的组装过程。如图 14-2 所示，一支普通的签字笔可以分解为 4 个部分：“笔帽”、“笔杆上半截”、“笔杆下半截”和“笔芯”。每个部分都相对于一个产品的组件，要想将它顺利组装起来必须按照以下步骤进行：

STEP 01 第一次集成：将“笔芯”放入“笔杆下半截”生成组装后的第一个半成品。

STEP 02 第二次集成：将第一个半成品与“笔杆上半截”进行组装，生成第二个半成品。

STEP 03 第三次集成：将第二个半成品与“笔帽”进行组装，生成最后的交付产品——一支完整的签字笔。

STEP 04 对最后装好的这支签字笔也要进行一下测试，拿一张纸在上面画一画看看是不是好用。

由此可见产品集成时是有一定先后顺序的，假如将此顺序打乱，例如先将笔杆的前、后部分进行组装，再组装其他部件，那么很有可能产品的集成将会失败。

再仔细看一下“笔杆上半截”和“笔杆下半截”，它们的对接处有相应的螺纹，这也就相对于软件中的接口，只有匹配的接口才能保证产品集成过程的顺利，假如笔杆前后部分的接口不匹配，那么也会导致产品集成的失败。

最后还要对组装后的产品或更为复杂的组件进行评估和测试，以确保每个集成步骤是符合设计要求的。

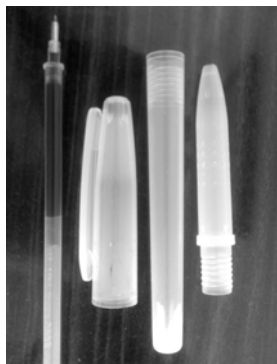


图 14-2 笔的分解结构图

这里用一支笔的组装过程模拟了软件产品的集成，以后在软件研发的过程中遇到产品集成的问题时请首先想想你手头正在使用的那支笔。

14.2 软件产品构建的流程及最佳实践

软件产品的构建过程分为产品集成和持续性的产品集成两大类。持续性的集成又是在重复产品集成的过程，因此本节将以产品集成为主，重点讲述软件产品构建的流程和最佳实践。

14.2.1 软件产品集成的准备工作

在项目计划阶段就应该完成软件产品集成的计划，在设计阶段就应该完成产品集成的方案，其方案内容至少包括以下 3 个部分。

- ① 确定产品与产品组件集成的顺序
- ② 确定集成产品或产品组件的环境
- ③ 建立产品或产品组件集成的流程与准则

1. 确定产品集成顺序

由上述“一支笔”的故事可以知道产品集成的顺序是多么重要，它是产品集成的基础。在软件研发过程中常见的产品集成顺序分为两类：一类是渐进式集成；一类是非渐进式集成。渐进式集成常用的方式有自上而下式集成与自下而上式集成。

非渐进式集成常用的方式有三明治式集成。

• 自上而下式集成

如图 14-3 所示，自上而下的集成顺序对“MVC 模型”而言可以分为以下 3 步：

STEP 01 先将表示层进行集成，集成后的半成品就好像是一个界面的原型。

STEP 02 当表示层通过测试后再将逻辑业务层集成进来。

STEP 03 在 **STEP 02** 基础上最后再将数据访问层集成进来。

• 自下而上式集成

如图 14-4 所示，自下而上式的集成顺序可以分为以下 3 步：

STEP 01 先将数据访问层进行集成。

STEP 02 在 **STEP 01** 基础上将逻辑业务层集成进来。

STEP 03 最后再将表示层集成进来。

自上而下式集成、自下而上式集成是一种渐进式的、增量的集成方式，产品功能在集成的过程中逐渐增加，通常在持续式集成模式中最常使用。

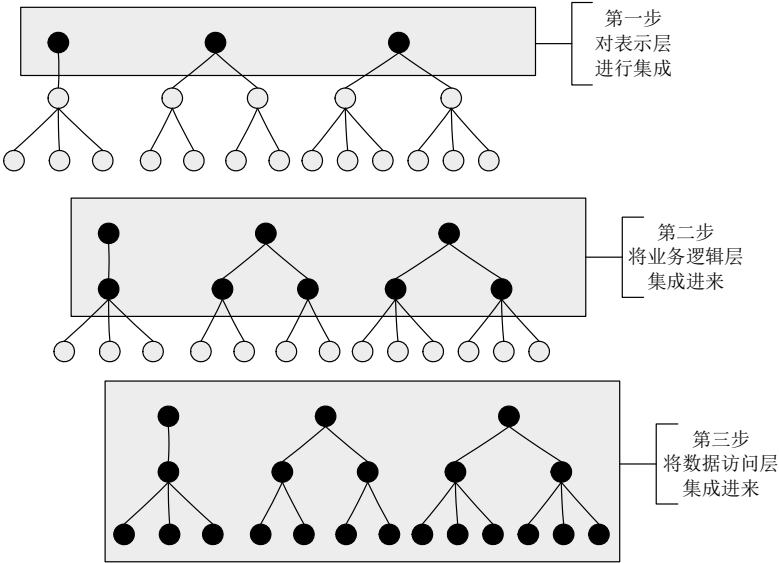


图 14-3 自上而下式集成

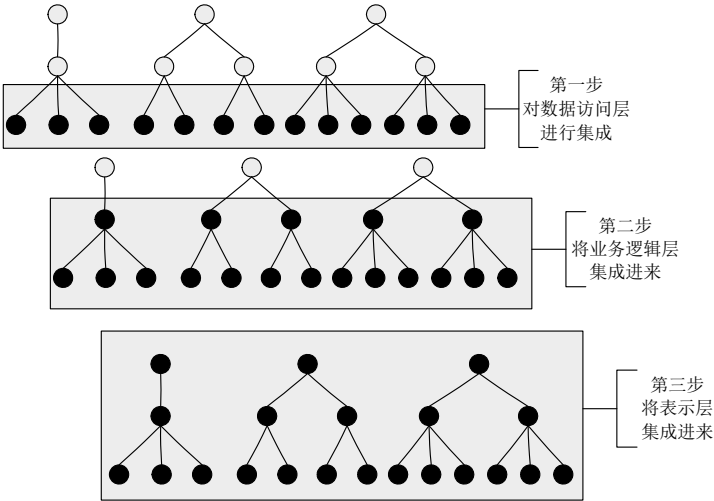


图 14-4 自下而上进行集成

• 三明治式集成

如图 14-5 所示，三明治式的集成也称为整体拼装方式或一步到位式集成，是一种非渐进式的集成，属于一次性集成的典型方式，适用于小型软件产品或者功能和技术都不复杂的软件产品，通常是在编码开发阶段后期进行的一次性努力。

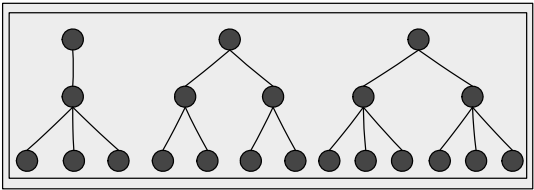


图 14-5 三明治式集成

• 不同集成方式的优缺点及对比

如表 14-1 及表 14-2 所示，以下将对不同集成方式之间的特点进行对比，并且对产品集成方式的优缺点进行分析。

表 14-1 不同集成方式之间的对比 1

	渐进式集成方式	非渐进式集成
优	<div>1. 发现问题早</div> <div>2. 对问题定位准确</div> <div>3. 测试效果更加彻底</div>	<div>1. 集成测试的工作量小，编码较少</div> <div>2. 可以采用黑盒测试的技术进行集成测试</div> <div>3. 产品集成的周期短</div> <div>4. 在集成测试中有助于全面的考虑</div>
缺	<div>1. 集成测试的工作量大，编码较多</div> <div>2. 产品集成时间较长</div> <div>3. 只能采用白盒测试技术，成本较高</div>	<div>1. 发现问题晚</div> <div>2. 无法准确定位问题</div>

表 14-2 不同集成方式之间的对比 2

集成方式	集成测试的技术	产品集成的成本	产品集成的周期	产品集成的人员
自上而下式集成	白盒测试技术需要开发测试桩	较高	较长	软件开发人员
自下而上式集成	白盒测试技术需要开发测试驱动	较高	较长	软件开发人员
三明治式集成	黑盒测试技术	较低	较短	软件测试人员

如果软件产品采用 MVC 式的开发模型，那么产品集成的顺序就要在设计阶段才能确定，因为只有等到概要设计或详细设计完成以后才能确定产品有哪些组件或模块。在制订产品集成顺序时必须非常详细，不能只说“自上而下”或“自下而上”，对其描述的颗粒度至少要精确到组件或模块的级别，这样才有指导性和可操作性。

在制订产品集成顺序时，应该首先识别待集成的组件或模块，也可以通过产品组件间的接口定义进行识别。根据已经识别出来的各种产品组件或模块，再定义集成顺序时也要识别所需要的集成测试的方法和工具，以及集成测试的环境要求，并将选择产品集成顺序的缘由记录下来以便日后备查，其流程如图 14-6 所示。产品研发的过程是个动态变化的过

程，因此产品集成的顺序也可能随时需要修订。

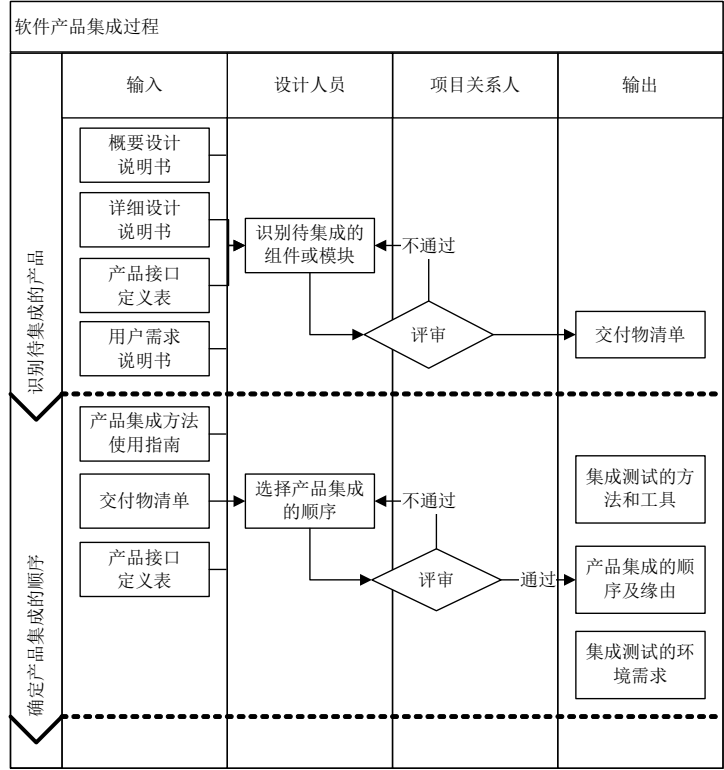


图 14-6 确定产品集成的顺序

2. 建立产品集成环境

软件产品集成的的工作大多数是由软件开发人员完成的，从软件配置管理的角度来看，软件开发人员并没有足够的权限直接去获取其他开发小组的代码或组件。另外软件开发人员本身所使用的机器中安装的开发环境、补丁或第三方的组件等都可能不尽相同，而且如果要在软件开发人员机器上进行产品集成也会影响其正常的工作。从软件测试角度来看，软件集成测试最好在软件产品集成的环境中进行，这样才能准确判断产品集成的有效性和正确性，因此产品集成的环境与集成测试的环境可以重用。所以建议用一台独立的服务器作为产品集成环境、集成测试环境和日构建服务器来使用。在建立产品集成环境时也会导致“Make or Buy”决策的产生。

在确定软件集成产品的环境后，还需要建立对该环境相应的验证和确认准则及流程，以确保软件集成的环境是符合软件产品集成需求的，其流程如图 14-7 所示。

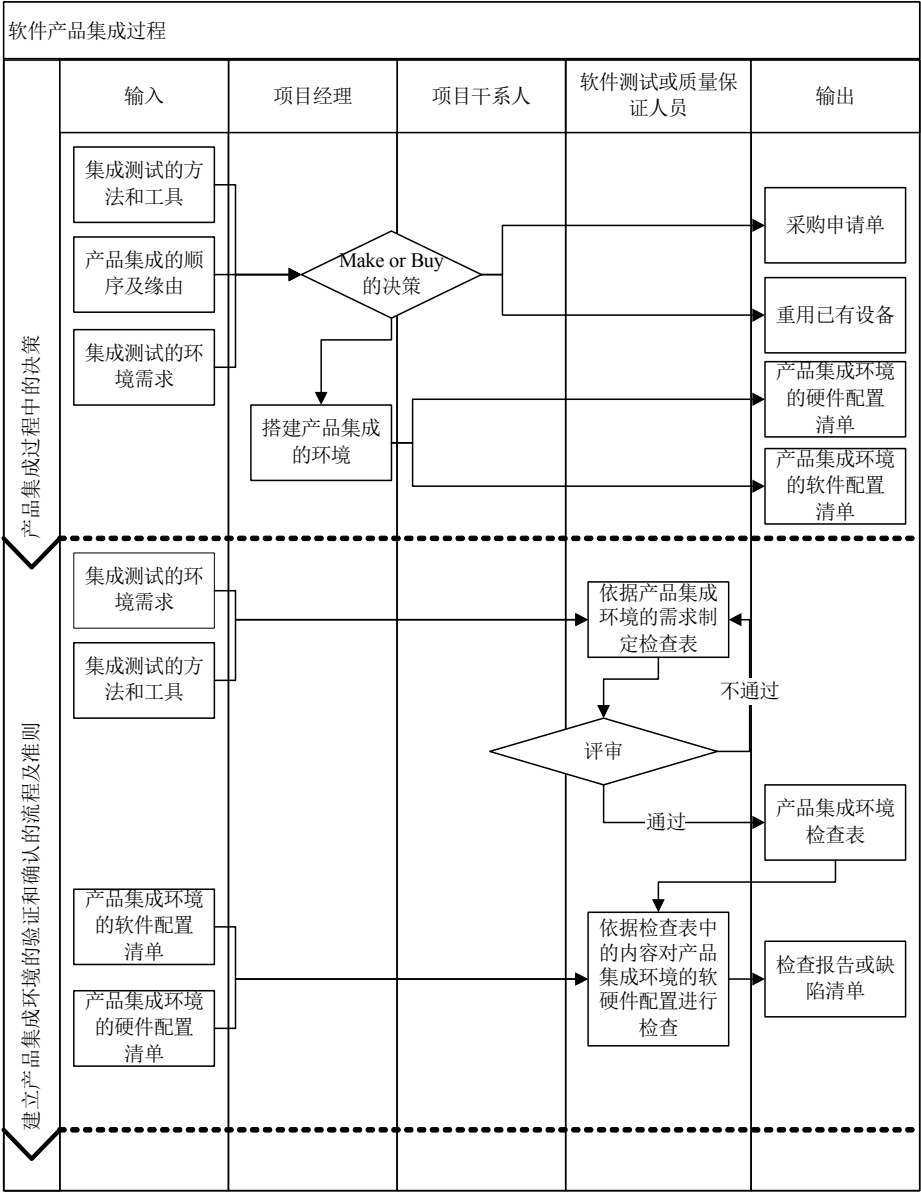


图 14-7 建立产品集成的环境

对产品集成环境的验证和确认可以通过制订检查表的方式对产品集成的环境进行检查，产品集成环境检查表的内容如表 14-3 所示。

表 14-3 某项目产品集成环境检查表

项目名称:		北极光		项目经理:		陆小凤	
QA 名称:		楚留香		审计时间:		2009-1-6	
序 号	检 查 项			审计结果	本次审计是否适用	备 注	
1	产品集成服务器的 CPU 是否是 2.0 的, 内存是否是 4GB 的, 硬盘是否是 200GB 的			√	√		
2	产品集成服务器的操作系统是否是 Windows 2003, 并且安装了补丁 2			√	√		
3	产品集成服务器的数据库是否是 SQL Server 2005, 并且安装了补丁 2			√	√		
4	产品集成服务器上是否有安装水晶报表 2.0			√	√		
5	产品集成服务器上是否有安装.NET 的开发环境			√	√		
6	产品集成服务器上是否有安装 Nunit 2.0			√	√		
7	产品集成服务器上是否有安装 IIS			√	√		
8	产品集成服务器上是否有安装 VSS 8.0			√	√		
9	待集成的产品是否已经放置到集成环境的指定目录中			√	√		

3. 建立产品集成流程与准则

产品集成的流程要与软件项目生命周期的定义相结合, 例如项目选择的迭代式开发模型, 在第一次迭代时应该先将 A、B 模块或组件进行集成, 在第二次迭代时应该将第一次集成的工作产品与 C、D、E 模块或组件进行集成。由此可见产品集成的流程与项目进度计划中任务的安排顺序是有密切关系的。

在计划好产品集成的流程后, 还要为每次集成定义相应的验证标准, 如表 14-4 所示, 明确集成后的产品或更为复杂的组件的质量要求。只有达到这个质量的要求, 本次产品集成的工作才算圆满完成。

表 14-4 某项目第二次迭代时产品集成结果的验证标准

序 号	验证标准
1	本次集成测试中集成测试用例的执行率必须为 100%, 代码行覆盖率必须达到 40%以上
2	数据在模块 A、B、C 之间流转时不能出现丢失或计算错误的情况
3	模块 A、B、C 中发现的缺陷不能影响其他模块的功能
4	模块 A、B、C 中发现的数据结构必须符合第三范式

续表

序 号	验证标准
5	模块 A、B、C 的接口必须匹配
6	模块 B 的核心算法必须经过技术委员会的评审

在定义完产品集成验证的标准后，还要为每次产品集成定义确认准入和确认准出条件，如表 14-5 所示。这是为了确认每次产品集成的准备工作是否就绪，集成后的产品或更为复杂的组件是否符合设计的要求。只有对每次集成的入口和出口进行确认，才能让大家对本次集成过程充满信心。

表 14-5 某项目第二次迭代时产品集成的确认准入和准出的条件

确认准入的条件	
序 号	准入条件
1	第一次迭代时集成的工作产品必须通过集成测试的再次验证
2	在进行本次集成时，待集成模块 A、B、C 的功能必须完成实现，并且分别通过单元测试的验证
3	待集成模块 A、B、C 单元测试执行率必须为 100%，代码行覆盖率必须达到 50%以上
4	要由项目经理对待集成模块 A、B、C 单元测试的结果和覆盖率进行确认
5	产品集成的环境已经就绪
6	软件测试人员或质量保证人员已经对产品集成环境进行再次检查，并且符合要求
7	接口是否匹配
确认准出的条件	
序 号	准出条件
1	本次产品集成后的工作成果已经通过集成测试用例的验证
2	本次集成测试中集成测试用例的执行率必须为 100%，代码行覆盖率必须达到 40%以上
3	在本次集成测试中所有发现的缺陷都已经全部修复，并且再次进行集成测试时没有发现新的缺陷

产品集成管理最佳实践“建立产品集成流程与准则”的流程如图 14-8 所示。

14.2.2 确保软件产品接口的完整性

软件项目中的接口通常可以分为两类：内部接口和外部接口。内部接口是指组件与组件之间、类与类之间为了解耦所使用的一种技术。外部接口通常是指软件产品与客户已经使用的或第三方厂商的软件、硬件和数据间的接口，目的是为了软件产品间的互联，避免“信息孤岛”的发生。

在软件研发过程中必须保持接口的完整性，但很多项目却忽略了这一点。有些项目组开始的过程一帆风顺，但是在系统测试前进行组装时却怎么也无法调试通过，报表数据的

统计和计算更是问题多多，这就是内部接口不完整造成的。有些项目自己开发得很好，过程也很顺利，但是在客户现场进行验收测试时，却无法导入客户的关键数据，这就是没有处理好外部接口的问题。

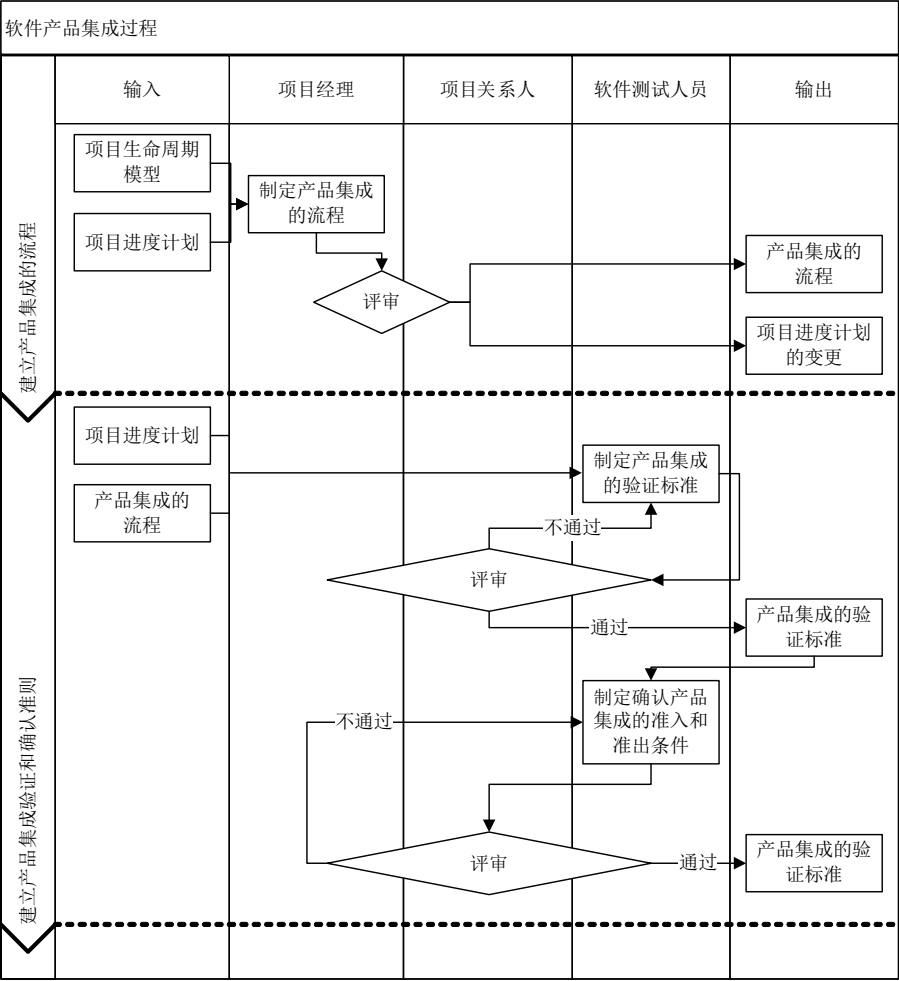


图 14-8 建立产品集成的流程和准则

如果是由于接口不匹配、不完整而造成的问题，往往都是发生在项目的中后期，这个时候留给项目的时间和资源都所剩不多，因此危害特别严重、风险也是最大的。为了确保软件产品质量的整体提高，如图 14-9 所示，项目管理人员必须把握好以下最佳实践。

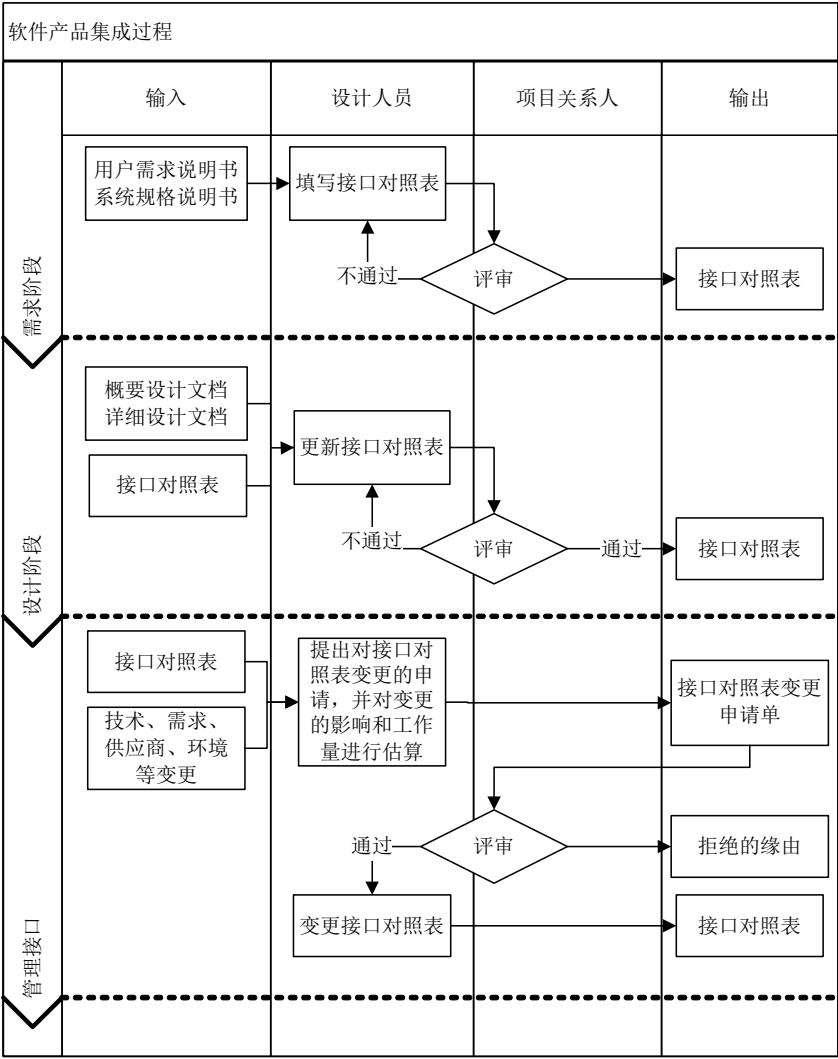


图 14-9 确保接口完整性流程图

1. 评审接口说明的完整性

在需求调研时就要对软件外部接口的信息进行调查，在软件设计时就应该对内部接口进行定义，并且要将接口定义的内容文档化，以便日后更新和维护。

软件接口的分类除了内、外部的区分外，还可以分为以下两类：

- 物理接口：软件产品与硬件设备之间的以软件技术方式实现的用于数据交互的接口

或用于硬件互联的设备。

- 功能接口：类与类之间、模块与模块之间、组件与组件之间为了实现某种功能而定义的，通过软件技术方式实现的接口。

软件项目中接口的信息通常会记录在接口对照表中，如表 14-6 所示，接口对照表中要记录接口的来源、类别、类型、参数和名称，以及使用该接口的所有组件或类。在接口对照表完成以后，项目关系人要定期对其进行的评审，以确保接口对照表内容的正确性和完整性。

表 14-6 接口对照表

接口来源	接口类别	接口名称	接口中方法的定义	使用该接口的组件或类
客 户	外 部 功 能接口	IProject	DataTable SearchProject(int projectID,int groupID, int programID,int filteringPhase);	项目产品将通过该接口与客户项目管理系统互联，以查询客户项目管理系统中的数据
			int DeleteProject (int projectAutoID, SqlTransaction transaction);	项目产品将通过该接口与客户项目管理系统互联，以删除客户项目管理系统中的数据
		IPO	int GetPO (int purchaseOrderID);	项目产品将通过该接口与客户订单系统互联，以查询客户订单的信息
设 计 人 员小李	内 部 功 能接口	IResource	int DeleteResource(int projectID, SqlTransaction transaction);	类 Project 和 Resource 中会使用到该接口来删除资源
			int InsertResource(ResourceInfo ResourceInfo, SqlTransaction transaction);	类 Resource 中会使用该接口新增资源
			DataSet GetResourcesWithRoles (int ProjectAutoID);	类 Resource 中会使用该接口查询资源的信息

2. 管理接口

项目接口的信息最早在需求调研时就会涉及，并在《软件需求说明书》和《系统规格说明书》中都有相应的定义和描述，因此对接口的跟踪和变更的工作应该从项目的早期开展。

项目是动态发展的过程，“变化是永恒的，不变是短暂的”。因此以下几种情况都有可能会导致项目接口的变更：

- ① 需求的变化
- ② 技术解决方案或设计文档的变化

- ③ 硬件供应商的变化
- ④ 第三方软件的升级
- ⑤ 产品缺陷导致的变更
- ⑥ 集成环境、系统测试环境和验收测试环境的改变

14.2.3 集成并交付产品

当产品集成的准备工作已经完成，并且对接口的完整性进行确认后，就将在产品集成的环境里按照产品集成的顺序对各个组件进行组装，以生成更为复杂的组件或最终交付的产品。

组装后的产品必须经过验证和确认后方可交付，如果是更为复杂的组件，那么通过集成测试后将交付给软件开发人员使用，如果是最终的软件产品，那么将交付给软件测试人员进行最终的测试。

如果在集成的过程中发现问题，应该及时记录并采取纠正措施。软件产品集成与交付的过程将通过以下几个最佳实践完成。

1. 确认要集成的产品组件已准备就绪

对产品集成的环境进行确认后，就要由项目的配置管理员从配置库中取出待集成的组件，并交给设计人员放置到产品集成环境指定的目录中。然后由软件测试人员对产品集成的环境进行检查和确认。

接下来在软件产品集成的准备工作中已经对待集成的组件进行了识别，在确保软件产品完整性的过程中已经对产品的接口进行了管理，那么在本过程中要做的就是依据《接口对照表》的内容检查待集成的组件与接口是否匹配。

然后由专人对待集成的产品进行逐一验证，这个验证的过程可以通过单元测试实现。最后依据产品集成确认的准入条件对检查表中的每一项进行逐一确认，通过这样的过程才能对本次集成的结果充满信心，整个流程如图 14-10 所示。

2. 集成产品组件

当确认各种条件已经就绪后，软件设计或开发人员就可以依据产品集成的顺序和流程对待集成的产品组件进行组装。

3. 评估集成后的产品组件

评估集成后的产品组件有两种方式，一种是针对组件技术实现简单且业务逻辑不复杂的集成，如图 14-11 所示，可以将所有组件在集成后进行整体一次性的评估和验证。另一

种是针对复杂组件的集成，如图 14-12 所示，需要每个组件拼装后都进行一次检查，也就是说如果有 n 个组件，那么就要进行 $n-1$ 次评估和验证。

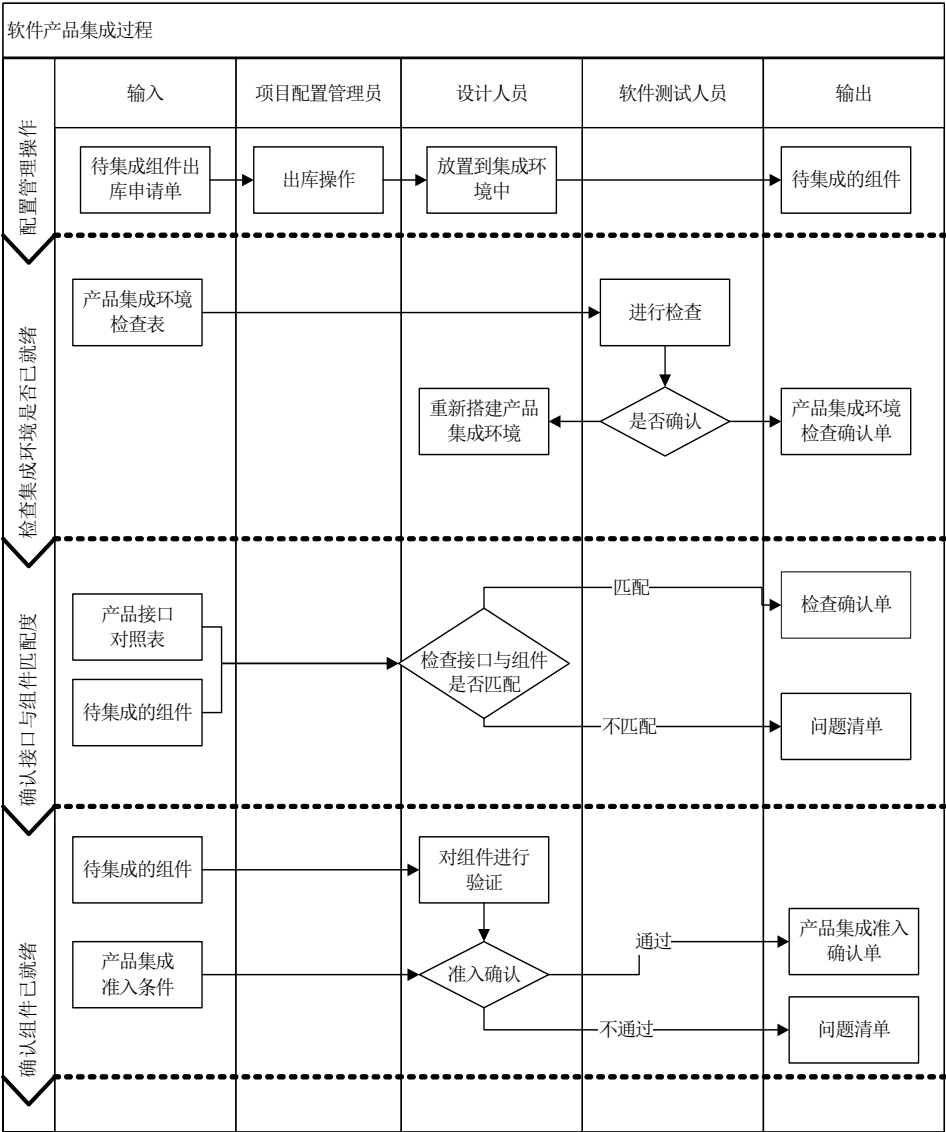


图 14-10 确认待集成的产品组件已就绪

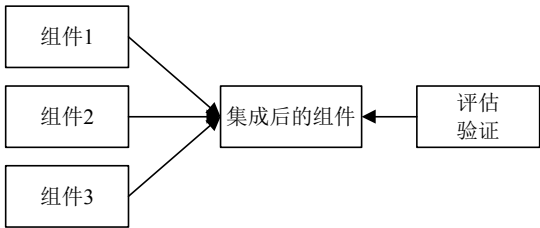


图 14-11 一次性的评估和验证

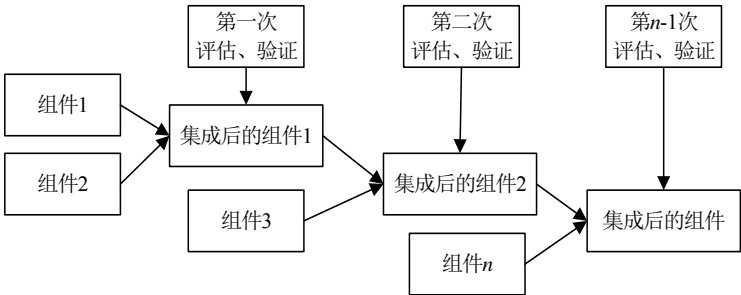


图 14-12 多次评估和验证

对集成后的软件产品或更为复杂的组件进行评估或验证的最好方法是集成测试，集成测试最好采用白盒测试，但有些特殊的产品或组件使用黑盒测试往往也可以达到效果，在测试结束后要及时记录发现的缺陷并采取纠正措施。

单元测试又称为独立测试，是针对单个函数或方法的测试。集成测试所采用的技术通常与单元测试相同，都是采用白盒测试的方法，只是集成测试的颗粒度比单元测试大许多，它不再是针对单个函数或方法的测试，而是通过一个集成测试用例对一组有关联的函数或方法进行的整体测试。单元测试的环境一般是软件开发人员自己的机器，集成测试通常会有专用的测试环境。

集成测试的主要目的是测试各模块之间的接口是否匹配、数据通过接口后是否会发生丢失、几个模块组合后的功能是否满足设计的需要、一个模块的功能是否与其他模块相冲突。实际上在概要设计完成之后集成测试的工作就可以开始了，例如制订集成测试计划、设计测试用例等。在设计集成测试用例时要知道集成测试的依据是《概要设计说明书》，而不是《软件需求说明书》或《详细设计说明书》。

以产品集成的顺序为依据，常见的集成测试方法有“自上而下的集成测试”和“自下而上的集成测试”，这两种集成测试的方式都是每次加入一个源代码模块，然后对代码进行编译并执行针对该模块和接口设计的集成测试用例。

对集成后的工作产品进行评估和验证并不能代表本次产品集成的工作已经完成，最后

相关人员还要依据产品集成测试准出条件进行检查，以确认产品集成是否圆满结束，其流程如图 14-13 所示。

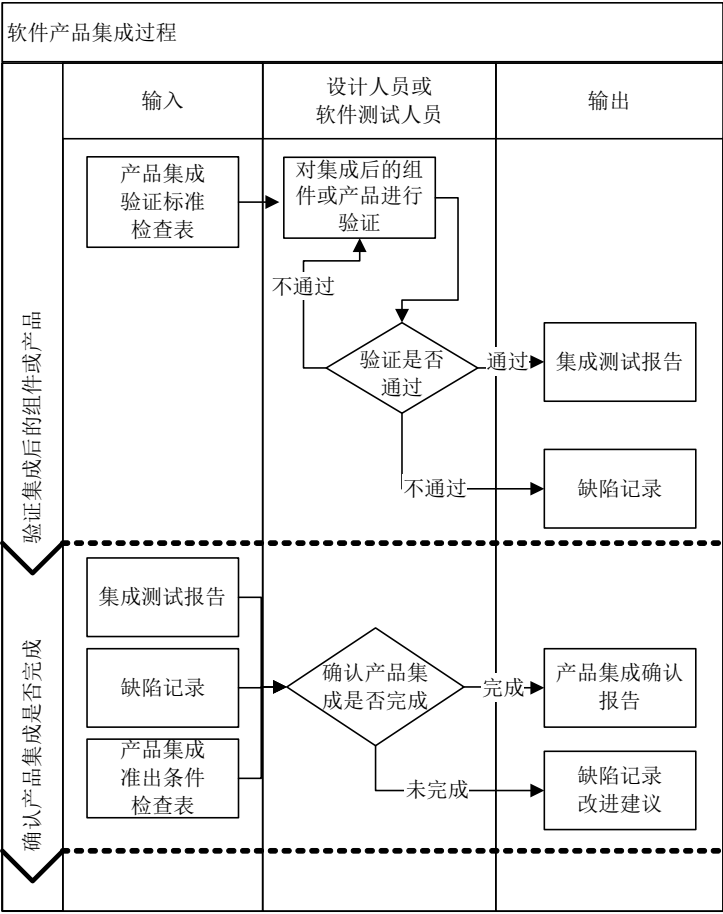


图 14-13 产品集成验证和确认流程

14.2.4 通过日构建来实现持续集成

在软件研发过程中，对项目危害最大的风险都来自于项目的中后期，而且越往后危害就越大。项目的延迟或成本的超支经常会出现在项目的中后期。试想一下在你以往的项目中是否出现过这样的场景：在编码工作结束以后，软件开发人员认为自己负责的任务都达到了质量的要求，因为全部代码都通过了单元测试。但是在产品集成时有些开发人员却发现自己的代码与某个接口不匹配，或者几个开发人员的工作成果集成后数据的流转和计算

不符合设计的要求。这个时候大家都不愿意修改自己的代码，互相推诿的情绪在逐渐蔓延，而且互相之间也越来越缺乏信任，可是此时距离项目结束却只剩下几周的时间了。随着软件规模的扩大、分工合作的加深，开发人员间的相互依赖程度越来越高，这种场景发生的几率也越来越大。

在软件质量体系中，这些影响项目整体质量的重大危害中有相当一部分可以通过持续式的产品集成在项目的中早期发现并避免，持续式的产品集成方式可以通过日构建来实现。大家对日构建这个名称应该都不陌生，当软件项目编码阶段快结束时，各个开发小组会将所有的模块整合起来，进行统一的编译，这就是最常见的构建过程，只不过它不是每天进行的。

日构建的过程就是将组件的集成工作变成一种日常行为，而不需要等到编码结束后才进行集成，这样做的好处是可以提早发现问题并解决问题。公司的各级领导也借此有更多的机会了解项目进展的真实情况，所以日构建的过程不但将产品集成日常化，而且还增强了项目的透明度和可视性，给各级管理人员更多的安全感和信心。

在软件开发过程中沟通非常重要，在项目缺少资源的时候，一些管理者认为增加若干软件开发或测试人员就可以帮助项目实现目标，这是完全不对的，因为沟通是有成本的。在软件开发的过程中，沟通的目的是为了让整个团队对有用的信息进行共享，并使其理解达成一致。日构建的一个重要功能就是给项目团队提供了一个日常沟通的渠道，大家一起总结经验、讨论问题。软件项目的团队是一个知识型的、学习型的团队，通过这些沟通和交流可以在后续的开发过程中避免问题的重复发生，从而也节约了时间和成本。这也使得团队成员间的交流更加友善。

从整个质量管理的角度对其进行分析，日构建不是一种技术，而是一种软件质量管理的方式，它可以与任何一种开发流程相整合，为其提供质量保证和信心。

在当今先进的软件工程中，日构建越来越被重视，但遗憾的是很多公司并没有真正行动起来。有些人认为日构建会过于浪费时间和资源，导致成本大量消耗。但从质量成本的角度去分析，发现问题越早就意味着解决问题的复杂度越低，解决问题的成本也越低。日构建的工作大多是在晚上做的，白天软件开发和测试人员都在忙各自的工作，晚上机器都被闲置下来，如果可以在晚上利用这些空闲的机器进行日构建，那就可以节约资源和成本。因此日构建所花费的成本和它产生的效益相比是微不足道的，从长远来看是绝对值得的。

总的来说，日构建对于加快发现和改正缺陷、降低集成风险、提高产品质量、加强成员沟通协作和信任、缩短产品上市时间、增加项目开发透明度、提高项目组成员信心和斗志都有着非常积极的作用和意义。

那么日构建到底在做什么呢？是不是日构建等同于软件开发人员对代码的编译呢？首

先日构建与代码编译是完全不同的两回事。编译代码的目的是为了发现语法的错误，以及类与类之间的调用和类与接口是否匹配。代码编译的范围可以是一个人做的也可以是一个项目团队所生产的所有代码。日构建所做的工作除了编译代码外，还要执行单元测试用例和集成测试用例，要将单元测试和集成测试的结果分发给项目关系人。更高级的日构建还要在此基础上进行业务功能的回归测试，并将通过测试的代码进行自动发布。

如果日构建是在晚上进行的，那么不是还要有人加班才可以吗？这样的话谁还愿意做日构建呢？在现在的软件工程和质量管理体系中，日构建不仅仅是一种理念，而且是通过一些工具实现的。在 Java 平台下有 Ant，在 .NET 平台下有 NAnt 和 TFS，项目组可以根据需要选择不同的日构建工具。

回想一下，在产品集成管理中有个最佳实践“集成并交付产品”，该最佳实践中的所有子实践都可以在日构建过程中自动完成。日构建的脚本可以自动执行单元测试用例来验证待集成的组件是否就绪；日构建的脚本可以自动将代码进行编译，这起到了集成的作用；日构建的脚本还可以在集成后自动执行集成测试用例，这起到了对集成后的组件或产品进行验证的功能。日构建的过程还可以将单元测试和集成测试的结果以邮件的形式发送给相关负责人，他们就可以依据邮件中的测试结果来对待集成组件的质量以及集成后组件或产品的质量进行确认。由此可见，日构建将产品集成准备工作和确保接口完整性以外的工作都通过自动化的形式来实现。自动化日构建的流程如下：

STEP 01 每晚下班后，日构建服务器会在指定时间从代码服务器获取待集成构件的最新代码到产品集成环境的指定目录中。

STEP 02 如果有需要，日构建服务器将从文件服务器中获取最新的配置文件，并将其复制到产品集成环境的指定目录中供编译使用。

STEP 03 在每晚指定时间，数据库服务器自动备份数据库到产品集成环境的指定目录中。

STEP 04 在每晚指定时间，测试用的数据库服务器会从集成环境的指定目录中获取数据库备份文件，并将其自动还原到测试数据库上。

STEP 05 日构建服务器对待集成的组件进行编译，并将编辑结果发送邮件给相关人员。

STEP 06 日构建服务器将集成后的程序发布到指定的测试服务器上。

STEP 07 测试管理服务器会在晚上指定的时间启动软件自动化测试脚本对集成后的程序进行回归测试。

自动化的日构建流程必然离不开各种各样的服务器作为支持，架构如图 14-14 所示。

服务器分工具体说明如下：

① VSS 服务器：是用于存放代码的服务器。

② SVN 服务器：是用于存放各种项目文档以及日构建过程中所使用的各种配置文件的文档服务器。

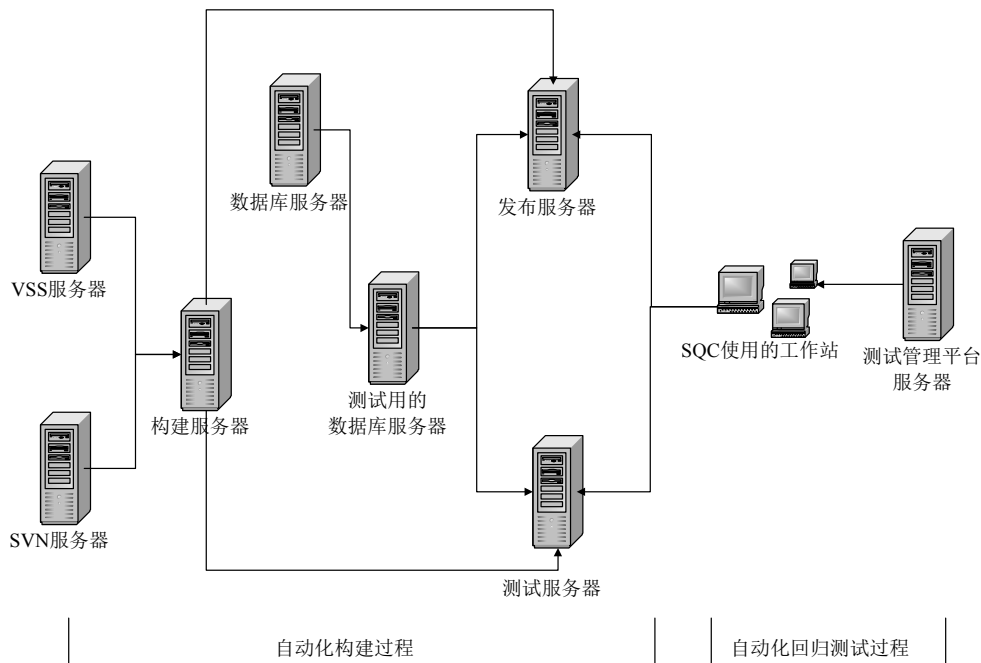


图 14-14 CART 全面的软件自动化回归测试模型

③ 日构建服务器：利用 NAnt 等日构建工具对待集成的组件或产品进行日构建工作的服务器。

④ 测试服务器：用于对集成后的组件或产品进行自动化回归测试用的服务器。

⑤ 数据库服务器：是项目团队开发时使用的服务器，其数据库中的数据是程序员在开发过程中调试程序时使用的。

⑥ 测试用的数据库服务器：是测试团队专用的数据库服务器，其数据库中的数据是测试用的。

对日构建的好处都已经非常清楚，但在日常软件研发过程中如何将日构建落实到位却有些难度，大家可以借鉴以下经验：

- ① “贵在坚持”是日构建的精神
- ② 实际可行的、详细的产品集成计划是日构建成功的保证
- ③ 开展单元测试和集成测试是日构建的基础
- ④ 对日构建理论的理解和培训是实施的前提
- ⑤ 自动化的回归测试是日构建的升华

日构建是持续集成理论的实际体现，如果要真正做到每日构建，那么就需要制订非常详细的项目进度计划、产品集成的顺序和流程，以及完备的单元测试和集成测试用例。但从软件项目的实际情况出发，不一定要每天进行构建，根据项目的周期一周一次或两周一次都是可以的，持续集成的关键是要在编码开发阶段尽早地进行产品集成，而且将集成的工作常态化。

14.2.5 日构建工具 NAnt 的使用

NAnt 是 .NET 平台下的日构建工具，它通过命令行方式运行。NAnt 是一种以 “.build” 后缀结尾的 XML 文件，一般会由以下三部分组成：

- ① <project> 标签
- ② <target> 标签
- ③ <property> 标签

NAnt 脚本范例 “Hello.build” 如下所示：

```
<?xml version="1.0"?>
<project name="Hello World" default="build" basedir=".">
  <target name="build" description="The Hello World">
    <echo message="The Hello World of build files." />
    <echo message="Welcome" />
    <echo message="This is my first sample" />
  </target>
</project>
```

NAnt 和 NAntContrib 都是开源的项目，NAntContrib 是对 NAnt 功能的扩充。在使用时可以将 NAntContrib 的 bin 目录中所有文件复制到 NAnt 的 bin 目录下，再将 NAnt 文件夹下 bin 目录的路径添加到系统的【环境变量】中，使 NAnt 在控制台程序下可以更方便地运行。右键单击“我的电脑”并选择【属性】菜单，然后选择属性窗口中【高级】标签，再单击【环境变量】按钮。如图 14-15 所示，在【系统变量】列表中选择变量【Path】并单击【编辑】按钮。在弹出的系统变量编辑对话框的【变量值】中填写“NAnt 文件夹下的 bin 目录的完整路径”，使用“;”对每个路径进行分隔。

配置完毕后用户就可以在控制台程序中输入 NAnt 指令来运行 NAnt 脚本，例如：

```
nant -buildfile:d:\hello.build -l:d:\hello.log
```

其中，【-buildfile:】指用户调用的 NAnt 脚本所在路径。

【-l:】指用户在运行 NAnt 脚本时，产生的日志文件所保存的路径。



图 14-15 设置环境变量

1. NAnt 脚本的构成

一个完整的 NAnt 脚本会由以下 3 个部分组成：

- <project>标签

NAnt 脚本是一个包含了各种构建任务的集合，用户可以把一个 NAnt 脚本想象成一个项目，所以每个 NAnt 脚本中都会要包含且只包含一个<project>标签，其语法如表 14-7 所示。

表 14-7 <project>标签常用属性

属 性 名	描 述	是否必须使用
name	项目名称	No
default	一个项目下可能有多个<target>，它标识项目运行时第一个被调用的<target>	No

范例：

```
<project name = "Hello World" default = "mybuild">
</project>
```

- <target>标签

一个<project>标签中可以包含多个<target>标签，就像一个项目可以有多个任务一样，而且在执行这些任务时<target>可以具有先后的依赖顺序，其语法如表 14-8 所示。

表 14-8 <target>标签常用属性

属 性 名	描 述	是否必须使用
name	<target>的名字	Yes
depends	该<target>的运行时依赖于哪个<target>	No
description	简单描述该 target 的功能	No

NAnt 脚本范例 Target.build 如下所示，请问以下 Target 的先后依赖顺序是什么？

```
<?xml version="1.0"?>
<project name="Target" default="TargetC" basedir=". ">

    <target name="TargetA" >
        <echo message="This is target A." />
    </target>

    <target name="TargetB" depends="TargetA" >
        <echo message="This is target B." />
    </target>

    <target name="TargetC" depends="TargetB" >
        <echo message="This is target C." />
    </target>

</project>
```

答案是：A 先运行，然后是 B 运行，最后是 C 运行。

• <property>标签

通常编写程序时都会为程序定义一些属性，这些属性通常也会被赋值。<property> 标签定义了 NAnt 脚本的属性，表 14-9 定义了该属性的值。

表 14-9 <property>标签常用属性

属 性 名	描 述	是否必须使用
name	属性的名称	Yes
Value	属性的值	Yes

范例：

```
<property name = "Web" value = "C:\Inetpub\wwwroot\MyProject " />
```

用户在 NAnt 脚本中定义了属性<property>后，在脚本中按照以下语法进行调用：

“\$ {<property>标签的名称} ”

NAnt 脚本 Property.build 如下所示：

```
<?xml version="1.0"?>
<project name="Property" default="Property">
    <property name="info" value="This is a Property" />
    <target name="Property" >
        <echo message="${info}" />
    </target>
</project>
```

2. NAnt 脚本常用的<Task>指令集

<task>指令集是由众多指令组成的，并在<target>标签下使用。如果把<target>标签理解为项目中的任务，那么<task>指令就是这些任务具体的操作行为。

• <echo>指令

在 NAnt 脚本运行时可以对运行过程添加简单的注释，以使用户对运行结果进行分析，其属性如表 14-10 所示。

表 14-10 <echo>指令常用属性

属 性 名	类 型	描 述	是否必须使用
message	String	向日志文件中写入的文字信息	No

NAnt 脚本范例 Echo.build 如下所示：

```
<?xml version="1.0"?>
<project name="Nightly Build" >
    <echo message="=====Echo=====" />
</project>
```

• <description>指令

用户对 NAnt 脚本添加描述信息便于对脚本进行阅读，其属性如表 14-11 所示。

表 14-11 <description>指令常用属性

属 性 名	类 型	描 述	是否必须使用
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No

NAnt 脚本范例 Description.build 如下所示：

```
<?xml version="1.0"?>
<project name="Nightly Build" >
    <description>
        This is my description
    </description>
</project>
```

• <call>指令

在同一 NAnt 脚本中可以有多个<target>标签，它们之间可以通过该指令互相调用，其属性如表 14-12 所示。

表 14-12 <call>指令常用属性

属 性 名	类 型	描 述	是否必须使用
target	string	被调用的<target>标签的名称	Yes
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No

NAnt 脚本范例 Call.build 如下所示，其运行结果如图 14-16 所示。

```
<?xml version="1.0"?>
<project default = "Mybuild" >
  <property name = "debug" value = "false" />
  <target name = "myinit">
    <echo message = "initializing" />
  </target>
  <target name = "mycompile" depends = "myinit">
    <echo message = "compiling with debug = ${debug}" />
  </target>
  <target name = "Mybuild">
    <property name = "debug" value = "false" />
    <call target = "mycompile" />
    <property name = "debug" value = "true" />
    <call target = "mycompile" />
  </target>
</project>
```

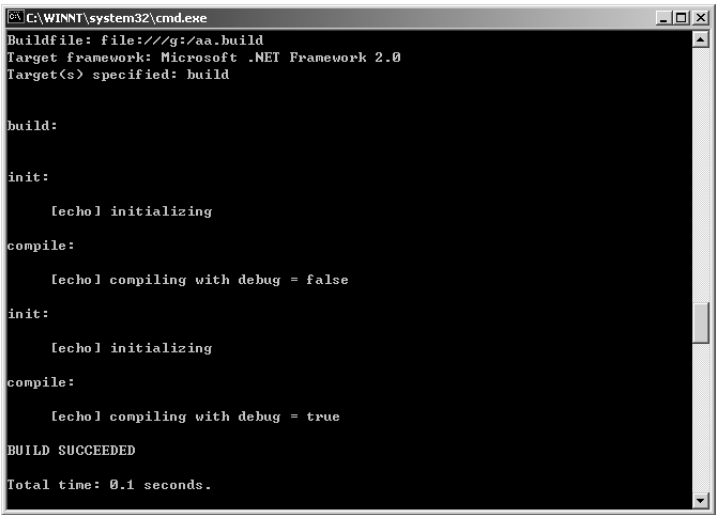


图 14-16 Call 指令范例运行结果

• <copy>指令

用户可以通过该指令将文件或目录复制到指定的文件夹中，其属性如表 14-13 所示。

表 14-13 <copy>指令常用属性

属 性 名	类 型	描 述	是否必须使用
file	String	需要复制文件的路径	No
includeemptydirs	Bool	连空目录一同复制。 默认值为 true	No
overwrite	bool	覆盖已经存在的文件。 默认值为 false	No
todir	String	目标文件夹的路径	No
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No

【嵌套元素】

<fileset>

当复制多个文件时，使用<fileset>元素。

</fileset>

范例 1: 复制 E:\MyTest.doc 文件到指定目录，如果目标文件夹下存在 MyTest.doc 文件，则将其覆盖。

```
<copy file = "E:\MyTest.doc" todir = "${Todir}" overwrite = "true" />
```

范例 2: 将 G:\doc 目录下所有 Doc 文件复制到 D:\doc 目录。

```
<?xml version="1.0"?>
<project name="Nightly Build" default="Copy">
  <target name="Copy" >
    <copy todir = "D:\doc">
      <fileset basedir = "G:\doc">
        <include name = "*.doc" />
      </fileset>
    </copy>
  </target>
</project>
```

• <delete>指令

用户可以通过该指令删除一个文件或一组文件/目录，其属性如表 14-14 所示。

表 14-14 <delete>指令常用属性

属 性 名	类 型	描 述	是否必须使用
dir	String	需要被删除的文件夹的路径	No
file	file	需要被删除的文件的路径	No
includeemptydirs	bool	在删除操作时，是否连同空目录一起删除。 默认值为 true	No
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No

【嵌套元素】

<fileset>

当删除多个文件时，使用<fileset>元素。

</fileset>

NAnt 脚本范例 Delete.build 删除 D 盘“doc”目录中所有 Doc 文件。

```
<?xml version="1.0"?>
<project name="Nightly Build" default="delete">
  <target name="delete" >
    <delete>
      <fileset>
        <include name = "d:\doc\*.doc" />
      </fileset>
    </delete>
  </target>
</project>
```

• <mail>指令

用户可以通过该指令发送 E-mail，其属性如表 14-15 所示。

表 14-15 <mail>指令常用属性

属 性 名	类 型	描 述	是否必须使用
from	string	发件人的 E-mail 地址	Yes
bcclist	string	暗送人的 E-mail 地址列表。 多个 E-mail 地址间使用“;”分隔	No
cclist	string	抄送人的 E-mail 地址列表。 多个 E-mail 地址间使用“;”分隔。	No
format	MailFormat	E-mail 内容默认为 Text 格式。	No
mailhost	string	E-mail 服务器的名称， 默认为 localhost	No

续表

属 性 名	类 型	描 述	是否必须使用
message	string	所发送 E-mail 的内容	No
subject	string	所发送 E-mail 的主题	No
tolist	string	收件人的 E-mail 地址列表。 多个 E-mail 地址间使用 “;” 分隔	No

【嵌套元素】

<files>
可以将文件的内容，作为 E-mail 的内容发送。
</files>

<attachments>
将文件作为附件进行发送。
</attachments >

Ant 脚本范例 Mail.build 如下所示：

```
<?xml version="1.0"?>
<project name="Nightly Build" default="Email">
<target name="Email" >
    <mail From = "jim@163.net"
    Cclist = "recipient2@163.net"
    Bcclist = "recipient3@163.net"
    Subject = "hello world"
    Mailhost = "smtpserver. 163.net">
    <files>
        <include name = "*.txt" />
    </files>
    <attachments>
        <include name = "*.xml" />
    </attachments>
    </mail>
</target>
</project>
```

● <solution>指令

用户可以使用该指令编译 Visual Studio .NET 2002 或者 2003 以下的项目，Visual Studio .NET 2005 的项目用户需要使用 MSBuild.exe 来实现。其属性如表 14-16 和表 14-17 所示。

表 14-16 <solution>指令常用属性

属 性 名	类 型	描 述	是否必须使用
configuration	string	设置编译的模式。是 Debug 还是 Release。	Yes
enablewebdav	bool	允许使用 WebDAV 来编译 Solution 中的文件。 默认值为 false	No
outputdir	directory	编译结果输出的目录路径	No
solutionfile	file	Visual Studio NET 的 solution 文件所在的路径	No
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No

【嵌套元素】

<projects>

需要编译的项目

</projects>

<excludeprojects>

不需要编译的项目

</excludeprojects>

<referenceprojects>

被编译的项目，引用到该项目的输出结果

</referenceprojects>

<webmap>

<map>

*该元素常用于 Web Service 的引用。

</map>

</webmap>

表 14-17 <map>常用属性

属 性 名	类 型	描 述	是否必须使用
path	file	项目文件在机器上的物理路径	Yes
url	string	项目文件使用 URL 表示的路径	Yes

NAnt 脚本范例 Solution.build 中用户以 release 模式编译 mytest.sln 文件，该 Solution

文件中包含一个 Web 项目。

```
<?xml version="1.0"?>
<project name="Nightly Build" default="Solution">
  <property name="configuration" value="debug"/>
  <property name="SolutionName" value="d:\MyNAnt\MyNAnt.sln"/>
  <property name="WebUrl" value="http://localhost/MyNAnt/MyNAnt.csproj"/>
  <property name="WebPath" value="D:\MyNAnt\MyNAnt\MyNAnt.csproj"/>

  <target name="Solution">
    <solution configuration="\${configuration}"
solutionfile="\${SolutionName}" enablewebdav="true">
      <webmap>
        <map url="\${WebUrl}" path="\${WebPath}" />
      </webmap>
    </solution>
  </target>
</project>
```

• <zip>指令

用户可以使用该指令将指定目录或文件压缩成 zip 文件，其属性如表 14-18 所示。

表 14-18 <zip>指令常用属性

属 性 名	类 型	描 述	是否必须使用
zipfile	file	zip 文件的名称	Yes
comment	string	zip 文件的注释	No
includeemptydirs	bool	在创建的 zip 文件中是否包含空目录。 默认值为 false	No
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No

【嵌套元素】

```
<files>
```

可以将多个文件或目录进行 zip 压缩。

```
</files>
```

NAnt 脚本范例 Zip.build 表示用户将 G 盘下 doc 目录中的所有文件压缩成 backup.zip 文件。

```
<?xml version="1.0"?>
<project name="Nightly Build" default="Zip">
  <target name="Zip" >
```

```
<zip zipfile = "d:\backup.zip">
    <fileset basedir="g:\doc" >
        <include name="**/*" />
    </fileset>
</zip>
</target>
</project>
```

• <unzip>指令

用户可以使用该指令解压缩 zip 文件，其属性如表 14-19 所示。

表 14-19 <unzip>指令常用属性

属 性 名	类 型	描 述	是否必须使用
zipfile	file	需要被解压缩 zip 文件的路径	Yes
todir	directory	解压出来的文件存放的路径	No
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true。	No

NAnt 脚本范例 UnZip.build 表示用户将 D 盘下的 backup.zip 文件解压到 D 盘的 doc 目录中。

```
<?xml version="1.0"?>
<project name="Nightly Build" default="UnZip">
    <target name="UnZip" >
        <unzip zipfile="d:\backup.zip" todir="d:\doc">
        </unzip>
    </target>
</project>
```

• <VSSGet>指令

用户可以使用该指令从 VSS 数据库中获取最新程序，该指令由 nantcontrib 提供，其属性如表 14-20 所示。

表 14-20 <VSSGet>指令常用属性

属 性 名	类 型	描 述	是否必须使用
localpath	directory	本地存放 VSS 数据库的物理路径	Yes
recursive	bool	是否采用递归方式获取该目录下的所有子目录及文件。 默认值为 true	No
replace	bool	确定是否替换本地可写的文件， 默认值为 false	No

续表

属 性 名	类 型	描 述	是否必须使用
writable	bool	确定这些文件的属性是否设置为可写的。 默认值为 false	No
dbpath	file	VSS 数据库中“srcsafe.ini”文件的 物理路径	Yes
path	string	VSS 数据库中目录或文件的相对路径。 该路径是从根结点“\$/”算起。	Yes
if	bool	如果表达式为 true，则执行该指令； 如果表达式为 false，则不执行该指令。 默认值为 true	No
password	string	登录 Visual SourceSafe 数据库的密码	No
username	string	登录 Visual SourceSafe 数据库的用户名	No

NAnt 脚本范例 VSSGet.build 表示用户从 VSSDB 数据库中获取最新版本。

```
<?xml version="1.0"?>
<project name="BuildSolution" default="build">
  <target name="build">
    <vssget
      user="admin"
      password="1111"
      localpath="D:\MyProject\"
      recursive="true"
      replace="true"
      writable="true"
      dbpath="\\127.0.0.1\VSSDB\srcsafe.ini"
      path="$demo/demo/"
    />
  </target>
</project>
```

● <svn-update>指令

用户可以使用该指令从 SVN 数据库中获取最新版本，该指令由 nantcontrib 提供，其属性如表 14-21 所示。

表 14-21 <svn-update>指令常用属性

属 性 名	类 型	描 述	是否必须使用
uri	string	连接 SVN 数据库访问的路径	Yes
destination	directory	客户端 SVN 目录的物理路径	No

续表

属 性 名	类 型	描 述	是否必须使用
if	Bool	如果表达式为 true，则执行该指令；如果表达式为 false，则不执行该指令。 默认为 true	No
password	string	登录 SVN 数据库的密码	No
quiet	bool	是否不打印重要的版本信息。 默认值为 false	No
username	string	登录 SVN 数据库的用户名	No

NAnt 脚本范例 SVNUpdate.build 如下所示：

```
<?xml version="1.0"?>
<project name="Nightly Build" >
  <svn-update
    Destination = "d:\test\svn\ "
    Uri = " svn://jim:5088 "
    Quiet = "true"
    Username = "jim"
    Password = "jim"
  />
</project>
```

14.3 软件产品集成管理常见问题及案例分析

软件产品集成对整个软件研发过程和质量体系都非常重要，而且与软件质量管理中的其他过程也息息相关，下面将对几个案例进行分析。

14.3.1 在配置管理下如何开展产品集成

【案例】

某软件公司网上交友项目的负责人小邱最近有些苦恼，因为公司资源有限，所以小邱除了作为项目经理外还要担任项目的配置管理员。可是小邱只接受过配置管理的培训，并没有实际的工作经验。小邱正在制订产品集成方案，他的项目将进行两次集成，可是在配置管理下应该如何对产品集成过程中的组件进行管理呢？

【分析】

配置库通常分为“开发库”、“受控库”和“产品库”，配置库是依据配置项的状态

来区分的。当进行产品集成时，软件开发人员应该对“开发库”中的代码进行编译，并运行单元测试用例来确保待集成的组件是合格的，待项目经理确认后开发人员就可以将代码“签入”到“开发库”。配置管理员接到项目经理的《交付物清单》后，就可以将待集成的组件放入“受控库”中，这样软件开发人员可以继续项目进度计划中的后续工作，而且不会对产品的集成造成影响。接下来设计人员就可以将“受控库”中的组件部署到产品集成的环境中等待集成。

由于本项目将进行两次集成，因此第一次集成后的组件在通过确认后，就可以将“受控库”中的对应代码进行替换。“受控库”会保证第一次集成后的组件不会被他人修改。

在第2次集成时以同样的方式将待集成的组件放入“受控库”，并与之前已经存放在“受控库”中的第一次集成后的组件一起进行第2次产品集成。

当第2次集成后的产品通过确认后，就可以将其放入“产品库”作为基线进行正式的发布。

14.3.2 产品集成的顺序与项目进度计划的关系

【案例】

某软件公司要开发一个绩效管理系统，项目进展到编码阶段，项目经理小田正打算安排一次产品集成，可是在查看产品集成顺序时却发现有一个组件还未完成，这让小田十分郁闷。于是小田就找来该任务的软件开发工程师小满并质问他为什么没有完工。小满十分委屈，因为他是按照项目进度计划中安排的任务进行开发的，他已经完成了他应该完成的工作。为什么该项目会发生这样的情况呢？

【分析】

项目进度计划中任务与任务之间有着四种依赖关系，一般最常用的是“完成～开始”。在产品集成方案中也会定义产品集成采用的是“自上而下”还是“自下而上”的方式，并且要精确到组件间的集成顺序。

在这两份文档中定义的时序对软件的研发都有指导作用，只有相互统一才不会发生以上项目所出现的情况。通常项目进度计划先被定义，因此在定义产品集成顺序时应该参考项目的进度计划，在必要时要对项目进度计划中任务的顺序进行变更。

14.4 小结

产品集成是软件工程中的重要环节，持续性集成是将产品集成的过程常态化、自动化。

做好了产品集成的工作就可以帮助项目组降低项目中后期严重风险的发生概率。在开展持续性集成之前，项目组必须很好地理解产品集成的过程和概念，为了加深记忆，项目管理人员可以多看看手中的那支“笔”。

14.5 思考题

1. 软件产品集成都需要做哪些准备工作？
2. 集成并交付产品的过程中都需要做哪些工作？
3. 持续性集成是通过什么方式来实现的？
4. 持续性集成对软件质量管理的好处有哪些？

15

第 15 章

软件质量的持续改进

软件的质量取决于软件过程的方方面面，例如：软件配置管理、软件项目管理、软件需求工程等。就像“木桶理论”所讲的那样，木桶盛放水的多少取决于木桶上最低的那块木板的高度。要想提高软件的质量无非是两种方式：一种是将软件公司内的所有制度都推倒重来，另一种是在原有的基础上持续地改进。

第一种方式就好比将水桶砸碎，重新买木板来新做一个更大、更好的木桶。这种方式的好处是非常彻底，并且魄力惊人，在重建“木桶”的过程中不会遇到阻力。它的缺点是对软件企业的影响太大，制造“木桶”是需要时间的，而当“木桶”还没有做好的时候大家就可能会因为缺水而死亡。

第二种方式就需要“木匠”买些材料回来修补“木桶”并逐渐加高“木桶”上的每块“木板”。这种方式的好处是大家不会出现因缺水而死亡的现象，但它的周期会较长，而且在修补的过程中会遇到各种各样的阻力。另外它需要“木匠”具有高超的修理技能，这样才可以发现并弥补木桶上渗水的地方，并在原有基础上增加“木桶”的容积。

我国改革开放采取的就是持续改进的方式。30年过去了，大家可以切身感到国家各项制度的逐步健全，而且国民的生活水平也随之逐步地提高。既然持续改进的方式都可以在国家范围内获得成功，那么在软件质量管理领域中也一定是可行的。

“好的、合适的过程才会有好的质量”，这是先进软件质量的理念，因此软件质量的持

续改进分为 3 个环节，它们分别是发现过程改进的机会、定义组织的标准过程、通过培训进行推广。既然是持续的改进，如图 15-1 所示，那么这 3 个环节也就构成了一个闭合的环形。

在软件质量的持续改进过程中，首先就要发现软件研发体系中可以改进的机会，识别过程改进的需求。根据这些需求来制订或修改组织的标准过程，然后通过培训的方式将新定义的标准过程在公司范围内进行培训，也就是以培训的方式进行推广。最后还要对重新定义的标准过程在实际工作中的执行情况进行检查，以判断本次改进是否取得预期的效果。如果还有改进的空间，那么这就为下次过程改进提供了机会。

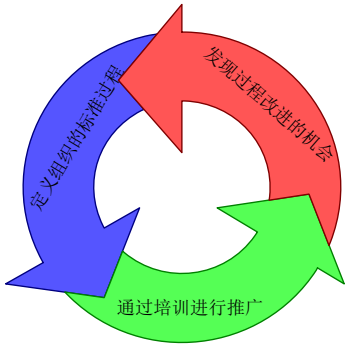


图 15-1 软件质量持续改进的 3 个环节

15.1 组织过程改进的焦点

组织过程改进的焦点就是发现过程改进的机会，其目的是充分了解当前组织过程和组织财富库中数据的优点与缺点，并计划和执行软件质量的持续改进。其中组织的过程包括了组织的标准过程和由组织标准过程裁剪而获得的已定义的项目过程。发现过程改进机会的流程如图 15-2 所示。

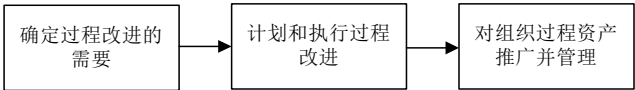


图 15-2 发现过程改进机会的流程

15.1.1 确定过程改进的需要

确定过程改进需要的流程如图 15-3 所示，过程改进小组应该定期通过与国际或行业标准和模型进行比较，并在需要的时候识别组织过程的优点、缺点与机会。

过程改进的机会应该符合依据组织商业目标制订出来的软件过程改进目标。在确定过程改进目标时可以制订一些度量的指标以将其进行量化，例如：组织级的生产率、组织研发过程的成熟度等。确定过程改进需要的方法如下：

- 与国际或行业标准进行对比发现差距，识别过程改进的需要。
- 对组织过程进行评估，以发现过程改进的需要。

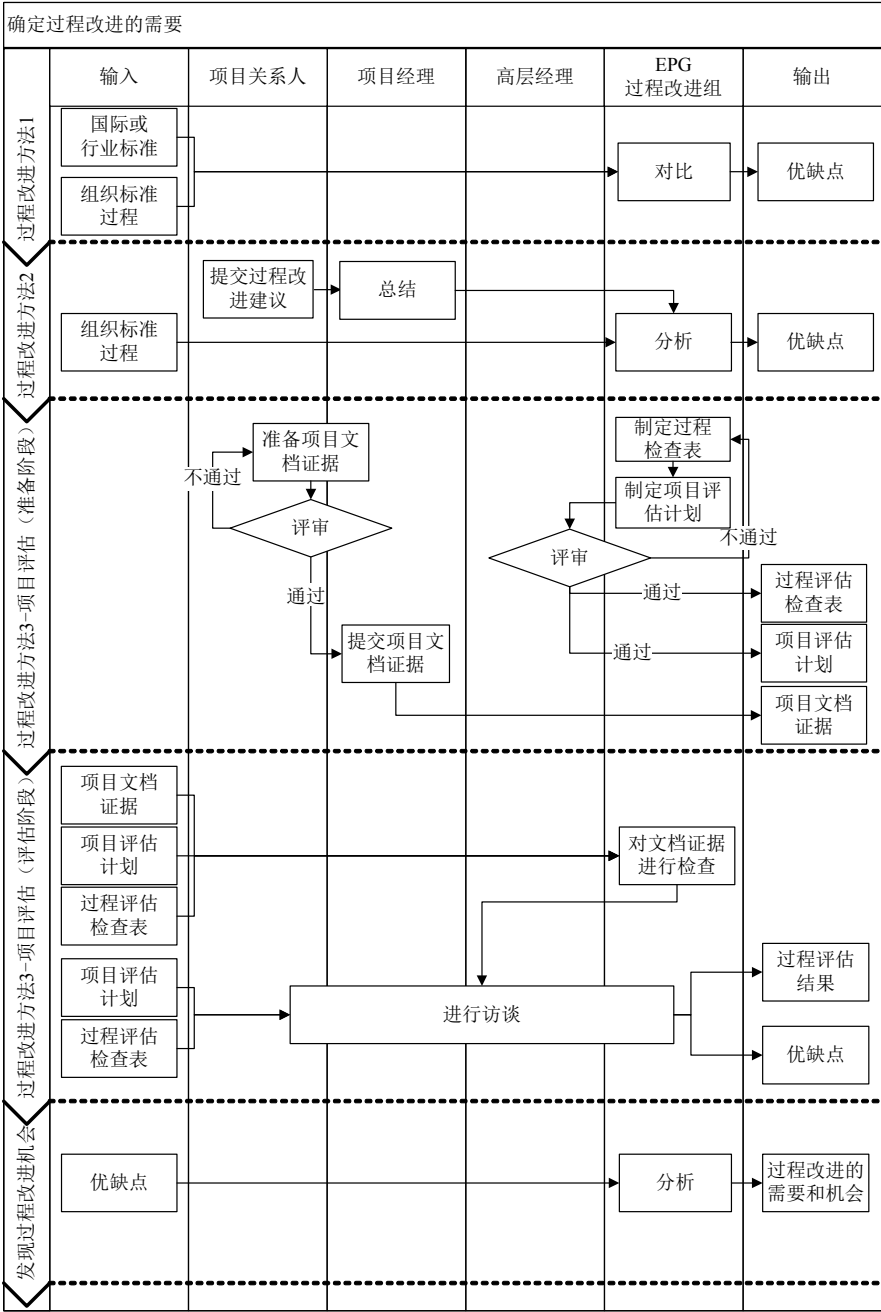


图 15-3 确定过程改进的需要

- 在项目结束后通过收集项目组成员对过程改进建议的方式来确定改进的需要，过程改进建议如表 15-1 所示。

表 15-1 过程改进建议表

xxx 项目过程改进建议表							
项目名称：	OA 项目	项目经理：	张明	建议人：	李明	建议时间：	2009-2-10
过程改进建议的内容：							
建议增强单元测试，这样代码质量会有所提高。							
是否采纳：	是	原因：开展单元测试的项目组产品质量明显高于其他项目组					
EPG 签名：	张芳						
签名时间：	2009-2-14						

过程改进小组可以在项目结束后对项目进行评估以发现项目的优缺点和改进的机会。在评估前需要建立一个评估的计划，以明确评估需要多长时间、需要哪些人参与本次评估，以及需要项目组提供哪些文档。项目过程评估计划如表 15-2 所示。

表 15-2 某项目过程评估计划

	第 1 天 2009-2-1	第 2 天 2009-2-2	第 3 天 2009-2-3	第 4 天 2009-2-4
9: 00	评估小组对评估方法进行培训	评估小组检查文档并标记问题	评估小组检查文档并标记问题	评估小组对各个过程的目标进行评估
10: 00	项目经理对项目进行介绍	培训专员访谈	软件开发人员访谈	评估小组准备评估报告
11: 00	评估小组检查文档并标记问题	软件质量保证人员访谈	软件测试人员访谈	给访谈人员（非管理类人员）进行讲评
12: 00	午餐时间			
13: 00	评估小组检查文档并标记问题			
14: 00	项目经理访谈	软件需求人员访谈	配置管理员访谈	给访谈人员（管理人员）进行讲评
15: 00	检查文档并标记问题		评估小组对各个过程进行评估	

续表

	第 1 天 2009-2-1	第 2 天 2009-2-2	第 3 天 2009-2-3	第 4 天 2009-2-4
16: 00	过程改进小组访谈	软件设计 人员访谈	评估小组对各个 过程进行评估	
17: 00	评估小组对当天评估结果 进行总结并记录	评估小组对当天评估 结果进行总结并记录	评估小组对当天 评估结果进行总结 并记录	
18: 00	预留时间			

对组织过程进行评估的方法可以参考 CMMI 标准的评估方法（Standard CMMI Appraisal Method for Process Improvement, SCAMPI），该方法分为 A 级、B 级、C 级三种，其中 SCAMPI A 级评估是一种正式的评估方法，常用于软件企业 CMMI 的正式评估过程中，该方法也支持 ISO/IEC 15504 标准。SCAMPI 评估方法是一种对软件企业研发成熟度的诊断工具，它通过对组织的一个或多个研发过程进行诊断以发现组织现有过程的强项、弱项和改进的机会，帮助软件企业对自身单个过程的能力和整体成熟度进行全面的了解。在 SCAMPI A 级评估的过程中会成立一个评估小组，该评估小组一般会由 5~10 人组成，通常该小组由 SEI 的主任评估师、企业的咨询顾问、企业过程改进小组的代表，以及第三方公司的过程改进代表共同组成。

1. SCAMPI A 级评估方法

该方法是由 SEI 授权的主任评估师和接受过 SAT（SCAMPI Appraisal Team）培训并在 SEI 注册的过程改进小组成员组成，通过对项目工作产品的检查和对项目组成员进行广泛的面谈来评判软件企业研发的成熟度。

2. SCAMPI B 级评估方法

该方法的目的与 SCAMPI A 级相同的地方都是为了找到企业研发过程中的强、弱项和改进机会，但 SCAMPI B 方法并不试图对企业的某个过程或组织整体的成熟度进行评定。这就是推荐给大家的，在软件企业内部自己开展的一种日常评估方式。

3. SCAMPI C 级评估方法

该方法的目的是快速对组织整体成熟度进行判断，以确定其是否可以进行 SCAMPI A 级的评估。

软件过程的改进是一件持续性的工作，而不是为了获取 CMMI 的证书。软件企业应该学会 SCAMPI B 级的评估方法，在企业项目结束后自己组织进行内部的评估，这是软件质量持续改进的重要手段和方法。

在评估计划完成后就需要制订本次评估的检查表，它是在对项目评估过程中使用的。检查表的内容要以该项目的过程定义书为依据，以项目裁剪后已定义的过程为标准。在对组织进行评估时以检查文档证据和访谈相结合，当文档证据不足时则可以通过访谈进行审查，在访谈时必须覆盖到项目组的所有角色和职责。对访谈和检查文档过程中发现的证据要逐一记录在检查表中，并为每项过程进行打分，检查表的大致内容如表 15-3 与表 15-4 所示。

表 15-3 对项目监控过程进行评估

实践活动	证 据	证据的种类	证据类型	评 分
对照项目计划监督项目的进度	项目进展报告； 与项目经理访谈中有回答项目监控的流程和“项目进展报告”的使用情况	文档和访谈证据	直接证据	70
对照项目计划监督项目的成本				65
对照项目计划监督项目的质量				60
对照项目计划监督项目的资源				69
对照项目计划监控项目人员的知识与技能				70
对照项目计划监督其中指定的允许				69
对照项目计划监督其中识别的风险				68
对照项目计划监督项目数据的管理				67
对照项目计划监督关系人的参与情况				65
对发现的问题制订纠正措施				60
执行纠正措施直到问题得到解决	该项目缺陷管理平台中记录的所有缺陷； 项目组成员都有讲述使用缺陷管理工具来对项目缺陷进行管理	文档和访谈证据		60
该过程得分：	65.72 分			

表 15-4 对项目风险管理过程进行评估

实践活动	证 据	证据的种类	证据类型	评 分
确定风险来源和类别	项目没有风险列表； 在访谈中项目经理有谈到风险管理的过程，但不是很正确	文档和访谈证据	间接证据	35
定义风险的参数				35
建立并维护风险管理的策略				35
将风险文档化				35
根据已定义的风险类别、参数对每个已识别的风险进行评估并分类，同时确定其优先级				35

续表

实践活动	证 据	证据的种类	证据类型	评 分
制订风险应对计划	项目没有风险列表；	文档和访谈证据	间接证据	35
定期监控每个风险的状态并执行适当的风险应对计划	在访谈中项目经理有谈到风险管理的过程，但不是很正确			35
该过程得分：	35 分			

在表 15-4 中可以看到证据的类型分为直接证据和间接证据，当没有直接证据可以证明时就需要间接证据来补充，否则就证明该过程没有执行。例如：如何证明小明请小李去吃饭了？出去吃饭的文档类直接证据是餐饮发票；访谈类直接证据是小李说：“那天晚饭是小明请的”；如果没有餐饮发票和小明的证词，那么还可以通过间接文档类和访谈类证据来证明，只是不如直接证据那么充分而已。例如，出租车发票也可以间接证明小明和小李在晚上几点去吃的饭；的士司机的证词可以作为间接的访谈证据证明他们一起去吃饭。

通过对每个过程的评估可以得到一个项目整体的成熟度，也就可以找到过程改进的机会。如图 15-4 所示，某项目风险管理和产品集成做得不够好，如果该项目具有代表性，那么过程改进的需要就是提高风险管理和产品集成管理的能力，过程改进的目标就是在今后将风险管理和产品集成的能力提高到 70 分以上。

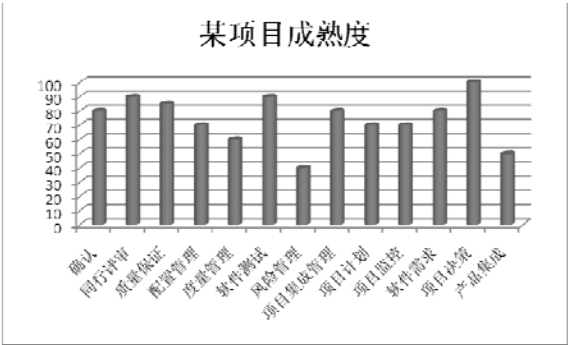


图 15-4 某项目评估结果

15.1.2 计划和执行过程改进

当过程改进的需要和目标都制订出来后，就要开始计划和执行过程改进。过程改进的计划分为 4 个部分，其流程如图 15-5 所示。

- 过程改进行动计划

- 试用计划
- 项目推广计划
- 组织标准过程培训计划

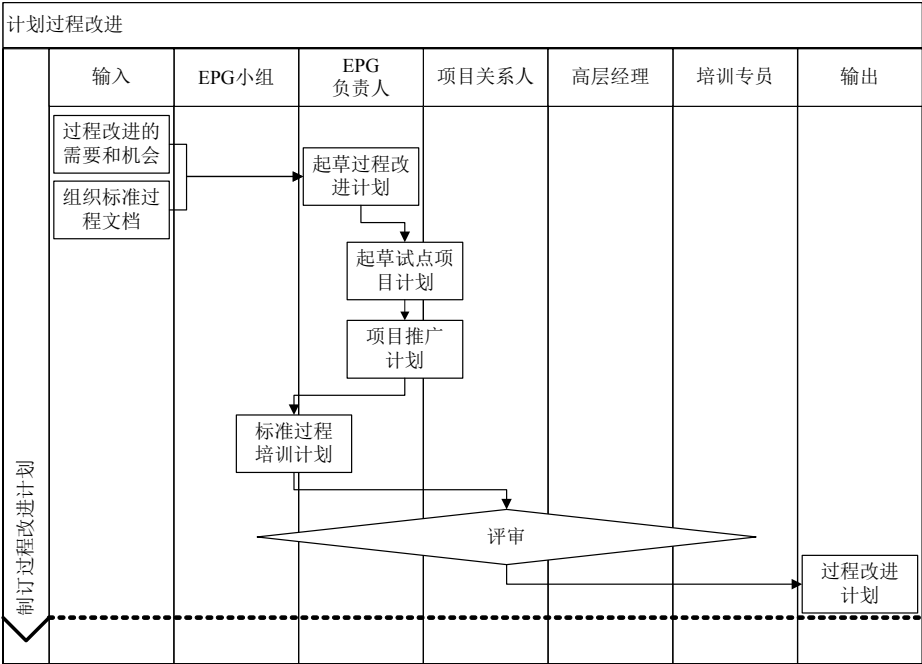


图 15-5 计划过程改进的流程

1. 过程改进行动计划和试用计划

过程改进的行动计划通常由评估的结果导出，并且以评估时所发现的缺点为目标，制订如何进行过程改进的计划。制订过程改进计划的方法与制订项目计划的方法和流程相同，如有需要请参考本书第 10 章的内容。在过程改进行动计划中要明确以下内容：

- 制订过程改进的目标
- 有哪些组织标准过程文档需要修改，由谁修改，谁来审批。
- 制订修改组织标准过程文档的时间表
- 过程改进行动的风险
- 试用计划
- 假设与约束

因为组织级的标准过程文档是公司内所有项目都必须参考和引用的，如果制订后就在公司内直接进行推广，一旦发现标准过程中存在隐患或不具有可操作性，那么就会导致项

目无法正常进行，给组织带来重大负面影响。因此需要选择项目先进行试用，对试用过程中发现的问题进行及时修改，最后才在全公司范围内进行推广，其流程如图 15-6 所示。这好像我国改革开放的初期要选择几个城市作为特区的目的的一样。

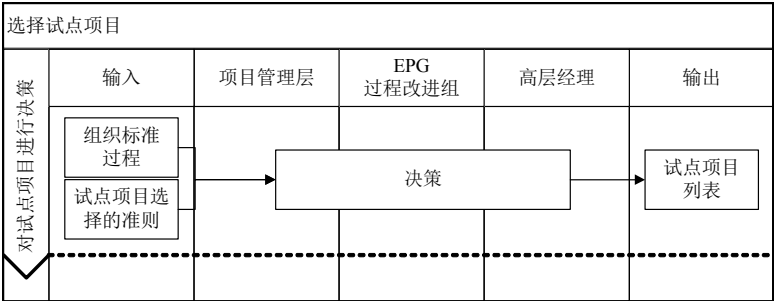


图 15-6 试点项目选择的流程

试点项目的选择可以参考本书第 13 章介绍的方法和流程进行，因为选择试点项目是过程改进中的一个重大决策，通常在对它进行决策时会参考如表 15-5 所示的准则。

表 15-5 对试点项目进行决策的准则

序号	准则内容	缘 由	是否是必须条件	权 重
1	该项目是否代表了公司的主营	内部环境	必选条件	20%
2	该项目是否采用了公司的主流开发技术	技术约束	必选条件	20%
3	试点项目的规模是否具有代表性	内部环境	必选条件	20%
4	试点项目的生命周期模型是否具有代表性	内部环境	必选条件	20%
5	试点项目是否能覆盖到组织标准过程的各个方面	内部环境	可选条件	20%

在试点项目结束后，过程改进小组还需要利用本节中介绍的评估方法来对组织标准过程进行评估，这个过程既是确定过程改进的需要，也是对组织标准过程可行性的一个检验，如果达到预期要求则可全组织范围进行推广。

2. 项目推广计划

在制订项目推广计划时需要考虑两个部分，一个是确定推广的时机，一个是如何进行推广。

• 确定项目推广时机

在根据过程改进需要制订过程改进计划时，以及在试点项目进行时，公司内的其他项目不会停止正常的工作。因此过程改进小组需要知道公司内所有项目的进展情况，以及未来还会有哪些新的项目。

另外，正在进行的项目裁剪了哪些组织标准过程也是必须清楚的，这是组织标准过程

推广的一个重点和难点。例如：某个软件外包项目裁剪掉了软件需求工程的内容，那么在该项目进行的过程中推广组织标准过程时就不能包括软件需求工程的内容。

组织标准过程对正在进行中的项目推广的时机有两种：一种是在项目进行中去推广，通常这种方式项目组的抵触情绪会很大，导致推广的难度加大，除非要推广的内容是该项目组迫切需要的或者就是该项目组自己提出的，否则不建议采用该方式；另外一种组织标准过程只在新项目中进行推广，这种方式不会遇到太多来自项目组的阻力，但组织标准过程使用的及时性稍有欠缺。推广计划的内容和格式如表 15-6 所示。

表 15-6 2008 年 8 月 10 日组织标准过程推广计划

组织标准过程	项目 A 2008-6-9—2008-10-9		项目 B 2008-8-10—2009-1-10		项目 C 2008-7-10—2008-9-10	
	已定义的过程	计划推广的内容	已定义的过程	计划推广的内容	已定义的过程	计划推广的内容
确认	√			√	√	
验证	√			√	√	
同行评审	√			√	√	
质量保证	√			√	√	√
配置管理	√	√		√	√	
度量管理	√			√	√	
风险管理	√			√	√	√
项目集成管理	√			√	√	
项目计划	√			√	√	
项目监控	√	√		√	√	
需求工程	√			√		
决策分析	√			√	√	
产品集成	√			√	√	
推广时间	项目即将结束，但配置管理和项目监控是项目组提出急于改进的，因此会在 2008 年 8 月 12 日进行推广		项目刚启动，所有组织标准过程就会在该项目进行立即推广		项目即将结束，不再对其进行推广	

● 如何进行推广

过程改进小组的负责人最好可以开会向全体员工介绍本次将要推广的内容、过程改进的需要、推广的计划、过程改进的目标，以及评判推广效果的标准和方法。

在推广的过程中 EPG 通常会采用培训的方式来进行推广，由 EPG 各标准过程的负责

人制订培训材料和培训考核标准，并起草组织标准过程培训计划，最后由培训专员来安排具体的培训时间的和地点，培训的相关内容可以参考第 15.3 节。

3. 执行过程改进

在执行过程改进计划的时候可以将它作为一个项目，过程改进小组的负责人就是这个项目的项目经理，过程改进的关系人都是该项目组的成员，其工作流程如图 15-7 所示。过程改进小组的负责人可以参考本书第 11 章的内容对过程改进计划的执行情况进行监督。

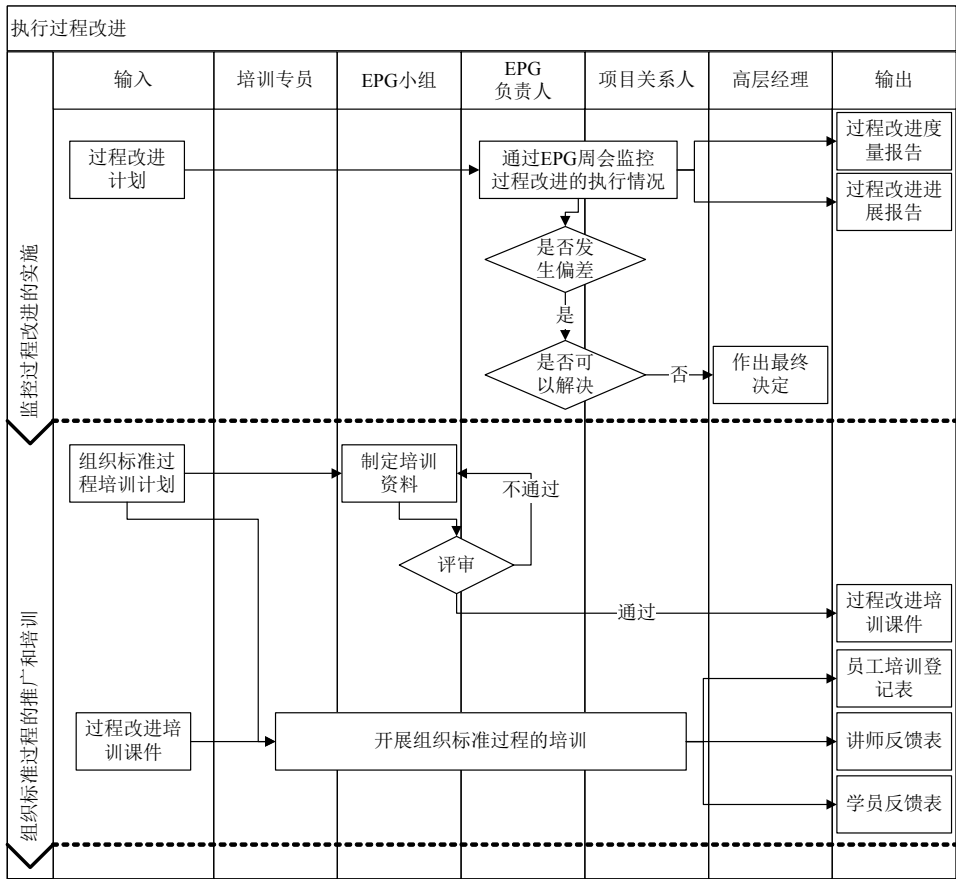


图 15-7 过程改进的执行流程

在执行组织标准过程推广培训计划时，过程改进小组负责制订培训资料，培训专员负责培训的安排和协调工作，过程改进的关系人需要按时参加培训。在培训结束后培训专员会将培训的情况进行登记，讲师和学员也需要填写相关反馈表来收集培训度量的基础数据。

在过程改进的计划和执行过程中会有很多经验和教训值得总结，还有过程改进小组成员以及项目团队在过程改进工作上的度量信息需要总结。如图 15-8 所示，这些信息都可以通过 EPG 周会进行收集和总结，在周会上与会人员还可以提出对过程改进计划和执行方面的建议。这些度量数据和过程改进的信息都可以由组织级的配置管理员提交到组织财富库中进行保存。

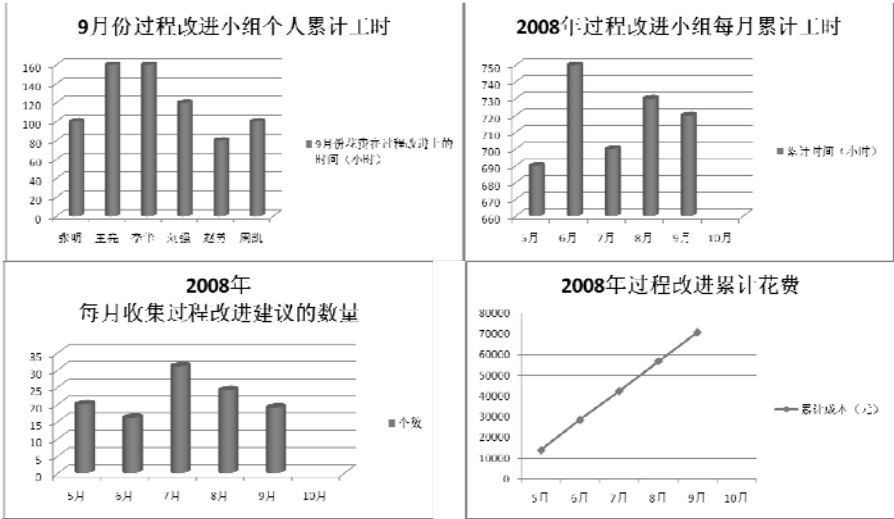


图 15-8 过程改进度量数据

15.2 定义组织标准过程的最佳实践

组织过程定义的目的在于建立和维护可用的组织过程资产。组织过程资产可以使整个组织有统一的过程用于参考和执行，并且是组织累积性、长期性效益的基础。

15.2.1 建立组织级标准过程和组织财富库

1. 组织财富库的介绍

组织财富库的建立、维护和使用同样属于配置管理的范畴，它也需要借助配置管理工具进行操作权限的管理，也需要对组织标准过程的配置项进行识别，制订配置项和基线的命名规则同样需要对基线的发表和变更进行管理以确保组织标准过程的完整性。具体可以参考本书第 6 章的内容。

组织财富库是收集数据的地方，由组织级配置管理员建立和维护，并提供给组织所有

项目使用。组织财富库所收集的数据包括：

- 标准过程与过程内容的说明
- 软件生命周期模型的说明
- 过程裁剪指南
- 项目组已定义的过程和相关文档及数据
- 最佳实践与心得体会
- 培训数据
- 组织级风险库
- 组织级度量库
- 过程改进数据库
- 组织级工作环境的描述

以 CMMI 3 级为例，组织的标准过程有 18 个，每套标准过程都应该包含以下内容：

- 过程中每个步骤的流程以及流程的详细说明
- 过程实施的指南
- 该过程的检查表
- 该过程的模板
- 该过程的使用范例

根据以上信息，组织级配置管理员就可以创建组织财富库，其目录结构如图 15-9 所示。

2. 建立过程改进小组

组织标准过程的建立是由过程改进小组负责并完成的，通常过程改进小组会有 1~3 名全职人员和若干个兼职人员组成，其中会选取 1 名过程改进的负责人。过程改进小组的兼职人员来自于各个项目组，他们按照所擅长的领域进行分组来制订相关的组织标准过程。在制订组织标准过程前，过程改进小组的所有成员要接受相关软件工程体系或国际标准的培训。

3. 建立组织标准过程

在一个公司内多套标准过程可能同时存在，以满足不同的应用领域、生命周期模型等的需要。这些标准过程通常包含技术、管理、行政、支持及过程改进等过程，它必须覆盖组织和项目所需的全部过程。

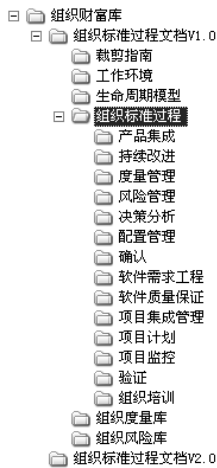


图 15-9 组织财富库目录结构

在制订标准过程文档时首先要将过程分解为多个子流程，画出每个子流程的流程图，并用文字详细描述每个步骤以及步骤之间的逻辑关系。在每个子流程中都要明确其重要属性，常用的属性有：

- 每个子流程中的角色
- 适用的范围和标准
- 子流程所需要的资源和工具
- 准入条件
- 准出条件
- 输入的文档
- 输出的文档
- 需要收集的度量
- 验证点（例如：同行评审、单元测试）
- 接口
- 软件质量保证人员检查的内容和时机

依据每个子流程中的输入和输出文档、评审环节等，可以制订软件质量保证人员的检查表。根据标准过程的内容、流程和所需要实现的目标制订相关文档模板，为了更好地让使用者正确使用该流程、文档模板、检查表等资料，标准过程的制订人员还需要编写该过程的使用指南和范例。最后经过评审就可以发布组织标准过程的基线。建立组织标准过程的流程如图 15-10 所示。

15.2.2 定义生命周期模型

在对项目进行整体策划前要根据公司的特点选择一些项目生命周期模型供项目组挑选。软件生命周期模型有很多，下面将为大家重点介绍微软 MSF 生命周期模型和 XP 极限开发模型。

1. 微软 MSF 生命周期模型

微软 MSF 开发模型的全称是 Microsoft Solution Framework，是微软从公司自身、合作伙伴以及客户那里总结的一套研发的流程，是微软产品的研发指南。

传统软件开发的流程大致是签署合同、需求调研、项目计划、软件设计、编码和测试，最后是验收和交付。但在该过程中软件项目组只关心如何交付产品，而对是否能够实现客户的商业目的并不在意。客户投资开发一套软件产品的目的不是为了要这个软件，而是希望通过该软件可以实现其商业目的。微软 MSF 却把客户的商业目标放在首位，整个生命周期都是围绕该目标去发展和细化的。该流程很好地弥补传统软件生命周期模型的不足，并

为企业在竞争日益激烈的情况下保持优势，它能给软件项目带来以下提升：

- 使研发流程更加有效

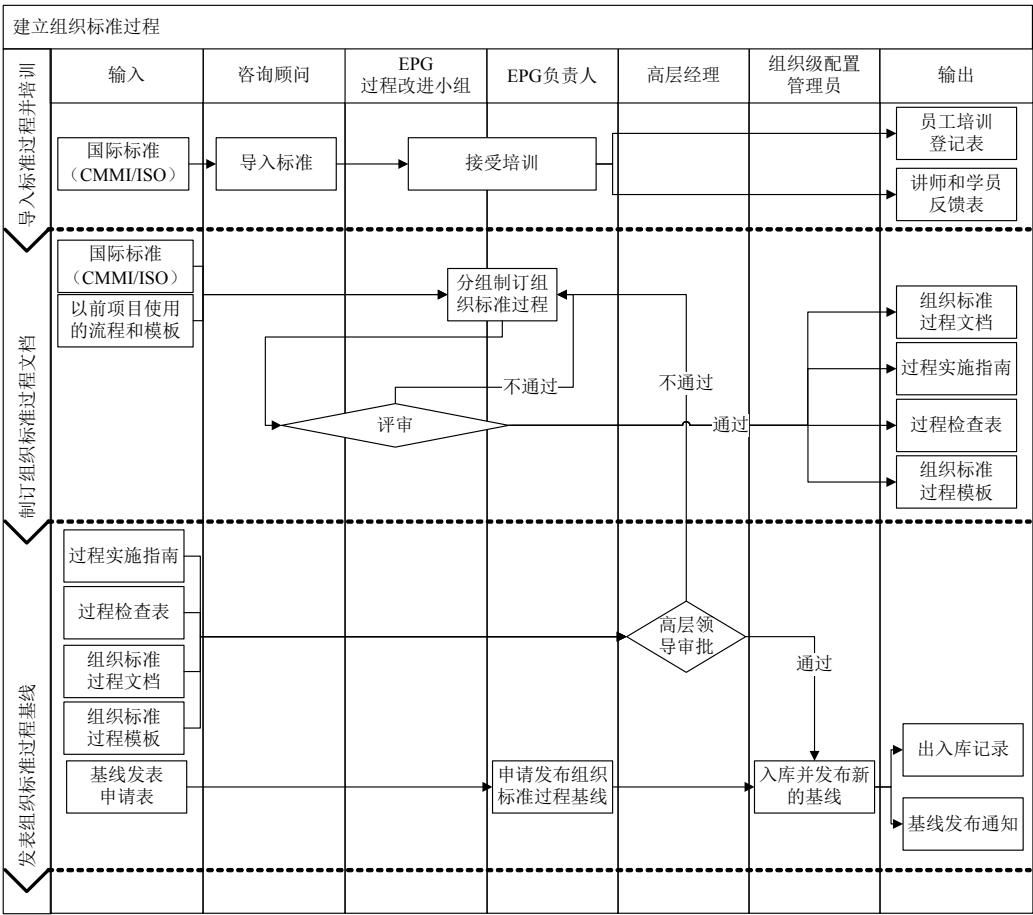


图 15-10 建立组织标准过程的流程

- 反应和决策能力更加迅速
- 在使用多层体系架构的同时又能满足不同层次、不同用户或组织的需要
- 是软件工业化的基础

如图 15-11 所示，MSF 将软件生命周期的组队模型分为以下 6 种核心角色。

① 产品管理角色：他是客户的代言人，他对客户的特征、行业特点都非常了解，是需求调研、把握产品需求方向的负责人。他要确保清晰地描述客户的期望值，并且使项目团队成员都能正确理解。产品管理角色有时也负责项目的立项、合同谈判等沟通和协调的工

作。产品管理角色不一定要具备软件开发的技能，但他要对软件技术有深刻的理解并确定哪种技术会给客户带来潜在的收益。

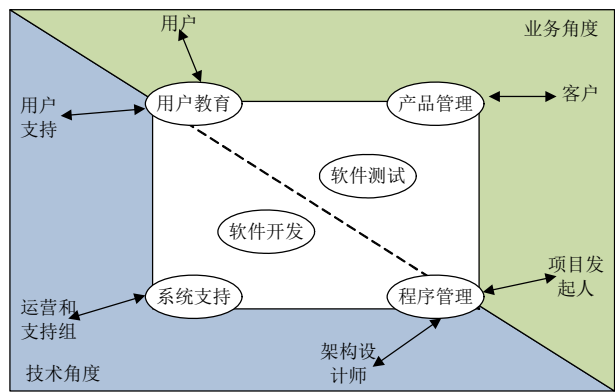


图 15-11 MSF 角色关系示意图

② 程序管理角色：他负责制订项目计划，并对项目风险进行管理，确保计划按质、按量完成。程序管理角色的人员要具备很强的技术实力，有时也充当架构设计师，他要与软件开发工程师一起对关键技术进行决策。

③ 用户教育角色：他负责设计出友好的界面，并对客户进行培训，确保客户喜欢并愿意使用该产品。用户教育角色通过演示和培训的方式最大限度地让产品使用者得到相关产品和服务所带来的价值体验。

④ 软件开发角色：他会尽早参与需求的分析和项目计划的制订，并按照计划构造或实现满足客户需要的产品或服务。

⑤ 软件测试角色：负责测试脚本的开发，确保在产品交付前能够发现所有产品的缺陷，跟踪缺陷状态直到关闭。

⑥ 系统支持角色：确保产品平稳过渡、安装和交付到客户手中。

MSF 组队模型只是一个模型，而不是项目的组织结构，软件公司也为项目团队指派一名项目经理，项目经理将会与这 6 个小组的组长共同工作。

MSF 生命周期模型中定义了 5 个阶段，如图 15-12 所示，每个里程碑都是一个阶段的终点，每个阶段都在周而复始地循环，直到软件产品符合用户的要求，下面分别对这 5 个阶段进行详细说明。

● 构思阶段

构思阶段结束的标记是“前景/范围确认”里程碑。在这个阶段所要完成的是项目启动和软件需求工程所规定的内容，里程碑评审是客户和项目组就项目的前景和范围达成一致

的机会。构思阶段还可以设置的子里程碑如图 15-13 所示。

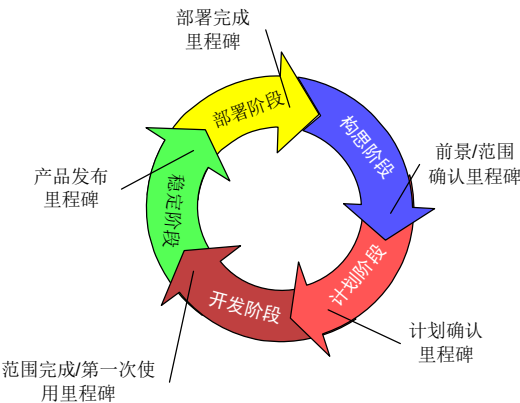


图 15-12 MSF 生命周期阶段及里程碑设置

确认前景和范围是两个相对独立的活动的，也是软件项目中必不可少的活动，前景描述了产品或服务的最终目标，为项目的发展指明了方向。但范围却是前景的约束，任何一个项目如果没有界定范围，那将是非常可怕的事情。

如图 15-14 所示，在构思阶段需要提交以下工作产品：

- ① 前景陈述文档是由项目团队成员和客户共同提出，并由产品管理角色创建的对软件产品或服务所提供目标的描述。
- ② 设计目标文档提供了产品范围的描述，由程序管理或产品管理角色创建。
- ③ 组织架构文档定义了项目组的组织架构。
- ④ 风险管理文档是一份在本阶段创建，在后续阶段动态更新的文档。

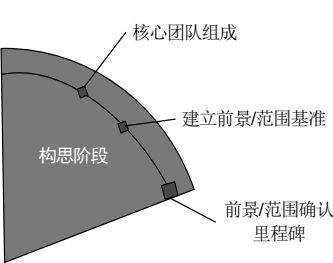


图 15-13 构思阶段里程碑

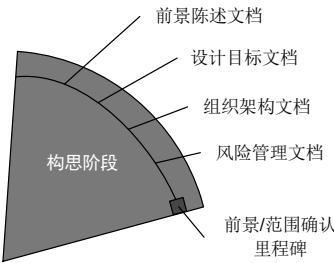


图 15-14 构思阶段工作产品

• 计划阶段

计划阶段结束的标记是“计划确认”里程碑。这个阶段需要客户和项目组就交付的内容、开发的优先级和期望值达成一致意见。里程碑评审是重新评估风险、建立优先级、对

项目进度、项目资源协调情况做最终评估的重要机会。如图 15-15 所示，计划阶段还可以设置以下几个子里程碑。

在这个阶段中，还需要将客户的期望分解成一系列的业务逻辑、用户服务和数据服务，并确定产品的界面、架构设计方案、接口等活动。项目组中的每个人都要提供一份自己的计划和时间进度文档，用于描述如何构建功能规定中的产品。如图 15-16 所示，在计划阶段需要交付以下工作产品：

- ① 功能定义描述了交付的产品将实现什么功能。
- ② 项目计划是项目团队中每个人各自计划的集合。
- ③ 项目的进度计划。
- ④ 更新后的风险管理相关文档。

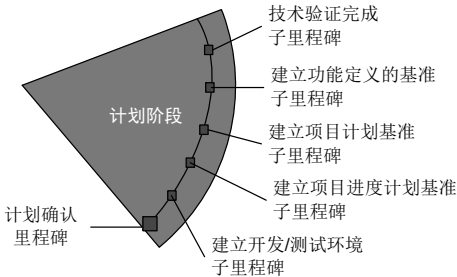


图 15-15 计划阶段里程碑

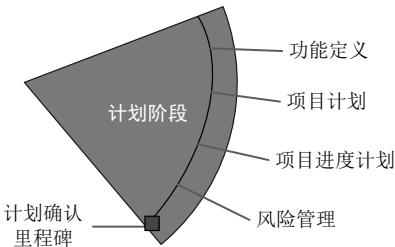


图 15-16 计划阶段工作产品

• 开发阶段

开发阶段结束的标记是“范围完成/第一次使用”里程碑。经过审批的项目计划是软件开发的基准，在本阶段要完成代码编写、基础架构的开发、培训资料的创建等任务。在这个阶段中，项目组可以设置若干个工作产品交付的子里程碑，如图 15-17 所示，每个子里程碑都要经过全面的验证检查。当所有产品都进入到第一个 Beta 版时就可以进行里程碑评审了，在评审时客户和项目组共同对产品的功能进行评估。

如图 15-18 所示，在计划阶段需要交付以下工作产品：

- ① 要交付所有可以用于 Beta 测试的代码。
- ② 由用户教育角色设计的所有演示方案、原型和培训文档。
- ③ 测试方案和计划描述了特定领域的测试需求以及测试的相关安排。
- ④ 更新后的风险管理相关文档。
- ⑤ 稳定阶段

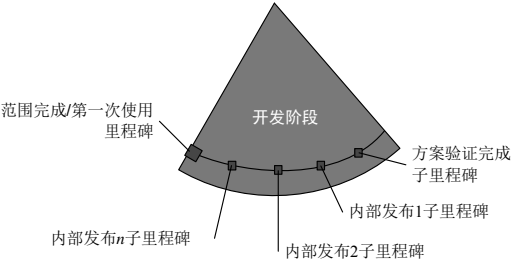


图 15-17 开发阶段子里程碑

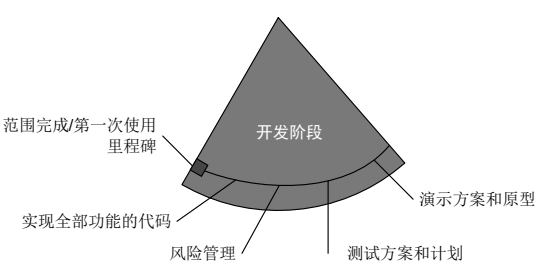


图 15-18 开发阶段工作产品

稳定阶段结束的标记是“产品发布”里程碑。稳定阶段的目标就是提高产品的质量，在此阶段中，测试、检查、排错是最主要的工作，当软件产品稳定时就可以准备发布了。在此阶段会有几个子里程碑，如图 15-19 所示。

如图 15-20 所示，在稳定阶段需要交付以下工作产品：

- ① 可执行的代码。
- ② 历史上不同版本的源代码。
- ③ 用于客户培训的手册和演示用的辅助工具。
- ④ 记录缺陷和问题的数据库。
- ⑤ 软件打包、安装和迁移的工具。

● 部署阶段

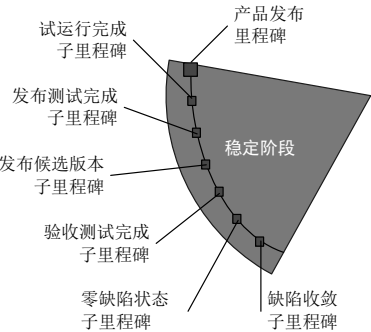


图 15-19 稳定阶段子里程碑

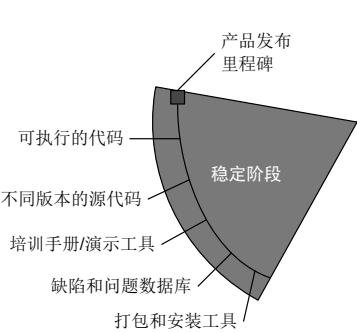


图 15-20 稳定阶段工作产品

部署阶段结束的标记是“部署完成”里程碑，其目的是将软件产品部署到客户的生产环境中。系统支持组的成员要确保该过程的顺利过渡。如图 15-21 所示，部署阶段还可以设置几个子里程碑。

如图 15-22 所示，在部署阶段需要交付以下工作产品：

- ① 项目收尾总结报告。

② 所有版本的文档、代码等工作产品。

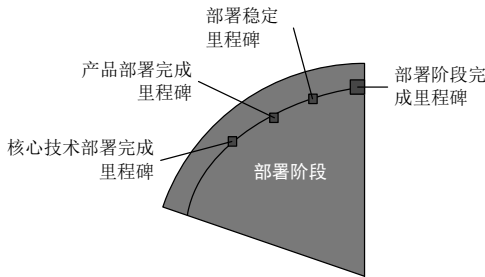


图 15-21 部署阶段里程碑

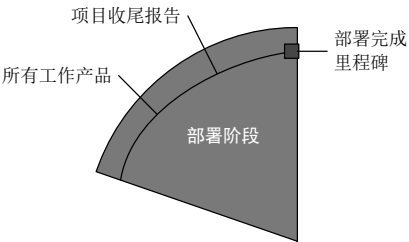


图 15-22 部署阶段工作产品

【小结】

如图 15-23 所示，MSF 有些类似于螺旋式开发模型或迭代式开发模型，都是将一个完整的项目分解为几个版本迭代式的开发，这样可以将项目风险减至最小，还可以增加很多用户体验的机会。

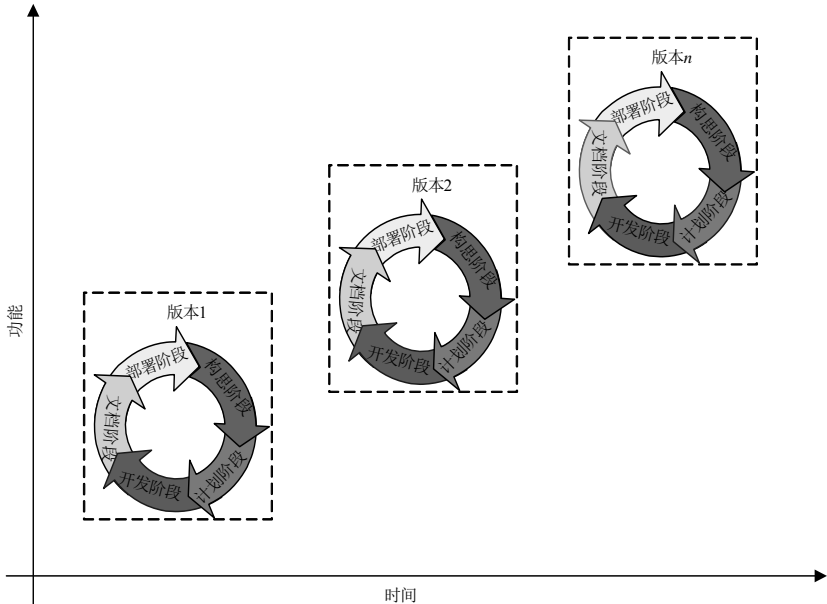


图 15-23 MSF 迭代的方式

直到部署阶段完成后，如果满足了客户的所有要求和商业目的，那么项目组通常会解散或加入其他项目组，如果没有达到客户的要求，那么就要继续从构思阶段开始进行下一轮生命周期的循环。

2. XP 极限生命周期模型

XP 极限式开发的英文全称是 Extreme Programming，在 1996 年由 KentBeck 提出的。XP 的定位是一个轻量级的生命周期模型，它的价值观是交流、朴素、反馈和勇气。XP 极限式开发有些类似于螺旋式开发或迭代式开发，它是将复杂的开发流程拆分为几个简单的、周期较短的过程，通过与客户和项目组的积极交流可以使所有项目关系人清楚项目待解决的问题、进度、目标等各个方面的情况。

XP 极限式开发强调的是“个体和交互胜于过程和工具”、“可以工作的软件胜于全面的文档”、“客户与项目组的合作胜于甲乙双方的谈判”、“拥抱变化胜于遵循计划”。XP 极限式开发的“极限”体现在以下几个方面：

- 极限的软件需求

与其他软件开发生命周期模型不同的是，在极限式开发中客户也被看做项目团队成员之一。项目团队成员与客户一起将复杂的需求划分成一个个简单的故事情节（User Story），并将其记录在卡片上（Story Card）。然后客户再根据其商业价值划分它的优先级，开发人员也会针对每个卡片来评估其技术的风险。最后这些卡片上的故事情节会在日后的迭代中被实现，客户也会依据其卡片上的内容进行验收测试。

- 极限式的工作环境

XP 项目的所有参与人员都在一个开放式的环境中工作。所有人都希望每周工作 40 小时，而不提倡加班；每天早上都要开一个例会来汇报各自的工作情况和发现的问题；房间里有一个大的白板上会贴着所有的 Story Card 和 CRC；最好下午再来点下午茶。

- 极限式的软件设计

XP 极限式开发提倡简单的设计（Simple Design），设计的内容恰好与当前系统的功能相匹配，而不像传统开发模型提倡的概要设计和详细设计，同样 XP 认为代码其实就是最好的文档。XP 还提倡通过大量的代码走查和重构来优化代码，尽量保持代码的整洁。

- 极限式的开发

XP 要求在统一的编码规范的前提下，两个开发人员一起结对编码，而且代码是属于整个团队的，任何人都都有权利修改发现的代码缺陷，但修改后要通过单元测试的检查。一旦有新的代码被添加，那么就需要进行产品集成。

- 极限式的测试

TDD 的开发模型提倡的是先写单元测试用例再编代码，对发现的缺陷都要增加相应的单元测试用例。验收测试的标准在定义故事卡片时就已经明确，XP 提倡使用自动化的测试脚本来进行验收。

XP 极限式开发生命周期模型如图 15-24 所示，它包含以下内容：

- User Stories：将项目团队成员与客户将软件的需求编写为一个个小的故事。

- **Release Planning:** 客户确定有哪些小的故事要被实现，项目团队会对其进行逐一估算，估算的内容包括成本、进度和风险。客户和项目组会根据估算的内容制订一个大致的项目计划。该计划不必非常准确，且会在实施过程中被不断地调整。
- **Iteration:** 实现 User Story 的过程，在这个过程中可能会发现新的需求，这些需求会被记录在小卡片上，在下次开发时会纳入计划。在 Release Planning 过程中会对开发的时间进行估算，在本过程中要记录项目实际的度量数据，例如一周可以完成几个小故事等。这些信息会作为下次计划的参考依据。
- **系统的隐喻:** 为了让每一个项目团队的成员都可以准确无误地理解项目的需求，XP 提倡使用一些比喻来描述系统或功能。
- **Acceptance Tests:** 在编写小故事的时候就会制订其验收标准，这样客户就可以很方便地对快速发布的每个版本进行功能检查。

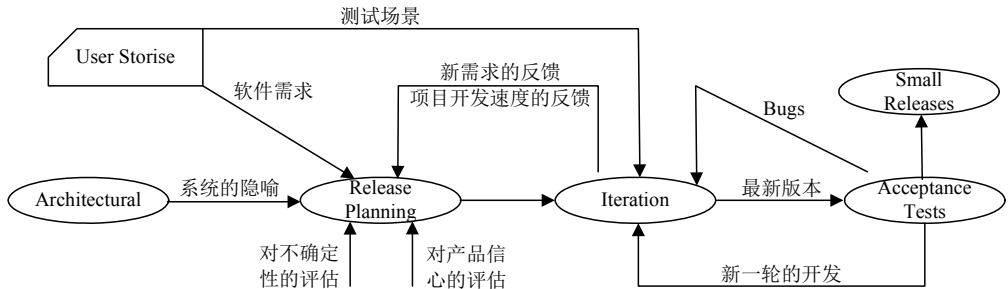


图 15-24 XP 极限式开发生命周期模型

总的来说 XP 极限式开发模型就是通过尽早并持续性地交付有价值的软件来给客户带来价值并满足客户的需要。即使到了项目的后期也要积极地拥抱变化，只有这样才能为客户带来竞争力。

15.2.3 定义裁剪指南

在软件工程中经常会出现“裁剪”两个字，大家千万不要把“裁剪”写成或理解成“裁减”。“裁剪”是项目集成管理工作中非常重要的工具，它是通过对组织财富库中的历史数据进行参考，并依据项目的实际情况对组织级的标准过程或软件工程的方法论进行“增加”或“裁减”。这就好比一个身材不好的人需要一套非常合体的西装，除了找裁缝量身定做以外很难在商场中买到十全十美的衣服。软件项目更是如此，没有两个软件项目是相同的，因此也就无法照搬任何一个项目的管理方式和流程。也没有一个项目是完全满足任何一种标准生命周期模型的前提条件的，如果生搬硬套那么必然会给项目带来风险和隐患。

例如一个项目使用瀑布式开发模型，但项目经理觉得 XP 极限式开发中提到的持续集成的理念非常好，而且对他的项目非常有帮助，那么就可以将其添加到项目的生命周期模型中并为其制订相关的计划；如果一个项目时间有限，而且开发人员对集成测试用例的编写又有技术困难，经过评估后如果觉得去掉该过程也不会对项目带来什么风险，那么就可以将集成测试过程裁减掉。

对于裁缝来说都有资料供他参考并指导其如何进行工作，那么软件管理人员在“裁剪”过程中也应该有一份指导性的文件来指引软件项目进行“裁剪”，告诉项目管理人员哪些是可以“裁”的，哪些是可以“加”的。裁剪指南应该包含的内容，如图 15-25 所示。

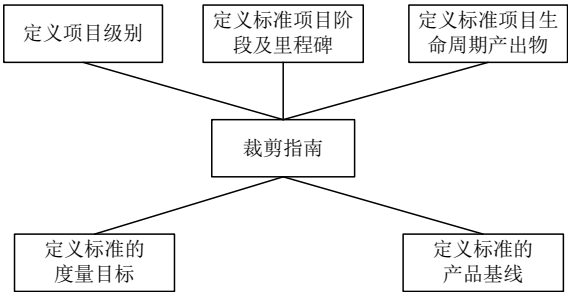


图 15-25 裁剪指南包含的内容

1. 定义项目级别

在裁剪指南中第一部分是对项目级别的分类。在日常工作中经常都会提到某某软件公司有个 100 万元的大型项目，可是另外一家公司却将 100 万元作为小型项目的衡量标准，这是根据软件公司自身规模和实力来定义的，没有一个硬性的定义标准，因此软件公司要对自身项目的情况进行总结并将它分级管理。

对项目分级管理的好处在于当项目需要参考历史数据时会比较有针对性，如果一个 100 万元的软件项目去参考以前一个项目金额为 10 万元的项目的历史数据，那么只会浪费时间和精力。

通常在对项目评级的时候项目才刚刚开始，此时根据项目合同可以知道总金额、项目周期和项目范围，虽然可以对项目的范围进行功能点估算以得到项目的规模，但此时项目的范围是非常不确切、不稳定的，因此在对项目分级时建议使用项目总成本和项目周期进行分类。如表 15-7 所示，通常会将项目分为 4~5 个级别，但一定要覆盖到公司内所有项目的类型。

表 15-7 项目级别分类

项目级别	项目级别描述
A 级项目	大于 150 万元的项目

续表

项目级别	项目级别描述
B 级项目	大于 100 万元且小于或等于 150 万元的项目
C 级项目	大于 50 万元且小于或等于 100 万元的项目
D 级项目	小于或等于 50 万元的项目

2. 定义标准项目阶段及里程碑

在裁剪指南中明确每种软件生命周期模型的各个阶段以及里程碑是非常必要的，当项目管理人员选择生命周期模型后就可以根据裁剪指南的定义对项目阶段里程碑进行裁剪。里程碑的定义必须和生命周期模型相匹配，因为里程碑标记着每个阶段的结束，不能只笼统地定义软件设计里程碑、产品验收里程碑等。对于有迭代特征的生命周期模型来说，预先对里程碑进行部分定义是非常重要的，这对公司内所有项目进行统一管理很有帮助。如表 15-8 所示，在制订项目阶段和里程碑裁剪标准时需要针对公司内不同类别的项目进行定义。

表 15-8 定义项目阶段和里程碑裁剪的标准

软件生命周期模型	阶 段	里 程 碑	A 类项目	B 类项目	C 类项目	D 类项目
瀑布式模型	项目立项阶段	需求调研确认里程碑	必选	推荐	—	—
	需求调研阶段		必选	必选	推荐	推荐
	项目计划阶段	项目计划确认里程碑	必选	必选	推荐	推荐
	软件设计阶段	软件设计确认里程碑	必选	必选	必选	推荐
	编码开发阶段	编码开发结束里程碑	必选	必选	必选	必选
	系统测试阶段	系统测试通过里程碑	必选	必选	必选	必选
	产品验收阶段	产品验收通过里程碑	必选	推荐	推荐	推荐
	项目结项阶段	项目结项里程碑	必选	必选	必选	必选
MSF 开发模型	构思阶段	核心团队建立子里程碑	必选	必选	推荐	---
		前景/范围确认里程碑	必选	必选	推荐	推荐
	计划阶段	技术方案确认子里程碑	必选	推荐	---	---
		功能定义确认子里程碑	必选	必选	推荐	---
		项目计划确认子里程碑	必选	必选	必选	---
		开发/测试环境确认子里程碑	必选	推荐	---	---
		计划确认里程碑	必选	必选	必选	必选
	开发阶段	方案确认子里程碑	必选	推荐	---	---
		内部发布子里程碑 1	必选	必选	---	---
		范围完成里程碑	必选	必选	必选	必选

续表

软件生命周期模型	阶 段	里 程 碑	A 类项目	B 类项目	C 类项目	D 类项目
MSF 开发模型	稳定阶段	零缺陷状态子里程碑	必选	推荐	---	---
		验收测试确认子里程碑	必选	必选	推荐	---
		试运行通过子里程碑	必选	推荐	推荐	---
		产品发布里程碑	必选	必选	必选	必选
	部署阶段	部署完成里程碑	必选	必选	推荐	---
XP 极限式开发模型	第 1 次迭代	发布计划确认子里程碑 1	必选	必选	必选	必选
		开发结束子里程碑 1	必选	必选	必选	必选
		验收通过里程碑 1	必选	必选	必选	必选
	第 2 次迭代	发布计划确认子里程碑 2	必选	必选	必选	推荐
		开发结束子里程碑 2	必选	必选	必选	推荐
		验收通过里程碑 2	必选	必选	必选	推荐
	第 3 次迭代	发布计划确认子里程碑 3	必选	必选	推荐	---
		开发结束子里程碑 3	必选	必选	推荐	---
		验收通过里程碑 3	必选	必选	推荐	---
	第 4 次迭代	发布计划确认子里程碑 4	必选	推荐	---	---
		开发结束子里程碑 4	必选	推荐	---	---
		验收通过子里程碑 4	必选	推荐	---	---
		项目结项里程碑	必选	推荐	---	---
.....

3. 定义标准的产品基线

项目基线的定义也需要与项目生命周期模型相匹配，在项目中基线的发布是有一定工作量的，因此在定义项目标准基线时也需要对此进行考虑。如表 15-9 所示，项目基线裁剪的标准也要根据项目的不同类别进行考虑。

表 15-9 定义项目基线裁剪的标准

软件生命周期模型	阶 段	基线名称	A 类项目	B 类项目	C 类项目	D 类项目
瀑布式模型	项目立项阶段	项目立项基线	必选	推荐	---	---
	需求调研阶段	项目需求基线	必选	必选	推荐	---
	项目计划阶段	项目计划基线	必选	必选	推荐	推荐
	软件设计阶段	软件设计基线	必选	必选	必选	推荐
	编码开发阶段	编码开发基线	必选	必选	必选	必选
	系统测试阶段	系统测试基线	必选	必选	必选	必选

续表

软件生命周期模型	阶 段	基线名称	A 类项目	B 类项目	C 类项目	D 类项目
瀑布式模型	产品验收阶段	产品验收基线	必选	推荐	推荐	推荐
	项目结项阶段	项目结项基线	必选	必选	必选	必选
MSF 开发模型	构思阶段	产品构思基线	必选	推荐	---	---
		功能定义基线	必选	推荐	---	---
	计划阶段	项目计划基线	必选	必选	必选	---
		方案基线	必选	推荐	---	---
		内部发布基线	必选	必选	---	---
	开发阶段	范围完成基线	必选	必选	必选	必选
		零缺陷状态基线	必选	推荐	---	---
		验收测试基线	必选	必选	推荐	---
		试运行基线	必选	推荐	---	---
		产品发布基线	必选	必选	必选	必选
XP 极限式开发模型	第 1 次迭代	计划确认基线 1	必选	必选	必选	必选
		编码基线 1	必选	必选	必选	必选
		产品验收基线 1	必选	必选	必选	必选
	第 2 次迭代	计划确认基线 2	必选	必选	必选	推荐
		编码基线 2	必选	必选	必选	推荐
		产品验收基线 2	必选	必选	必选	推荐
	第 3 次迭代	计划确认基线 3	必选	必选	推荐	---
		编码基线 3	必选	必选	推荐	---
		产品验收基线 3	必选	必选	推荐	---
	第 4 次迭代	计划确认基线 4	必选	推荐	---	---
		编码基线 4	必选	推荐	---	---
		产品验收基线 4	必选	推荐	---	---
		项目结项基线	必选	推荐	---	---
.....

4. 定义项目生命周期产出物

不同项目生命周期模型的各个阶段都会产出不同的工作产品，项目管理人员可以根据不同的项目级别来对工作产品的裁剪标准进行定义，这样就不会使得有些类型的项目过于臃肿，如表 15-10 所示。

表 15-10 定义项目各阶段工作产品裁剪的标准

软件生命周期模型	阶 段	工作产品	A 类项目	B 类项目	C 类项目	D 类项目
瀑布式模型	项目立项阶段	项目立项书	必选	推荐	---	---
	需求调研阶段	需求调研计划	必选	必选	推荐	推荐
		软件需求说明书	必选	必选	必选	推荐
		系统需求规格说明书	必选	必选	推荐	---
	项目计划阶段	项目过程定义书	必选	必选	推荐	推荐
		项目估算表	必选	必选	必选	推荐
		项目计划	必选	必选	必选	必选
		配置项列表	必选	必选	推荐	---
		项目进度计划	必选	必选	必选	必选
		配置管理计划	必选	推荐	推荐	---
		质量保证计划	必选	推荐	推荐	---
		风险管理计划	必选	必选	推荐	推荐
		度量计划	必选	必选	推荐	推荐
		配置库结构和权限表	必选	推荐	---	---
		系统测试计划	必选	必选	必选	必选
		功能点估算记录	必选	必选	推荐	---
		采购计划	必选	推荐	推荐	---
		WBS	必选	必选	推荐	推荐
		产品集成计划	必选	推荐	---	---
	软件设计阶段	概要设计说明书	必选	必选	必选	推荐
		详细设计说明书	必选	必选	---	---
		接口与组件对照表	必选	必选	必选	必选
		产品集成方案	必选	推荐	推荐	---
	编码开发阶段	产品代码	必选	必选	必选	必选
		代码交付确认单	必选	推荐	推荐	---
	系统测试阶段	系统测试用例	必选	必选	必选	必选
		系统测试总结报告	必选	必选	必选	必选
	产品验收阶段	用户验收测试用例	必选	必选	推荐	推荐
		验收测试报告	必选	必选	必选	必选
	项目结项阶段	项目总结及度量分析总结报告	必选	必选	必选	必选
		过程改进解决方案	必选	必选	必选	必选
		过程改进建议	必选	必选	必选	必选
.....

5. 定义标准的度量目标

度量工作是所有项目必不可少的内容，不管哪种生命周期模型都有对度量的相关要求，因此在制订裁剪指南时也要根据项目的规模来定义度量目标的裁剪标准，其大致内容如表 15-11 所示。

表 15-11 定义度量目标裁剪的标准

度量目标	A 类项目	B 类项目	C 类项目	D 类项目
项目实际进行了多少个月	必选	必选	必选	必选
项目的规模大小	必选	必选	推荐	---
项目的进度偏差	必选	必选	推荐	---
项目的成本偏差	必选	必选	推荐	---
项目计划估算准确性	必选	推荐	---	---
项目的生产率	必选	必选	推荐	推荐
项目发生了多少次变更	必选	必选	必选	必选
同行评审的投资回报率	必选	推荐	---	---
各种文档的页数	必选	必选	必选	---
项目各阶段工作量的分布情况	必选	必选	必选	必选
项目培训成本	必选	必选	推荐	---
配置管理员的成本	必选	必选	推荐	---
QA 质量保证人员的成本	必选	必选	推荐	---
度量工程师的成本	必选	必选	推荐	---
产品平均缺陷率	必选	必选	必选	推荐
每个模块的缺陷率	必选	必选	必选	必选
每个项目团队成员的生产率	必选	必选	推荐	---
每个项目团队成员的缺陷率	必选	必选	推荐	---
完成某个任务所花费的成本	必选	推荐	---	---
缺陷修正率	必选	必选	必选	必选
缺陷未修复率	必选	必选	推荐	---
缺陷未复查率	必选	推荐	---	---

至此裁剪指南的基本内容就定义完毕了，其流程如图 15-26 所示，对定义后的裁剪指南还要经过组织的讨论才能颁布使用。裁剪指南是项目集成管理和组织标准过程的核心内容，是指导项目管理人员将组织级标准过程转化为项目过程的依据。

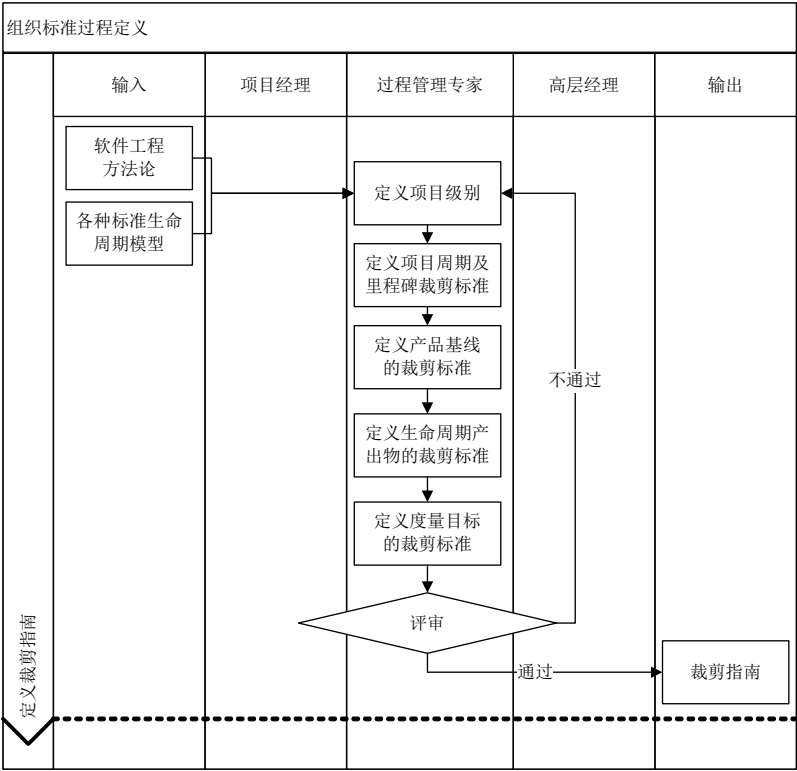


图 15-26 定义裁剪指南

15.2.4 建立组织级度量库和工作环境

组织度量库也是组织标准过程中的一个部分，它可以是数据库，也可以是诸如 Excel 之类的电子表格，它记录了与组织标准过程相关的产品度量、过程度量和培训度量等，常见的有：

- 软件项目规模的估算（例如：功能点、代码行）
- 项目资源和成本的估算
- 项目实际规模、人力和成本
- 发现的缺陷数量、缺陷的严重程度
- 同行评审的覆盖率
- 测试的覆盖率

组织度量库中的内容是在项目结束后由项目经理进行收集和总结，然后提交组织级度

量管理员进行存档并更新组织度量库中的数据，最后发布新的版本基线。项目组在对项目进行估算时引用组织度量库中的数据要指明所参考的基线版本号，因为组织级度量库中的数据会随着每次项目的提交而变化，其操作的流程如图 15-27 所示。

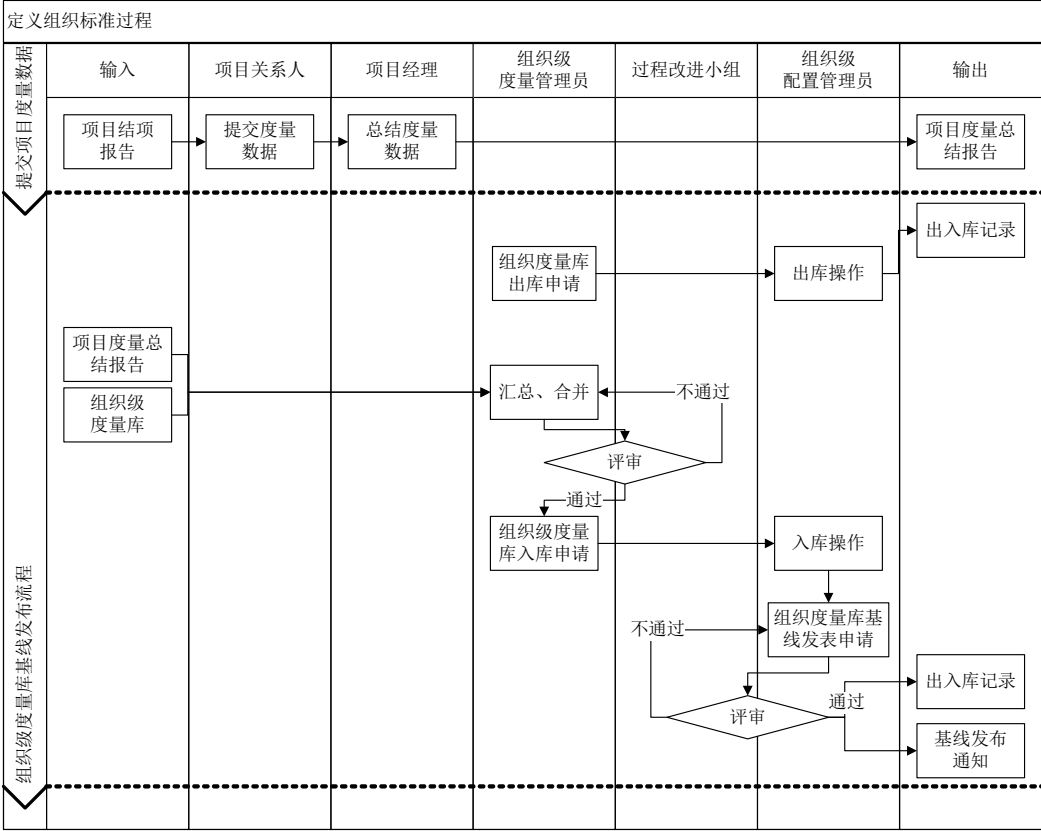


图 15-27 组织级度量库操作流程

组织级工作环境也是组织标准过程中的一个组成部分，它记录了软件开发过程中所需要使用到的软、硬件工具，以及公司内涉及软件研发人员的各种规章制度。它为“项目集成管理”过程中制订项目组的工作环境提供了参考的依据。常见的组织工作环境包括以下内容：

- 软件工程师个人机器最低配置，以及操作系统、开发工具等
- 软件测试师个人机器最低配置，以及操作系统、测试工具等
- 产品集成环境所使用机器的最低配置，以及操作系统、集成测试环境等

- 数据库服务器的最低配置、操作系统、数据库、补丁等
- 公司图书借阅制订
- 公司考勤制订
- 编码规范

15.3 组织培训的最佳实践

培训的目的是提高公司员工的技能和知识，使他们能有效地完成各自的工作。组织培训包括两方面的工作：一个是配合组织的商业目标进行培训，一个是为了支持项目的需要而进行的培训。如果项目的培训具有推广意义，也可以将其纳入组织培训的范畴，否则项目

培训可以单独进行。如图 15-28 所示，组织培训分为两个部分，建立组织培训的能力主要是负责完成收集培训的需求、对培训需求进行整理和分析、制订年度培训计划和开发培训材料。实施培训是按照年度培训计划开展相关工作，并记录、反馈和度量培训的效果。

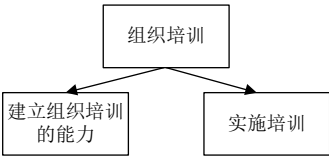


图 15-28 组织培训的组成部分

15.3.1 建立组织培训的能力

建立组织培训能力首先要收集组织内所有成员的培训需求，识别并确定哪些培训是组织所需要负责的，然后根据这些培训需求制订年度培训计划，最后根据培训计划寻找公司内部或外部的资源来承担培训的课程，其步骤如图 15-29 所示。

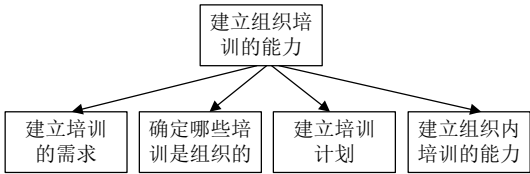


图 15-29 建立组织培训能力的步骤

1. 建立培训的需求

组织培训需求通常来源于以下方面，培训专员可以对此进行逐一分析。

- 组织标准过程和过程改进计划：组织标准过程制订后需要通过培训的方式进行推广。
- 组织商业目标和构想：为了支持公司未来的发展而需要开展的培训。
- 技术能力的评估：项目的技术需要与组织成员技能之间的差别。
- 项目风险的分析：通过培训的手段来应对项目的风险。

除了对以上内容进行分析外，培训专员还可以对组织全体成员进行问卷调查，问卷调查的内容如表 15-12 所示。

表 15-12 组织培训问卷调查表

被调查信息			
被调查人:		员工编号:	
调查日期:		职务:	
培训需求信息			
期望培训的类型	<input type="checkbox"/> 管理类 <input type="checkbox"/> 开发类 <input type="checkbox"/> 市场类 <input type="checkbox"/> 营销类 <input type="checkbox"/> 其他		
期望参加的培训课程	希望培训的内容	培训方式	
	1	<input type="checkbox"/> 外训 <input type="checkbox"/> 内训	
	2	<input type="checkbox"/> 外训 <input type="checkbox"/> 内训	
	3	<input type="checkbox"/> 外训 <input type="checkbox"/> 内训	
	4	<input type="checkbox"/> 外训 <input type="checkbox"/> 内训	
	5	<input type="checkbox"/> 外训 <input type="checkbox"/> 内训	
		<input type="checkbox"/> 外训 <input type="checkbox"/> 内训	
需要培训的原因			
培训的费用	<input type="checkbox"/> 公司承担 <input type="checkbox"/> 个人承担（个人承担的最大比例__%）		
对公司培训的建议			

- 用决策分析的方法来确定哪些培训是组织级的。

组织培训的需求会跨越组织内的多个部门，这些需求有些是共同的，有些是某个部门所特需的。组织级的培训专员通常只需要筛选共同的需求，对于某些特殊的培训需求可以与其部门进行协商，以确定它是否具有推广意义。然后再通过决策分析的方法与公司内的相关负责人一同对培训的需求进行决策，决策分析的具体流程详见本书第 13 章。

2. 建立培训计划

一旦培训需求被定义，即可提出为满足这些需求而制订的培训计划。组织一般会在年底制订下一年度的培训计划，在培训计划中应该尽可能覆盖到组织内的所有员工，例如公司领导需要接受“领导力和战略”方面的培训、销售人员应该接受人际关系技能方面的培训等。年度培训计划的流程如图 15-30 所示，它中通常会包含以下内容：

- 识别组织培训的需求
- 取得培训的资源

- 建立和维护培训的资料和课件
- 建立和维护培训的记录
- 评估培训的效果

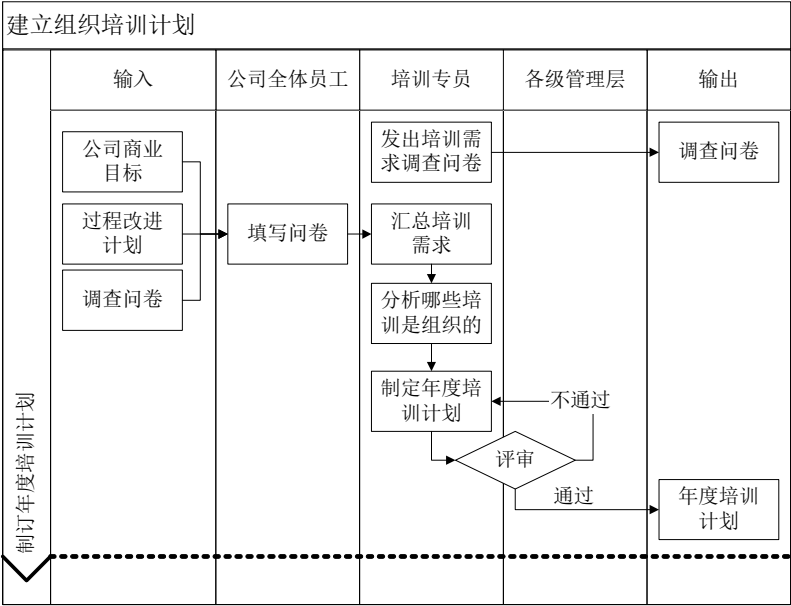


图 15-30 制订年度培训计划

3. 建立组织内培训的能力

在建立完培训计划后，培训专员开始整理并定义培训的课程。培训课程应该由该领域的专家进行评审，其流程如图 15-31 所示。外训的课程必须要经过相关领导的审批，经过批准的培训课程可以记录在课程定义表中，其样式如表 15-13 所示。依据培训需求建立的每项培训课程都需要提供以下信息：

- 预期的培训对象
- 参加培训的必备条件与准备事项
- 课程的目标
- 课程的具体时间安排
- 完成课程评估的准则
- 免训的准则

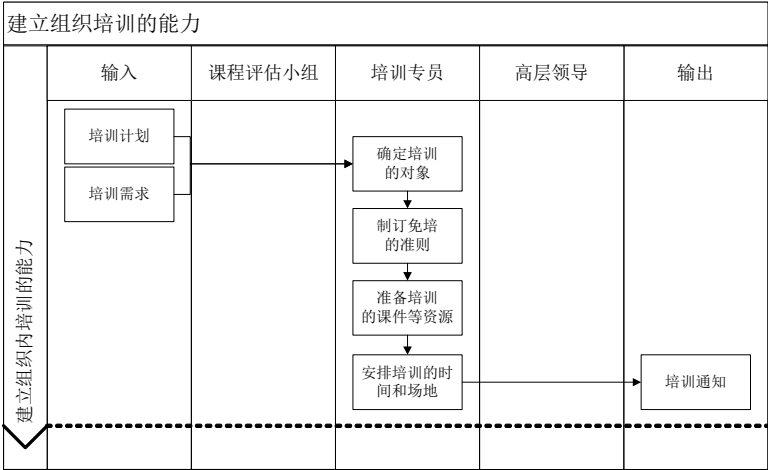


图 15-31 建立组织内培训的能力

表 15-13 课程定义表

课程名称	课程类别	培训方式	培训对象	课程目标	教材标准	考核方式	讲师技能要求	免培准则	证书类型

组织培训的能力可以分为内部和外部的能力。建立外部培训能力的工作是拓展与外部培训机构的合作关系，根据年度培训计划寻找符合培训需求的讲师。参与外部培训的形式有：

- 商业性训练课程
- 专业交流会
- 专题讨论会

建立组织内部培训能力的工作分为以下几个部分：

① 依据培训需求确定培训的方法：

- 计算机辅助教学
- 自学
- 传统的师生教学模式
- 视频教学
- 午餐讨论会

- 系统地在职训练

② 筛选公司内具有各种技术专长或资质的人员，共同组建内部讲师团队来承担部分培训的需求。为确保从组织内部寻找的讲师具备所必需的知识与教学技巧，请按照本书第 13 章的内容制订讲师评选标准，并对讲师的人选进行决策。

③ 编写培训的课件、考题及其他相关资料。

15.3.2 实施培训

培训的目的是传授知识和技能给组织内执行各种任务的人员。培训实施的流程如图 15-32 所示。

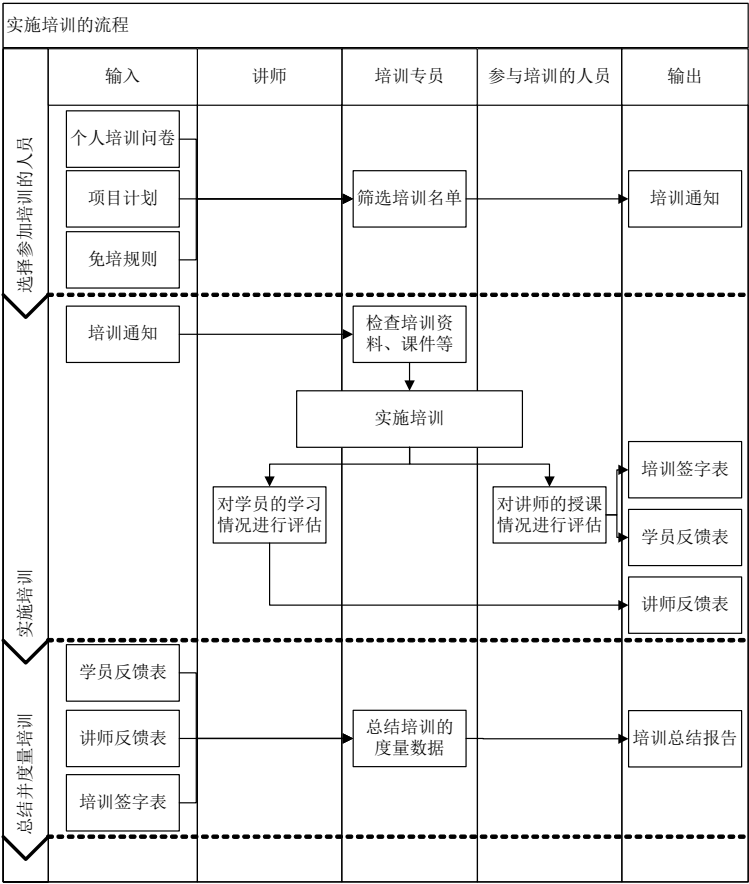


图 15-32 培训实施的流程

如果某些人已经具备了必要的知识与技能，并可以圆满地完成所分派的任务，那么就可以根据免培规则进行筛选。另外也可以配合培训需求问卷以及正在实施的项目计划选择受训人员，确保组织培训是有用的并且不会影响项目的正常工作。

培训专员需要提前发出培训通知，安排好培训的场地和设备，检查培训的课件和资料是否齐全，在培训时协助讲师做好培训。

在培训结束后发放问卷给讲师和学生，以对本次培训的效果进行调查。最后将每次培训的情况汇总到员工培训一览表中进行备案，员工培训一览表的格式如表 15-14 所示。

表 15-14 员工培训一览表

员工名称	培训课程 1			培训课程 2			培训课程 3		
	是否需要参加培训	是否已经参加培训	培训的效果	是否需要参加培训	是否已经参加培训	培训的效果	是否需要参加培训	是否已经参加培训	培训的效果

待培训反馈表收集上来后，如图 15-33 所示，培训专员还需要对培训的效果、时间、成本等进行度量，并且提出培训的过程改进意见。

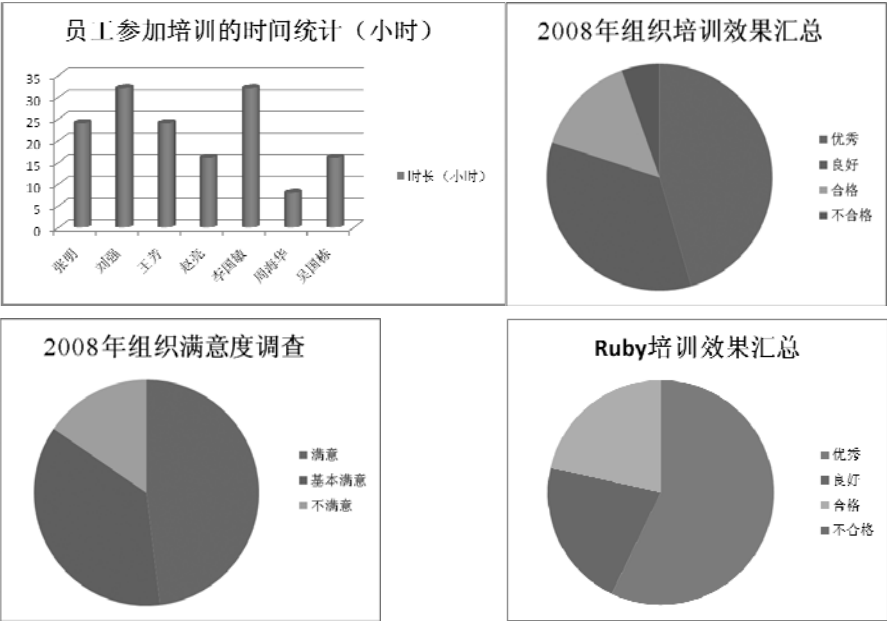


图 15-33 组织培训度量内容

15.4 案例分析在实际工作中如何把握质量改进的时机

软件质量的持续改进会涉及公司的各个方面，如何可以更好地开展质量持续改进的工作，质量管理人员可以通过以下案例得到启示。

【案例】

某软件公司 EPG（过程改进小组）负责人小赵最近遇到了一些问题，他觉得过程改进的工作寸步难行，要不是项目组工作特别忙，要不就是项目组认为 EPG 在有意挑他们的毛病。那么过程改进小组应该如何开展工作呢？

【分析】

过程改进小组在日常工作中往往被项目组当作“麻烦的制造者”，过程改进主要是寻找管理上的问题，因此项目经理的抵触情绪最大。

过程改进小组首先从工作的出发点上进行转变，过程改进所做的工作是帮助项目经理更好地对项目进行管理，要从“麻烦的制造者”转变为“项目经理的好帮手”。

过程改进小组要多吸收一些项目管理人员、软件设计人员、软件开发和测试等人员作为兼职的改进组成员，这样在过程改进的时候就可以争取到更多的支持者，而且他们在日常工作中也会起到表率作用。

多采纳项目组提出的改进意见，从他们最需要的地方入手，真正成为“项目经理的好帮手”。

把握过程改进的时机，只有发生问题并且对项目影响严重时才是过程改进阻力最小的时候。

15.5 小结

软件质量的持续改进需要依靠发现过程改进的机会、定义组织标准的流程以及组织培训这三驾马车来推进，它们之间的关系是紧密相关、缺一不可的。如果说本书前面章节的内容分别是“木桶”上的每块“木板”，那么本章的内容就是“木匠”所需要具备的技能。掌握了这个技能就可以找到“木桶”上有隐患的地方并进行修补，又可以不断地加大“木桶”的容积，使企业的软件质量得以持续提高。

15.6 思考题

1. MSF 生命周期模型分为几个阶段？
2. 裁剪指南包含哪些内容？
3. 软件质量持续改进的三驾马车是什么？

16

第 16 章

TFS在软件研发中的应用

TFS（Team Foundation Server）是微软推出的一套全方位的软件研发、项目管理、质量管理的平台，如图 16-1 所示，它是整合了软件开发工具、软件测试工具、软件配置管理工具、软件集成和发布工具、软件项目管理工具、Office 文档工具、软件项目管理的流程、软件项目管理的各种文档模板、软件项目管理的各种报表，以及软件项目的门户网站。

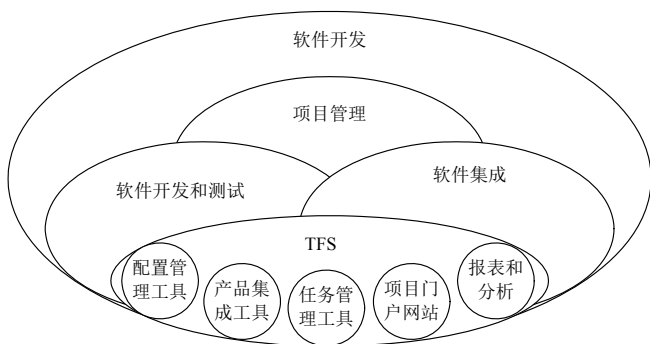


图 16-1 TFS 功能示意图

如图 16-2 所示，TFS 的逻辑体系采用了 3 层架构，分别为客户层、应用层和数据层。使用 TFS 的用户可以通过 Visual Studio 的开发环境、Office 中的 Excel 和 Project 工具，以及 IE 浏览器来使用 TFS 所提供的服务。这些服务都是由应用层的组件所提供的，软件项

目的所有数据都存储在 SQL Server 数据库中。

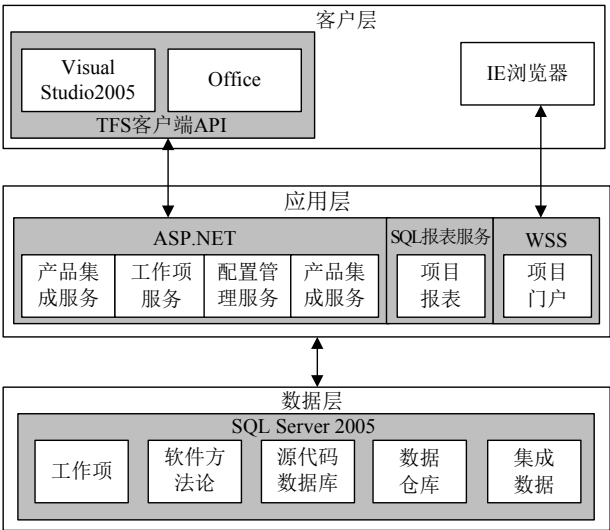


图 16-2 TFS 体系结构

16.1 TFS 的拓扑结构

在安装和部署 TFS 时有单服务器和多服务器两种拓扑结构可供选择。单拓扑结构如图 16-3 所示，TFS 所提供的服务与 SQL Server 安装在同一台服务器上。多服务器拓扑结构与单服务器拓扑结构的区别是将 TFS 应用服务器中的不同应用进行单独部署，例如：项目门户网站可以单独部署到另外一台服务器上。

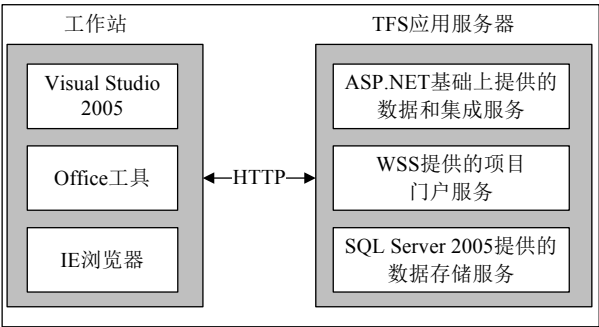


图 16-3 TFS 单服务器拓扑结构

但无论是哪种拓扑结构，都必须遵循以下原则：

- 必须在同一个域里安装应用层和数据层。
- TFS 的服务器必须安装 Windows Server 2003 SP1 或最新的版本。
- 必须将 TFS 应用层 Web 服务安装到同一个服务器上。
- 必须将 TFS 的实例安装到一台单独的物理机器上。
- 不能跨越多个数据库服务器来部署 TFS 的数据库，所有项目的数据库必须在同一个群组中。
- 不能在域服务器上安装 TFS。
- 若采用多服务器拓扑结构，必须使用加入到 Active Directory 域中的计算机。

单服务器和多服务器体系结构的设计主要是为了满足不同规模软件公司的使用，这两种结构所需要的硬件配置有着较大区别，分别如表 16-1 和表 16-2 所示。软件公司可以根据列表中的定义在安装前确定拓扑结构的类型。

表 16-1 单服务器拓扑结构适用的范围及硬件要求

组 件	小型团队	中等团队
用户数量	小于或等于 20 人	小于或等于 50 人
CPU	2.2GHz	
操作系统	带 SP1 的 Windows Server 2003 企业版或标准版	
内存	1GB	2GB
硬盘	8GB 可用空间	30GB 可用空间

表 16-2 双服务器拓扑结构适用的范围及硬件要求

组 件	部 门 级		公 司 级	
用户数量	小于或等于 250 人		小于或等于 500 人	
操作系统	带 SP1 的 Windows Server 2003 企业版或标准版			
逻辑层次	应用层	数据层	应用层	数据层
CPU	2.2GHz	双处理器，2.2GHz	双处理器，2.2GHz	四处理器，2.2GHz
内存	1GB	2GB	2GB	4GB
硬盘	20GB 可用空间	80GB 可用空间	40GB 可用空间	150GB 可用空间

在部署 TFS 服务器时，要确保服务器已经安装如表 16-3 所示的软件，否则 TFS 将无法安装。

表 16-3 TFS 服务器端需要安装的软件

服务器端需要安装的软件	用 途
SQL Server 2005 及补丁	用作 TFS 和 Windows SharePoint Service 的数据库
.NET FrameWork2.0 的补丁 KB913363	用于更有效地支持大文件处理的 ASP.NET 更新
带 SP2 的 Windows SharePoint 2.0	项目的门户网站

由于 TFS 提供了多种应用服务，因此用户在使用和安装这些服务时还要具备相应的权限。在安装 TFS 的过程中需要创建如表 16-4 所示的用户。

表 16-4 TFS 安装过程中需要创建的用户

示例用户名	用 途
TFSSETUP	<ul style="list-style-type: none">➤ 用于运行 TFS 的安装程序➤ 该账户必须是 Team Foundation Server 计算机上的管理员➤ 该账户和接下来的两个服务账户必须是同一个域的成员
TFSSERVICE	<ul style="list-style-type: none">➤ 用作 TFS 服务和 SharePoint Timer Service 服务的账户➤ 用作 TFS 应用程序池和 Windows SharePoint Services 应用程序池的标识➤ 必须对 TFS 计算机具有“本地登录”权限➤ 要获得最佳安全性，此服务账户不是 TFS 计算机上的管理员➤ 应为域上的 Active Directory 选择“账户敏感，无法委托”选项
TFSREPORTS	<ul style="list-style-type: none">➤ 用作 SQLServer Reporting Services 数据源的服务账户➤ 该账户不是 TFS 计算机上的管理员➤ 该账户必须对 TFS 计算机具有“本地登录”权限

下面我们使用 TFS 创建一个项目。

TFS 安装成功后就可以开始创建一个团队项目，首先打开 Visual Studio 2005，然后单击菜单【工具】并选择【连接到 Team Foundation Server】子菜单，系统弹出窗口如图 16-4 所示。

单击【服务器】按钮，如图 16-5 所示，在弹出的“添加/移除 Team Foundation Server”窗口中单击【添加】按钮，在弹出的对话框里输入 TFS 服务器的地址、端口号和协议类型。

TFS 服务器连接成功后就可以开始创建新的团队项目，用户可以在“团队资源管理器”中看到 TFS 服务器的名称，如图 16-6 所示，右键单击 TFS 服务器，在弹出的右键菜单中选择【新建团队项目】子菜单。

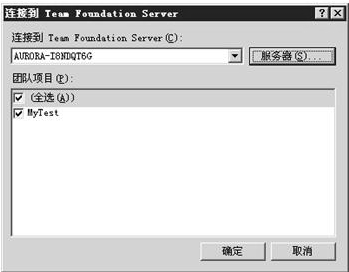


图 16-4 连接 TFS 服务器



图 16-5 “添加/移除 Team Foundation Server” 窗口



图 16-6 准备创建 TFS 团队项目

STEP 01 如图 16-7 所示，用户在弹出菜单中输入团队项目的名称。

STEP 02 如图 16-8 所示，选择团队项目所需要使用的开发过程模板。TFS 提供了两套开发过程模板供用户选择：一套是“极限开发过程模型”，适用于小团队快速开发的软件项目；一套是“CMMI 开发模型”，适用于大型的、复杂的软件项目。用户可以根据实际情况进行选择。



图 16-7 创建团队项目的名称

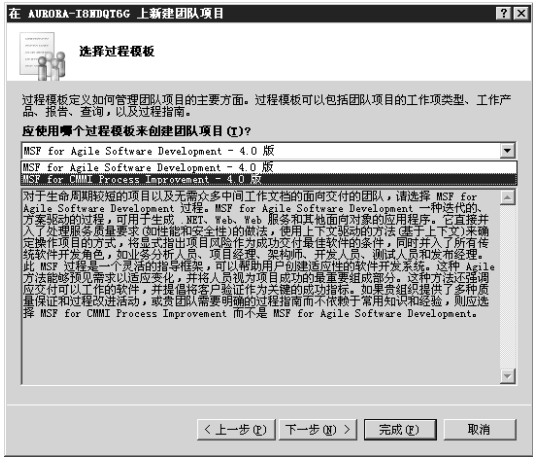


图 16-8 选择团队项目的过程模板

STEP 03 TFS 会创建项目的门户，当团队项目创建成功后，身处各地的项目关系人都可以通过项目门户网站了解项目的进展状况。如图 16-9 所示，在安装向导窗口中用户可以输入项目门户网站的名称和描述。

STEP 04 TFS 集成了版本管理工具，用户可以像使用 VSS 一样在 TFS 中对项目文档、代码进行版本管理。如图 16-10 所示，在创建 TFS 的过程中用户可以选择创建一个空的源代码文件夹还是选择已经存在的源代码目录，或者暂时不创建用于版本管理的源代码文件夹。单击【完成】按钮 TFS 开始创建团队项目。



图 16-9 创建团队项目的门户网站



图 16-10 创建团队项目的源代码版本管理文件夹

16.2 TFS 团队项目功能简介

团队项目的目录如图 16-11 所示，分为“工作项”、“文档”、“报告”、“团队项目生成”和“源代码管理”5 个部分。

打开“工作项”目录，如图 16-12 所示，在“团队查询”子目录下 TFS 提供了许多默认的查询视图。这些视图可以帮助项目团队成员了解项目的方方面面。



图 16-11 团队项目的目录



图 16-12 工作项的查询视图

当用户单击某个查询视图时，例如“所有任务”。如图 16-13 所示，在 Visual Studio 2005 中会显示该视图的查询结果。双击查询结果列表中的一条记录，Visual Studio 2005 会显示该任务的详细信息，例如任务的标题、状态、类型、历史记录等具体内容。



图 16-13 TFS 提供的查询视图

打开 TFS 团队项目下的“文档”目录，如图 16-14 所示，在该目录下会依据用户选择的“开发过程模型”生成相应的过程文档，其中包括项目管理类、开发类和质量类的文档模板。这些文档模板不但给项目提供了标准化的定义，而且在一定程度上降低了项目组编写文档的工作量。

TFS 还提供了项目的报告，以便各级管理人员可以随时了解项目的进展情况，并且这些报告为项目组提供了准确的度量数据。如图 16-15 所示，例如“Bug 比率”、“Bug（按优先级别）”、“剩余工作”、“质量指示器”等。

16.2.1 创建工作项

工作项是 TFS 团队项目管理中的重要组成部分，它是项目团队各种工作类型的总和。在 TFS 中，工作项类型根据团队项目选择的“开发过程模型”的不同而有所区别。在“敏捷式开发过程模型”中工作项的分类如表 16-5 所示。

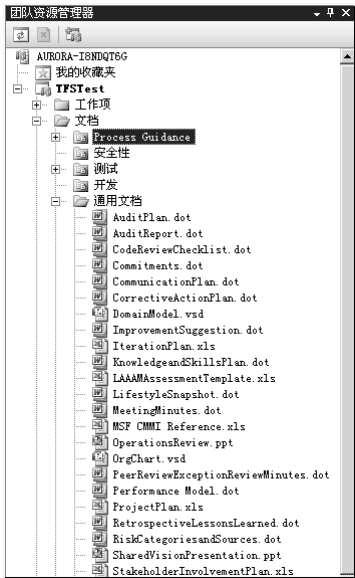


图 16-14 TFS 提供的项目文档模板



图 16-15 TFS 提供的报告

表 16-5 敏捷式开发过程模型中的工作项分类

工作项分类	描 述
Bug	表示应用程序中的缺陷或潜在缺陷
风险	表示可能会对项目产生负面或正面影响的事件或条件
方案	表示用户交互通过系统的单一路径
任务	表示分配给某个项目团队成员的工作任务
服务质量需求	表示约束系统应如何工作的需求

在 TFS 的敏捷式开发过程模型中工作项的状态有以下几种，它们之间的关系如图 16-16 所示。

- 活动。例如：某个团队成员正在从事的任务。
- 已解决。例如：某个缺陷已由开发人员解决。
- 已关闭。例如：某个任务由测试人员测试合格并关闭。

TFS 提供的“CMMI 开发过程模型”中对工作项定义的种类会多于“敏捷式开

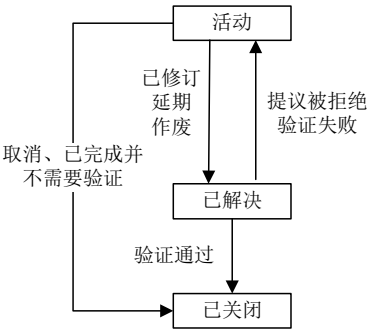


图 16-16 敏捷式开发模型中工作项的状态流程图

发过程模型”，如表 16-6 所示。

表 16-6 CMMI 开发模板中的工作项分类

工作项分类	描 述
Bug	表示应用程序中的缺陷或潜在缺陷
风险	表示可能会对项目产生负面或正面影响的事件或条件
需求	表示客户需求的描述
任务	表示分配给某个项目团队成员的工作任务
评审	表示代码、设计或部署评审的结果
变更请求	表示对应用程序的修改建议
问题	表示可能阻塞工作或当前正在阻塞工作的事件

CMMI 过程模型中工作项的状态有以下几种，它们之间的关系如图 16-17 所示。

- 建议。例如：开发或测试人员提出的建议。
- 活动。例如：某个团队成员正在从事的任务。
- 已解决。例如：某个缺陷已由开发人员解决。
- 已关闭。例如：某个任务由测试人员测试合格并关闭。

在 TFS 中除了通过 Visual Studio 2005【团队】菜单下的子菜单【添加工作项】外，还可以通过以下两种方式来创建工作项。如图 16-18 所示，一种是通过微软的 Excel 来添加工作项，一种是通过微软的 Project 来添加工作项。

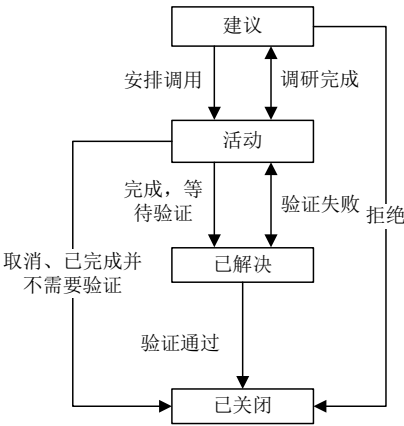


图 16-17 CMMI 模型中工作项的状态流程图

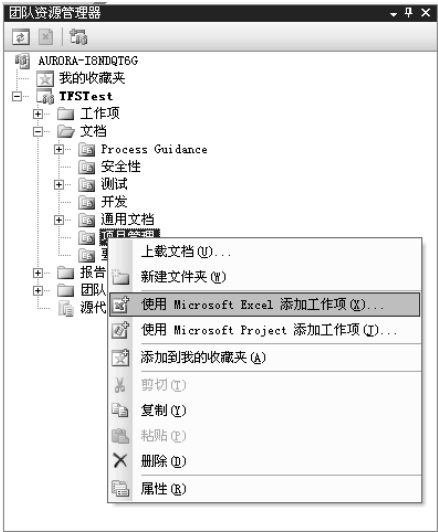


图 16-18 创建 TFS 团队项目的工作项

如果用户选择通过 Excel 方式来创建工作项,那么就可以选择右键菜单上的【使用 Microsoft Excel 添加工作项】子菜单,如图 16-19 所示,TFS 会打开 Excel 并将团队项目的相关信息显示在 Excel 中,用户只需要按照格式填写相关内容即可,其中工作项的 ID 是只读的,在发布工作项后 TFS 会自动为其分配相应的编号。

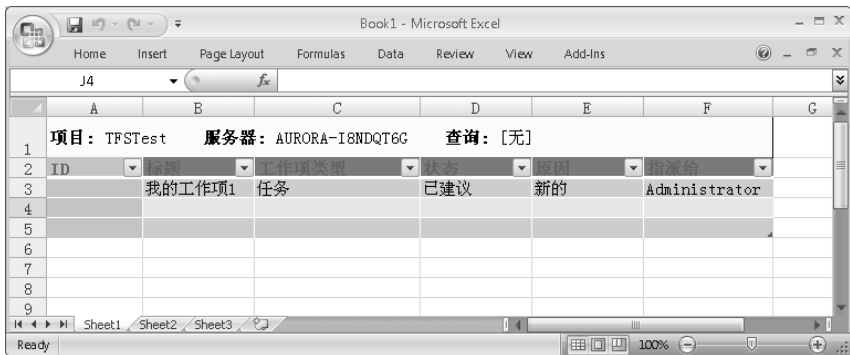


图 16-19 在 Excel 中添加工作项

当工作项的信息在 Excel 中填写完毕后,如图 16-20 所示,就可以单击 Excel 工具栏中的【发布】按钮将刚才创建或修改的工作项更新到 TFS 服务器。当工作项发布到 TFS 服务器后,用户就可以通过 TFS 团队项目的相关查询视图来进行查看。

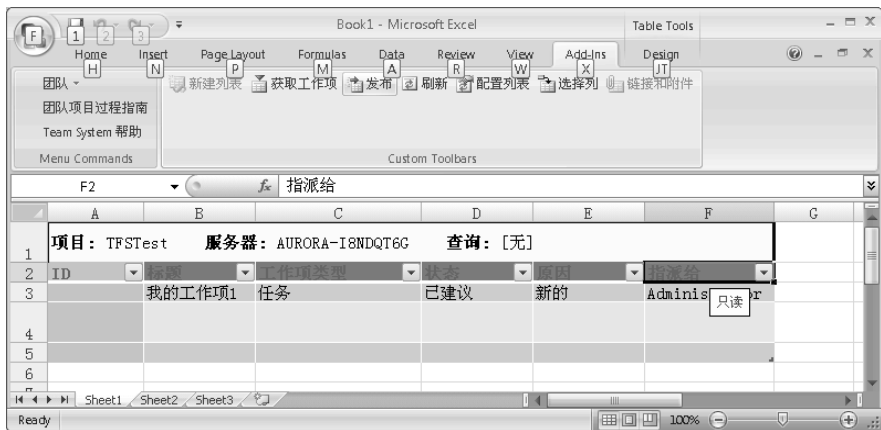


图 16-20 将 Excel 中的工作项发布到 TFS 服务器

用户也可以通过 Microsoft Project 来创建工作项。选择右键菜单上的【使用 Microsoft Project 添加工作项】子菜单,系统会打开 Microsoft Project 软件,用户可以在这里安排软件项目的相应任务。如图 16-21 所示,单击【Publish】按钮,系统会将填写的工作项信息发布到 TFS 服务器。

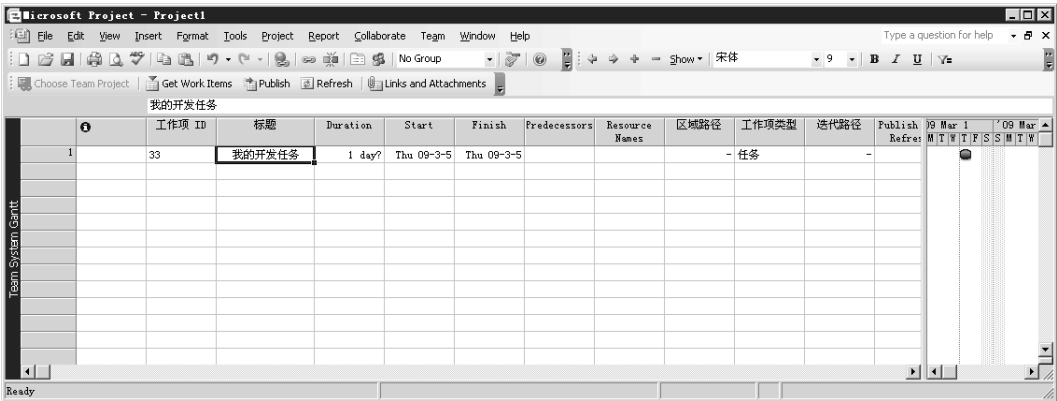


图 16-21 在 Microsoft Project 中创建工作项

16.2.2 添加查询视图

在 TFS 团队项目中默认提供了很多用于查询项目各方面情况的视图，如果用户需要自定义其他查询视图，如图 16-22 所示，右键单击【我的查询】文件夹，在弹出菜单中选择【添加查询】子菜单。

在 Visual Studio 2005 窗口中，用户可以自定义查询的条件。如图 16-23 所示，在自定义查询条件时 TFS 还提供了一些通配符供用户使用，例如：“@Me”就代表当前登录 TFS 的用户。





图 16-22 添加查询步骤 1



图 16-23 添加查询步骤 2

自定义视图的查询条件设置完毕后，就可以将其进行保存。如图 16-24 所示，在弹出窗口中输入视图的名称，并选择视图存放的位置，如果用户选择“团队查询”选项，则视图保存在 TFS 团队项目的“团队查询”目录下，项目团队的所有成员都可以使用。如果选择“我的查询”选项，则视图保存在“我的查询”目录下，那么只有“创建人”才能使用。

查询视图创建完毕后，就可以单击工具栏中的  按钮来执行查询。在默认情况下查询结果显示工作项的“编号”和“标题”，用户可以通过工具栏中的  “列选项”按钮来设置查询结果显示的信息。如图 16-25 所示，在弹出窗口中定义查询列表里每个列的内容和顺序。

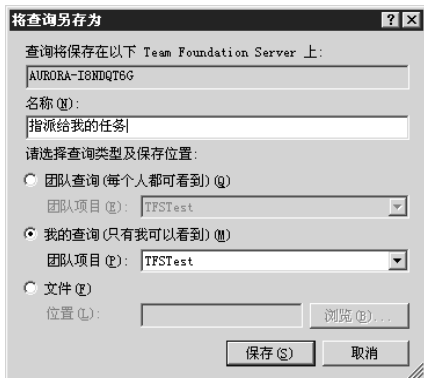




图 16-24 保存查询视图



图 16-25 设置查询结果列表中显示的信息

用户可以将查询的结果导出到 Excel 或 Project 中进行查看。首先选择需要导出的信息，然后在工具栏中单击  或  按钮，查询的结果会显示在 Excel 或 Project 中，分别如图 16-26 及图 16-27 所示。用户可以在 Excel 或 Project 中对每条查询的结果进行编辑，最后再发布回 TFS 服务器。

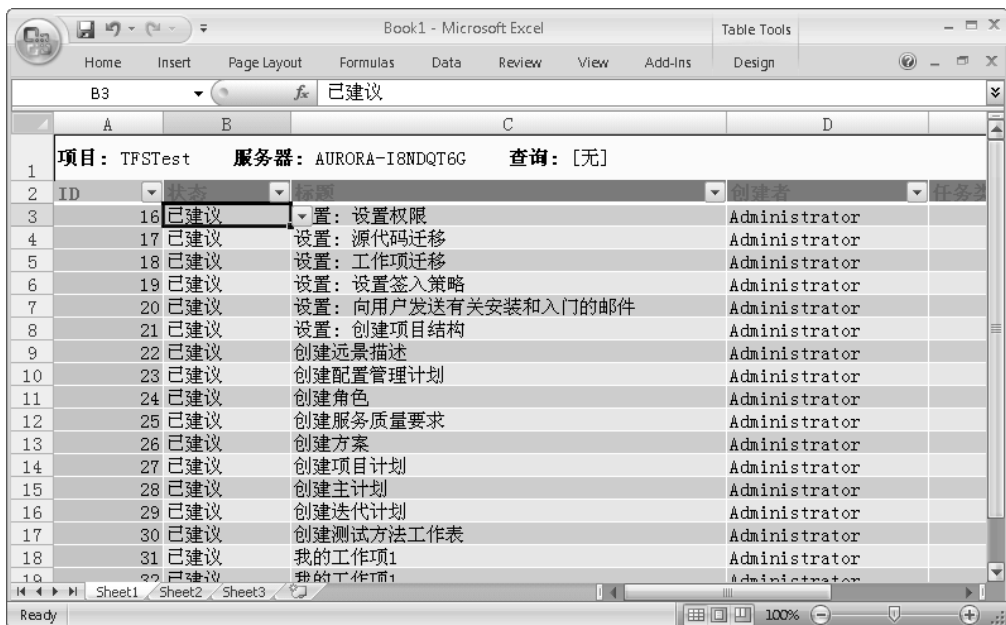


图 16-26 将查询结果导出到 Excel 中

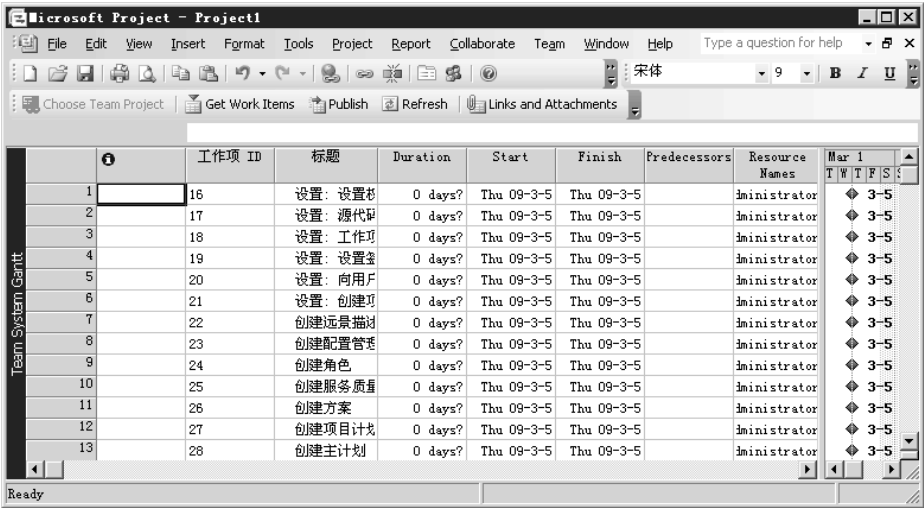


图 16-27 将查询结果导出到 Project 中

16.2.3 源代码管理

TFS 提供了对源代码进行管理的功能,所有对源代码的操作都与 VSS 的使用方法基本相同。用户在“团队资源管理器”中打开自己的团队项目,然后双击“源代码管理”,如图 16-28 所示, Visual Studio 2005 会在左边显示 TFS 服务器下所有团队项目的源代码管理目录。

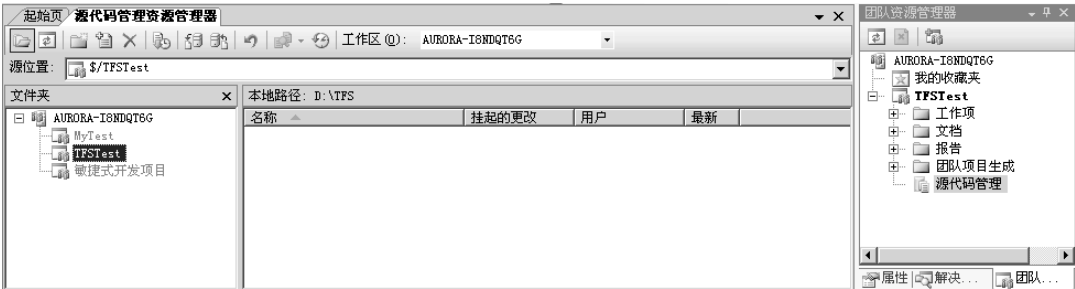


图 16-28 源代码管理

如果用户第 1 次使用 TFS 的源代码管理工具,如图 16-29 所示,那么可以在工具栏中打开【工作区】菜单,设置 TFS 团队项目在本地机器上对应的源代码目录。

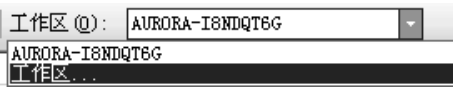


图 16-29 设置本地机器上的工作区域

在列表中选择【工作区】，如图 16-30 所示，在系统弹出的窗口中设置 TFS 团队项目在客户端本地的映射目录。用户只需要在“源代码管理文件夹”列中设置 TFS 数据库中的源代码文件夹，并在“本地文件夹”中设置客户端用于下载并保存源代码的目录，然后单击【确定】按钮即可。

当用户通过 Visual Studio 2005 创建一个项目时，在弹出对话框中选择【添加到源代码管理】并单击【确定】按钮后，如图 16-31 所示，用户可以在弹出对话框中选择将源代码的解决方案保存到哪个 TFS 团队项目中。



图 16-30 映射 TFS 团队项目在本地的目录



图 16-31 添加解决方案到 TFS 团队项目中

单击【确定】按钮，如图 16-32 所示，用户可以在“解决方案资源管理器”中看到项目的源代码前都添加了版本管理的标记，这就证明项目的源代码都处于版本管理工具的控制下。



图 16-32 处于版本管理工具下的源代码

用户右键单击源代码，在弹出菜单中选择【签入】子菜单，则会显示待签入的源代码的列表。用户单击【工作项】标签，如图 16-33 所示，即可以选择本次签入代码所对应的工作项。这样就可以标记哪些工作任务已经完成并提交到服务器。



图 16-33 在签入操作中选择对应的工作项

单击【签入说明】标签，如图 16-34 所示，如果本次签入的代码由相关人员对其进行了评审或代码走查，则可以在此注明评审人员的名单。



图 16-34 在签入操作中注明评审的参与者

在对源代码进行管理时可以按照 CMMI 的配置管理理论来对源代码的目录进行规划，如图 16-35 所示，可以在源代码管理中分别创建“开发库”、“受控库”和“基线库”的目录，配置管理理论详细内容请参见本书第 6 章的描述。

在软件开发过程中经常需要发布基线，当基线发布后又时常会需要修改其中的缺陷，那么软件研发人员可以通过 TFS 源代码管理的“分支”与“合并”功能进行操作，其流程如图 16-36 所示。



图 16-35 使用 TFS 建立配置库

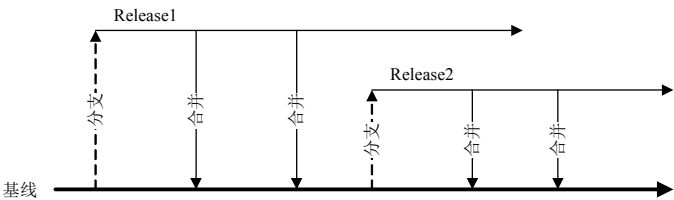


图 16-36 代码基线分支与合并示意图

根据示意图的描述以及 CMMI 配置管理的理论，当产品发布后会在“基线库”中进行保存，当需要修改缺陷时，如图 16-37 所示，右键单击“基线库”中源代码的目录并选择【分支】子菜单。

如图 16-38 所示，在弹出窗口的【目标】中输入代码分支将被创建的位置，通常分支会被创建到“开发库”的某个目录中，以便开发人员对代码进行修改。



图 16-37 准备创建代码分支

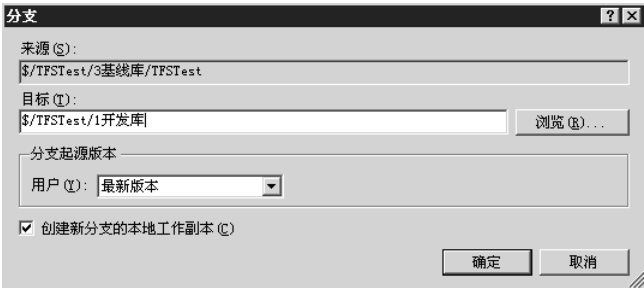


图 16-38 选择被创建的分支所在目录



图 16-39 创建分支成功

如图 16-39 所示，用户单击【确定】按钮后，分支就创建成功。

当缺陷修改完成后，用户可以右键单击“开发库”中的源代码目录，在弹出菜单中选择【合并】子菜单。如图 16-40 所示，在【源分支】中显示“开发库”源代码所在的目录，用户在【目标分支】中应该选择“基线库”中的源代码文件夹。

单击【下一步】按钮，在弹出的窗口中可以选择将“开发库”中某个版本的代码合并

到“基线库”中，如图 16-41 所示。



图 16-40 源代码管理合并向导对话框

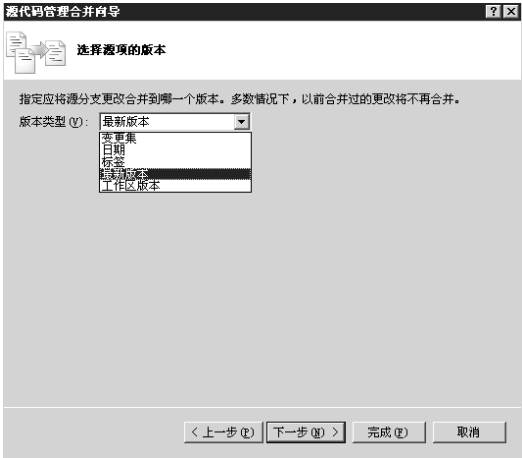


图 16-41 选择待合并的代码

单击【完成】按钮，TFS 会自动对比“开发库”和“基线库”中代码的差异，并对代码进行合并。如图 16-42 所示，被合并后的代码 TFS 通过特殊标记进行注明，以使用户对其进行内容核对和检查。

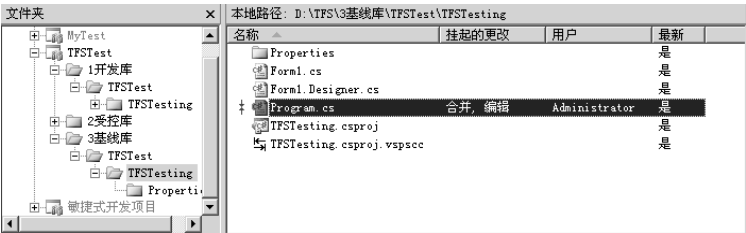


图 16-42 源代码合并成功

16.2.4 项目门户

TFS 通过微软的 SharePoint 为每个项目都提供了一个门户网站，这样可以使项目的相关关系人在任何时候、任何地点都可以通过该项目的门户网站了解项目的动态。如图 16-43 所示，在“团队资源管理器”中右键单击某个需要浏览的团队项目，在弹出菜单中选择【显示项目门户】子菜单。

TFS 会通过 IIS 将项目门户网站进行发布，如图 16-44 所示，用户可以通过 TFS 团队项目门户网站查看项目的各种文档以及通过报表功能来了解项目的各种动态。



图 16-43 创建项目门户



图 16-44 TFS 团队项目门户首页

1. 添加新通知

用户可以在 TFS 团队项目首页中发布有关项目的通知，以便让所有项目关系人无论身处何处，都可以了解到项目的各种重要的情况或事件。单击首页上【添加新通知】的链接，如图 16-45 所示，在页面上填写通知的“标题”和“正文”以及通知的结束日期，然后单击【保存并关闭】按钮来发布新的通知。

项目通知创建完毕后，如图 16-46 所示，会在首页上显示通知的内容、创建的时间和发布通知的作者名称等信息。



图 16-45 新建通知



图 16-46 在门户首页上发布的项目通知

2. 添加新链接

用户可以在 TFS 团队项目门户首页上添加与项目相关的友情链接，单击首页上的【添加新链接】，如图 16-47 所示，在界面上填写新建链接的地址和说明信息。

3. 添加文档

团队项目门户首页上显示的文档分类与 TFS “团队资源管理器”中“文档”目录下的分类相同，都是由“开发”、“项目管理”、“要求”、“安全性”、“测试”和“项目文档库”组成。例如：单击首页上“文档”分类下的【开发】链接，如图 16-48 所示，用户可以从项目门户网站创建或上传文档到 TFS 团队项目中。单击【新建文档】按钮，TFS 会自动打开 Word 给用户进行编写。



图 16-47 创建新链接

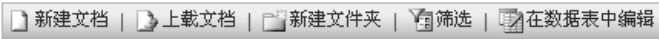


图 16-48 文档类工具栏

当文档完成后，单击 Word 文档的【保存】按钮，如图 16-49 所示，在弹出的对话框中显示的是 TFS 团队项目的目录结构，用户可以将新建的文档保存至“团队资源管理器”的相应目录下。

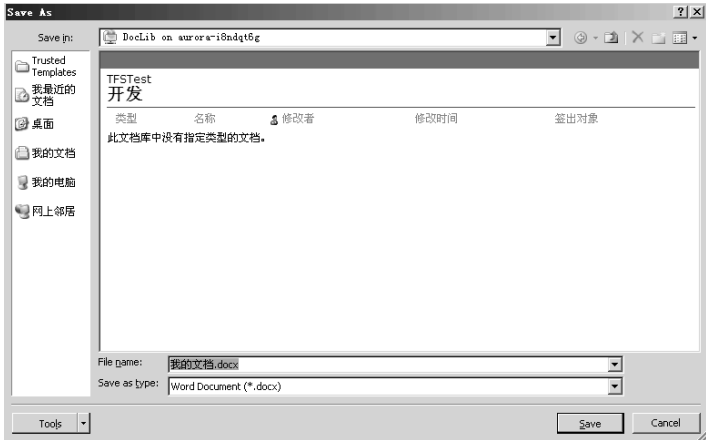


图 16-49 保存新建的开发文档

当文档保存成功后，如图 16-50 所示，在 TFS 项目团队门户首页中就可以看到新创建的文档。用户打开“团队资源管理器”，如图 16-51 所示，在相应的目录下即可查询到刚

刚创建的那份文档。



图 16-50 在 TFS 项目团队门户网站中查询新创建的开发类文档

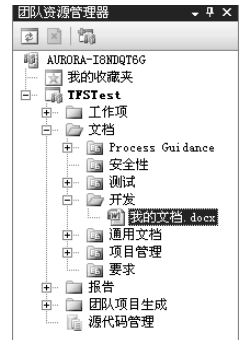


图 16-51 在“团队资源管理器”中查询新创建的开发类文档

4. 过程指南

在创建团队项目时，TFS 会提供两种不同类型的标准过程模型供项目组选用：一种是用于小型项目快速开发的敏捷式开发模型；一种是用于大团队开发的 CMMI 模型。为了让项目团队成员更好地对所选择地开发模型进行理解，TFS 特在门户网站中增加了对这两种开发过程的使用指南，如图 16-52 所示。



图 16-52 TFS 提供的两种过程指南

5. 创建图片库

用户可以通过项目门户网站来建立项目的图片库，图片库实际上就是存放图片的文件

夹，也是对项目图片的分类。单击门户首页上的【创建】按钮，然后选择【图片库】视图，在打开的页面中单击【图片库】按钮，在打开的窗口中，用户可以填写图片库的“名称”和“说明”，以及图片的版本控制和是否要在团队项目门户首页上显示，如图 16-53 所示。

创建完毕后的效果如图 16-54 所示，用户选择相应的“图片库”，然后单击【添加图片】按钮就可以将项目的各种图片进行上传并保存。

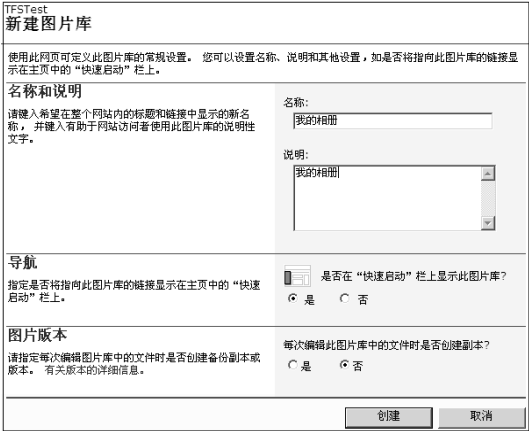


图 16-53 新建图片库



图 16-54 上传图片

6. 创建讨论区

在团队项目中用户还可以建立各种类型的讨论区，以便项目关系人对技术、风险、问题等进行讨论。在门户网站中单击【创建】按钮，然后在视图中选择【讨论板】，如图 16-55 所示，在页面上可以填写讨论区的“名称”和“说明”，以及是否在团队项目门户首页上显示。

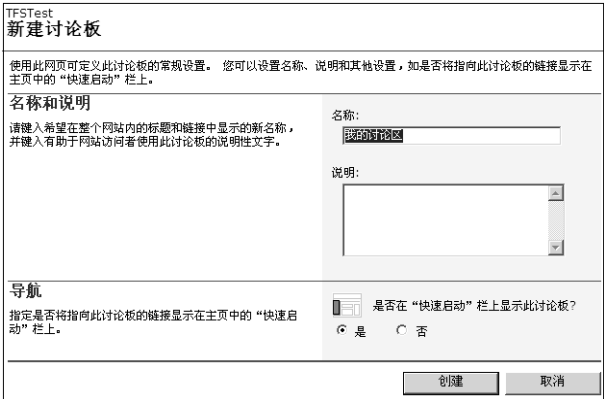


图 16-55 新建讨论区

当讨论区创建完毕后，如图 16-56 所示，用户可以在讨论区中单击【新建讨论】按钮来发起需要讨论的话题，例如：“Java 技术如何发展”等。

如果项目关系人对某个讨论的话题感兴趣，那么可以在该讨论话题中单击【提交答复】按钮来进行回复，如图 16-57 所示。



图 16-56 新建讨论话题



图 16-57 回复讨论的话题

7. 问卷调查

用户可以通过项目门户对某个问题创建问卷调查，以收集项目关系人的建议，也可以用于同行评审准备阶段过程中收集与会人员的建议和反馈。如图 16-58 所示，用户可以填写问卷调查的“名称”、“说明”以及设置问卷调查是否在首页的“快速启动”栏中显示，并且还可以设置问卷调查的相关属性，例如：“是否在调查结果中显示用户名”和“是否允许多次答复”。

问卷调查的主题创建完毕后，就可以定义问卷中所需要调查的问题。如图 16-59 所示，用户可以在文本框中填写“问题”的内容，并选择该问题以何种方式显示。单击【下一个问题】按钮，用户可以通过这种方式对问卷中的问题进行逐一创建。

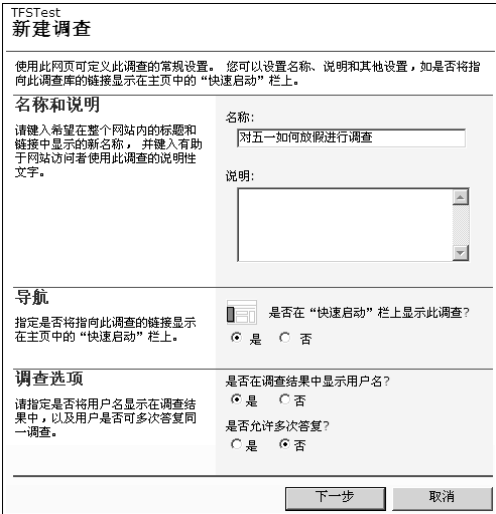


图 16-58 新建问卷调查

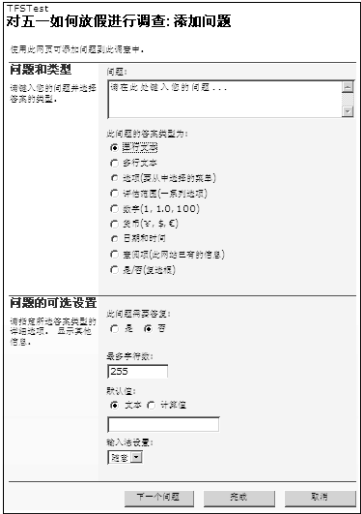


图 16-59 设置问题的内容和显示的形式

项目关系人可以通过门户网站参与问卷的调查，如图 16-60 所示，在参与问卷调查的过程中，用户可以单击【保存并关闭】按钮来提交问卷的答案。

TFS 团队项目门户还可以对问卷调查的结果进行统计，只要用户在问卷调查页面中单击【显示答复的图形汇总】按钮，如图 16-61 所示，TFS 会以图形方式对问卷调查的结果进行统计并显示。

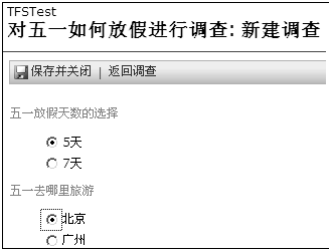


图 16-60 用户参与问卷调查

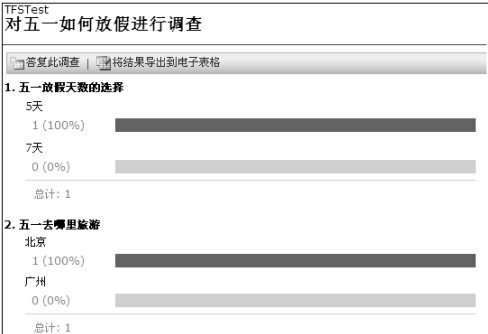


图 16-61 以图形方式显示问卷调查的结果

8. 项目报告

TFS 在 SQL Server 的 Reporting Service 基础上提供了强大的报表功能，供项目组对项目的度量数据进行统计和分析。用户可以在“团队资源管理器”中右键单击文件夹“报告”，在弹出菜单中选择【显示报告站点】，如图 16-62 所示，TFS 会打开已有报告的列表。

内容		属性			
图标	名称	说明	修改日期	修改者	运行时间
	Bug (检查未解决)		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	Bug 比率		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	概述		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	带有测试结果的工作项		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	带有任务的工作项		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	负载测试详细信息		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	负载测试摘要		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	软件 实际质量与计划速度		2009/3/11 16:27	AURORA-IBNDQT6G\Administrator	
	工作项		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	回归测试		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	会集		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	计划外工作		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	剩余工作		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	实际质量与计划速度		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	通过测试，但有活动 Bug		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	未通过测试，无活动 Bug		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	问题相关的工作项		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	相关的工作项		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	项目速度		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	要求测试历史记录和概述		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	要求详细信息		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	失败的无时测试的 Bug 数		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	质量预警		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	
	重新激活		2009/3/4 19:48	AURORA-IBNDQT6G\Administrator	

图 16-62 TFS 团队项目报告站点页面

用户双击某个报告，如图 16-63 所示，在打开的页面中可以定义查询条件，单击【查看报表】按钮即可进行查看。



图 16-63 查看项目的报告

16.3 Team Build

软件开发过程中离不开产品的集成，日构建的工作就可以通过 TFS 的 Team Build 工具来自动实现。用户可以运行 TFS 安装盘 Build 目录下的 Setup.exe 文件来安装 Team Build 插件。安装完毕后，右键单击团队项目的“团队项目生成”目录，选择【新建团队项目生成类型】子菜单，在弹出对话框中填写“团队项目生成类型”的名称和说明，如图 16-64 所示。



图 16-64 新建“团队项目生成类型”

单击【下一步】按钮，在弹出对话框中选择需要进行构建的解决方案名称，如图 16-65 所示。

单击【下一步】按钮，用户可以选择构建 Release 版本的程序还是 Debug 版本的程序，如图 16-66 所示，在【平台】列表中可以选所构建的程序所适用的硬件平台，例如：32 位的操作系统还是 64 位的操作系统。



图 16-65 选择需要生成的解决方案



图 16-66 配置构建的类型和平台

单击【下一步】按钮，用户可以选择在哪台机器上进行构建的工作，以及构建时源代码存放的目录和构建后执行程序所输出的目录，如图 16-67 所示。

单击【下一步】按钮，用户可以选择在构建过程中是否要执行单元测试，以及是否要对单元测试的结果进行覆盖率分析，如图 16-68 所示。最后单击【完成】按钮，结束“团队项目生成类型”的创建。



图 16-67 选择构建的位置



图 16-68 选择在构建过程中是否执行单元测试

用户在“团队资源管理器”中右键单击“团队项目生成”目录，然后选择【生成团队项目】子菜单来对团队项目的代码进行构建。如图 16-69 所示，在 TFS 打开的界面中显示构建的步骤和日志。

如果团队项目构建成功，那么在构建输出目录里会看到构建好的可执行程序，如图 16-70 所示。

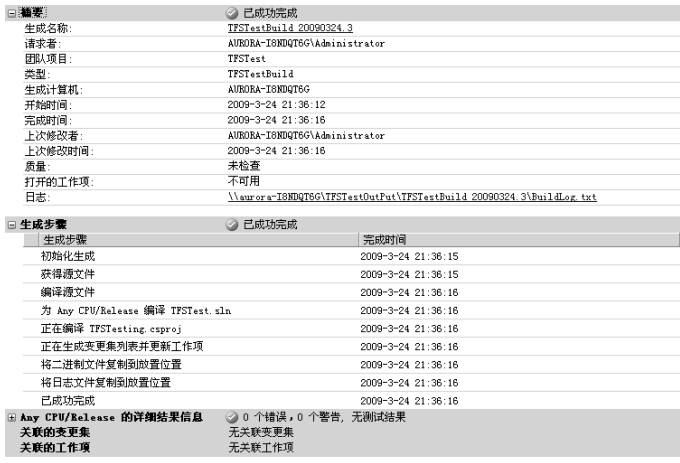


图 16-69 运行“团队项目生成类型”

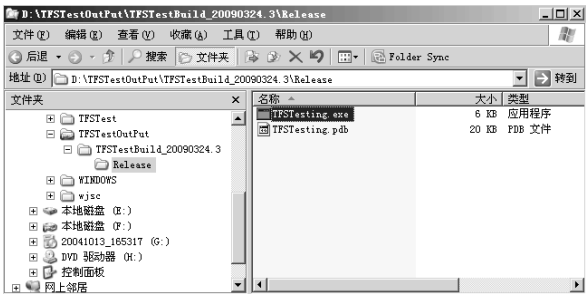


图 16-70 团队项目构建的结果

16.4 小结

通过以上对 TFS 功能的简单介绍，可以了解到 TFS 是一个全面的软件研发管理平台，它代表了软件研发工具发展的潮流。用户可以通过工作项对项目的任务、缺陷、风险等进行有效的跟踪和管理。集成了微软的 Project 和 Office 等软件的 TFS 更符合软件项目管理人员日常的工作习惯，使项目管理与研发过程合二为一。TFS 特有的 Team Build 工具使得自动化的产品集成与每日构建成为现实。在 SQL Server 的 Reporting Service 基础上提供的报表功能为项目管理人员准确了解项目动态提供了有力而准确的支持。通过 Share Point 和 IIS 构建的项目门户网站给项目关系人了解项目带来了便利。由此可见，TFS 是一个非常强大的软件研发平台，它将软件开发、测试和管理工作进行有效整合，提高了软件研发的效率，增强了各个角色之间的沟通。

A

附录A

思考题答案

■ 第 1 章

1. CMMI 第 3 级的 18 个 PA 分别是什么？

答案：如表 A-1 所示。

A-1 CMMI 第 3 级 PA 的列表

英文缩写	中文名称
ReqM	需求管理
PP	项目计划
PMC	项目监控
SAM	供应商管理
MA	度量管理
PPQA	过程与产品质量保证
CM	配置管理
RD	需求开发
TS	技术解决方案
PI	产品集成
VER	验证
VAL	确认
IPM	集成项目管理

续表

英文缩写	中文名称
RSKM	风险管理
DAR	决策分析
OPF	组织过程焦点
OPD	组织过程定义
OT	组织培训

2. 软件质量的提高是通过软件测试还是软件研发各过程的整体提高？为什么？

答案：软件质量的提高是通过软件研发各个过程的整体提高来实现的。因为软件测试只是起到弥补质量缺陷的作用，而不能避免缺陷的产生。要想避免缺陷的产生必须通过软件工程中各个环节系统、全面地提升才能最终实现软件质量的提高。

■ 第 2 章

1. 软件测试抽象后的 3 个步骤是什么？

答案：验证的准备工作、验证的执行工作、纠正措施。

2. 戴明环的 4 个要素是什么？

答案：PDCA：计划（Plan）、执行（Do）、检查（Check）、纠正（Action）

3. 单元测试、集成测试、系统测试、验收测试的依据和准则是什么？

答案：单元测试的依据是代码；集成测试的依据是详细设计；系统测试的依据是概要设计；验收测试的依据是需求分析文档

■ 第 3 章

1. 软件确认的准备工作有哪些？

答案：选择需要确认的工作产品与产品组件；建立和维护确认环境；建立确认的流程及准则

2. 常用的确认方法有哪两大类？

答案：对文档类型的工作产品进行确认，通常可以与其文档的评审合并进行；对产品或产品组件进行确认时，通常可以与单元测试、集成测试、系统测试和验收测试合并进行。

3. 确认是如何来提高团队合作信任感的？

答案：软件生命周期内凡是一个环节“输出”的工作成果将成为后续环节的“输入”，那么上一个环节的生产者要承诺该工作产品是符合质量要求的，后续环节的“使用者”也要对其工作产品进行确认。这就好比“亲兄弟明算账”，通过这样的方式来建立相互间的信任关系。

■ 第 4 章

1. 同行评审分为几个阶段？

答案：同行评审分为计划阶段、启动阶段、执行阶段、收尾阶段。

2. 同行评审的结果有几种?

答案: 同行评审的结果分为通过、有条件通过和不通过三种情况。

■ 第 5 章

1. PPQA 的英文全称是什么, 分别代表对哪两个方面的审计?

答案: Process and Product Quality Assurance, 分别代表了对过程和产品的审计。

2. 过程审计关注的重点是什么?

答案: QA 对过程审计时关注的是项目对已定义过程的遵循程度。

3. 产品审计关注的重点是什么?

答案: QA 对产品审计时关注的是工作产品的正确性。

■ 第 6 章

1. 什么是配置项?

答案: 配置项应该具有以下特点之一:

- 交付或非交付的产品、工作产品或工具
- 可能被两个或更多小组或人共享的工作产品
- 随时间改变而改变的工作产品
- 具有相互依赖性的工作产品, 其中一个的改变时将会影响其他工作产品
- 重要性高的工作产品

2. 基线与配置项的关系是什么?

答案: 配置项与基线就好比“一条绳上的蚂蚱”, 每个“蚂蚱”就是一个配置项, 用绳子穿起来就成了基线。

3. 配置库通常划分为几个部分, 分别是什么?

答案: 配置库通常划分为开发库、受控库、基线库。

4. 通过什么样的方式对软件配置工作进行审计?

答案: 通过物理审计和功能审计的方式对软件配置工作进行审计。

■ 第 7 章

1. 度量和测量有什么区别?

答案: 测量就是通过某种刻度对物体进行刻画的过程。度量是在测量基础上进行的一系列活动, 它需要进行多次相同的测量工作, 从而可以收集、存储大量的测量数据。然后将这些数据按照度量的目的进行统计和分析, 项目管理人员再利用分析的结果进行判断和预测, 最后将度量的结果分发给项目相关人员。由此可见测试只是度量活动的一个子集。

2. 什么是基础数据, 什么是派生数据?

答案: 基础数据就是直接测量中使用的数据, 该数据的特点是都属于同一实体对象的同一类型的属性。派生数据是指经过间接测量后得到的结果, 是一系列数学运算或者经济模型将基础数据转换后得到的结果。

3. 什么是软件度量所使用的指示器？

答案：指示器为进行一次或多次测量后得到结果的体现，它是支持用户分析和决策所导出的信息，这种信息通常以图形或图表的形式进行展现。

4. 常用的指示器有哪些？

答案：散点图、直方图、对比直方图、比例直方图、趋势图、帕累托图、控制图。

■ 第 8 章

1. 风险的可能性和严重性在项目生命周期中是如何变化的？

答案：随着项目的进展，风险的可能性在逐步降低，风险的严重性在逐步增加。

2. 对于“已知-未知”类型的风险应该采取什么样的应对措施？

答案：“已知-未知”类型的风险可以识别和分析，但由于其可能性和严重性无法预测，因此很多软件项目会在项目计划中制定一些风险储备来对其进行应对。

3. 风险识别的方法有哪些？

答案：头脑风暴法、德尔菲法、参考历史资料、SWOT 分析法、鱼骨图法、检查表法。

4. 常用的风险属性有哪些？

答案：风险发生的可能性、风险影响的严重性、 $\text{风险值} = \text{风险发生的可能性} \times \text{风险的严重性}$ 、阈值、风险的预防措施、风险预防措施责任人、风险的应对措施、风险应对措施责任人。

■ 第 9 章

1. “裁剪”与“裁减”有什么不同？

答案：“裁剪”的过程也就是对项目整体进行策划的过程，可能是对过程中的某个环节“增加”或“减少”一些内容。“裁减”是只减少不增加。

2. 项目的依存关系有几种，分别是什么？

答案：强制性依赖是指所从事的工作中固有的依存关系，又称为硬逻辑；可选择的依赖是项目管理团队所规定的依赖关系，通常是一些经验或最佳实践。也称为优先选用逻辑关系、软逻辑。外部依赖是指项目任务与项目外部活动之间的依赖关系。例如：设备是否到位。

■ 第 10 章

1. IFPUG 国际标准功能点估算法中对功能点分为了哪几类？

答案：ILF: Internal Logical File 内部逻辑文件；EIF: External Interface File 外部接口文件；EI: External Input 外部输入；EO: External Output 外部输出；EQ: External Inquiry 外部查询。

2. 项目的范围与产品的范围有什么不同？

答案：项目范围指的是完成产品、服务的功能或特性所需要的工作。产品范围是否完成是以产品的功能或特征是否实现为衡量的标准。

3. 任务与任务之间的逻辑关系有几种？哪种是最常见的？

答案：“完成~开始”、“开始~完成”、“完成~完成”、“开始~开始”。

■ 第 11 章

1. 项目监控的主要参数和次要参数都有哪些？

答案：主要对项目的进度、成本、工作成果的质量进行监控；次要参数是监控工作产品或工作任务的属性、监控资源的提供与使用、监控项目人员的知识与技能、监控承诺事项、监控数据管理、监控关键人员的参与、监控项目的风险。

2. 项目监控的方法有几种？分别是什么？

答案：定期监控是指项目组可以通过定期召开周例会、月例会的方式来对项目的各种参数进行跟踪和监控；阶段性监控是以项目生命周期各阶段的里程碑为标记，通过里程碑的评审会议来对项目的各种参数进行跟踪和监控；事件触发性监控是指当项目发生重大变更时都需要对项目的某些参数进行跟踪和监控。

3. 在挣值分析中 CV 和 SV 分别代表什么？

答案：Cost Variance (CV) 成本偏差、Schedule Variance (SV) 进度偏差。

4. 在挣值分析中 CPI 和 SPI 分别代表什么？

答案：Cost Performance Index (CPI) 成本绩效指数、Schedule Performance Index (SPI) 进度绩效指数。

■ 第 12 章

1. 需求工程分为哪两大部分？

答案：需求开发和需求管理两大部分。

2. 需求调研、需求分析与需求规格化之间是怎样的关系？

答案：需求调研、需求分析与需求规格化之间的关系如图 A-1 所示。

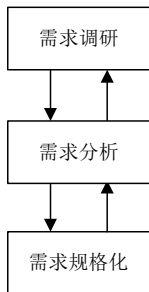


图 A-1 需求调研、需求分析与需求规格化之间的关系

3. 需求管理有哪些部分组成？

答案：软件需求管理由以下 5 个部分组成：

- (1) 取得对需求的理解
- (2) 管理需求的变更
- (3) 获取对需求变更的承诺
- (4) 维护需求的双向跟踪性
- (5) 识别需求与工作产品之间的差异

4. 通过什么来记录和维护需求与工作产品间双向跟踪的关系？

答案：通过“需求跟踪矩阵”来记录和维护需求与工作产品间双向跟踪的关系。

■ 第 13 章

1. 决策分析都包含了哪些步骤？

答案：建立决策分析指南、建立决策的准则、选择评估的方法、识别候选解决方案的提案、评估候选解决方案的提案、进行决策选择解决方案。

2. 决策分析常用的方法有哪些？

答案：模拟、调查、基于经验、基于原型进行推断、测试、加权打分法、一票否决法。

3. 决策分析与同行评审和软件确认过程有哪些相似的地方？

答案：它们相似的地方是都可以采用评审的方式进行工作，也都需要制定相关的准则。

4. 决策分析与同行评审和软件确认过程有哪些不同的地方？

答案：它们不同的地方是决策分析的结果是项目在后续的过程中会按照该解决方案进行实施，是为未来发展进行规划；软件确认过程的结果是活动相应的承诺并且建立信任的机制；同行评审的结果是对工作产品有效性、正确性的验证。

■ 第 14 章

1. 软件产品集成都需要做哪些准备工作？

答案：确定产品与产品组件集成的顺序；确定集成产品或产品组件的环境；建立产品或产品组件集成的流程与准则。

2. 集成并交付产品的过程中都需要做哪些工作？

答案：确认要集成的产品组件已准备就绪；集成产品组件；评估集成后的产品组件。

3. 持续性集成是通过什么方式来实现的？

答案：持续集成通过日构建的方式实现。

4. 持续性集成对软件质量管理的好处有哪些？

答案：持续性集成的好处有：可以提早发现问题并解决问题，增强了项目的透明度和可视性，给各级管理人员更多的安全感和信心。而且还提供了一个日常沟通的渠道，大家一起总结经验、讨论问题以避免问题的重复发生。

■ 第 15 章

1. MSF 生命周期模型分为几个阶段？

答案：构思阶段、计划阶段、开发阶段、稳定阶段、部署阶段。

2. 裁剪指南包含哪些内容？

答案：定义项目的级别、定义标准项目周期及里程碑、定义标准项目生命周期产出物、定义标准的度量目标、定义标准的产品基线。

3. 软件质量持续改进的三驾马车是什么？

答案：OPF 组织过程改进焦点、OPD 组织标准过程的定义、OT 组织级培训。



《软件质量管理指南》读者交流区

尊敬的读者：

感谢您选择我们出版的图书，您的支持与信任是我们持续上升的动力。为了使您能通过本书更透彻地了解相关领域，更深入的学习相关技术，我们将特别为您提供一系列后续的服务，包括：

1. 提供本书的修订和升级内容、相关配套资料；
2. 本书作者的见面会信息或网络视频的沟通活动；
3. 相关领域的培训优惠等。

请您抽出宝贵的时间将您的个人信息和需求反馈给我们，以便我们及时与您取得联系。

您可以任意选择以下三种方式与我们联系，我们都将记录和保存您的信息，并给您提供不定期的信息反馈。

1. 短信

您只需编写如下短信：B09010+您的需求+您的建议

发送到1066 6666 789（本服务免费，短信资费按照相应电信运营商正常标准收取，无其他信息收费）

为保证我们对您的服务质量，如果您在发送短信24小时后，尚未收到我们的回复信息，请直接拨打电话（010）88254369。

2. 电子邮件

您可以发邮件至jsj@phei.com.cn或editor@broadview.com.cn。

3. 信件

您可以写信至如下地址：北京万寿路173信箱博文视点，邮编：100036。

如果您选择第2种或第3种方式，您还可以告诉我们更多有关您个人的情况，及您对本书的意见、评论等，内容可以包括：

- （1）您的姓名、职业、您关注的领域、您的电话、E-mail地址或通信地址；
- （2）您了解新书信息的途径、影响您购买图书的因素；
- （3）您对本书的意见、您读过的同领域的图书、您还希望增加的图书、您希望参加的培训等。

如果您在后期想退出读者俱乐部，停止接收后续资讯，只需发送“B09010+退订”至10666666789即可，或者编写邮件“B09010+退订+手机号码+需退订的邮箱地址”发送至邮箱：market@broadview.com.cn 亦可取消该项服务。

同时，我们非常欢迎您为本书撰写书评，将您的切身感受变成文字与广大书友共享。我们将挑选特别优秀的作品转载在我们的网站（www.broadview.com.cn）上，或推荐至CSDN.NET等专业网站上发表，被发表的书评的作者将获得价值50元的博文视点图书奖励。

我们期待您的消息！

博文视点愿与所有爱书的人一起，共同学习，共同进步！

通信地址：北京万寿路 173 信箱 博文视点（100036）

电话：010-51260888

www.phei.com.cn

www.broadview.com.cn



電子工業出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

E-mail : jsj@phei.com.cn , editor@broadview.com.cn

Broadview[®]
www.broadview.com.cn

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396；（010）88258888

传 真：（010）88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036