

浙江省高等教育重点教材

软件工程基础

编著 周 苏 王 文
吴 艳

RUANJIAN GONGCHENG JICHU



浙江科学技术出版社

图书在版编目(CIP)数据

软件工程基础/周苏,王文,吴艳编著. —杭州:浙江科学技术出版社,2008.8

(浙江省高等教育重点教材)

ISBN 978-7-5341-3371-8

I. 软… II. ①周…②王…③吴… III. 软件工程—高等学校—教材 IV. TP311.5

中国版本图书馆 CIP 数据核字(2008)第 110456 号

丛 书 名 浙江省高等教育重点教材
书 名 软件工程基础
编 著 周 苏 王 文 吴 艳

出版发行 浙江科学技术出版社

杭州市体育场路 347 号 邮政编码: 310006

联系电话: 0571-85170300-61712

E-mail: cl@zkpress.com

排 版 杭州大漠照排印刷有限公司制作

印 刷 杭州丰源印刷有限公司

开 本 787×1092 1/16 印张 22.75

字 数 505 000

版 次 2008 年 8 月第 1 版 2008 年 8 月第 1 次印刷

书 号 ISBN 978-7-5341-3371-8 定价: 39.50 元

版权所有 翻印必究

(图书出现倒装、缺页等印装质量问题,本社负责调换)

策划组稿 张祝娟 封面设计 金 晖
责任校对 张 宁 责任印务 李 静

目 录

▶ 第一章 软件工程概述	1
第一节 计算机系统及其软件的发展	1
第二节 软件、软件生存周期和软件生存周期过程	3
第三节 软件生存周期模型	5
第四节 软件工程定义	8
第五节 软件工具与环境	11
第六节 软件工程的发展	13
主要术语	14
阅读：《人月神话》作者布鲁克斯	14
习题与思考	16
实验：软件工程的计算环境	16
▶ 第二章 系统定义与软件计划	20
第一节 系统定义	20
第二节 软件计划	29
第三节 进度安排	31
第四节 计划文件与复审	32
主要术语	33
阅读：软件思想家杰拉尔德·温伯格	33
习题与思考	34
实验：工具、环境与 CASE	35
▶ 第三章 软件需求分析	42
第一节 需求分析阶段的任务	42
第二节 结构化分析方法	43
第三节 数据流程图	45
第四节 数据字典	48
第五节 加工的分析与表达	51
第六节 需求分析文件与复审	55
主要术语	57



阅读: SA/SD 研究的领导者 Edward Yourdon	57
习题与思考	58
实验: 软件开发绘图工具 Visio	59
▶ 第四章 软件概要设计	70
第一节 模块的划分	70
第二节 结构化设计方法	75
第三节 Parnas 方法	79
第四节 Jackson 方法	80
第五节 程序的逻辑构造方法	81
第六节 概要设计文件与复审	81
主要术语	82
阅读:《未来之路》和《数字化生存》.....	83
习题与思考	83
实验: 用 Visio 绘制工程图形	86
▶ 第五章 软件详细设计	92
第一节 概述	92
第二节 结构化构造	93
第三节 图形设计工具	94
第四节 伪码与程序设计语言	100
第五节 各种详细设计工具的比较	101
第六节 详细设计文件与复审	102
主要术语	103
阅读: 软件产业的设计大师——VB 之父 Alan Cooper	103
习题与思考	104
实验: 软件工程国家标准	105
▶ 第六章 软件编码	113
第一节 结构化程序设计方法	113
第二节 程序设计风格	114
第三节 源代码文件	116
第四节 程序设计与评价	118
第五节 编程语言的特点	120
第六节 选择编程语言	123
第七节 编码文件与复审	125
主要术语	125
阅读: 19 世纪的传奇合作——巴贝奇与阿达	126
习题与思考	127

实验：指定或自选项目编制软件需求分析与概要设计文档	130
▶ 第七章 软件测试	132
第一节 测试的基本概念	132
第二节 测试方法	134
第三节 单元测试	136
第四节 组装测试	137
第五节 确认测试	138
第六节 测试用例设计	140
第七节 测试工具与测试自动化	144
第八节 测试文件与复审	147
主要术语	150
阅读：从程序员到软件测试工程师	151
习题与思考	154
实验：软件自动化测试环境	159
▶ 第八章 面向对象分析与设计	163
第一节 面向对象方法	163
第二节 面向对象的概念	166
第三节 面向对象软件的开发过程	167
第四节 面向对象分析 OOA	170
第五节 面向对象设计 OOD	175
主要术语	180
阅读：极限编程 XP 方法学的先驱 Kent Beck	181
习题与思考	182
实验：了解功能测试软件 WinRunner	185
▶ 第九章 面向对象的实现	192
第一节 面向对象编程	192
第二节 面向对象测试	194
第三节 OO 软件的测试用例设计	198
主要术语	200
阅读：CASE 与信息工程的创始人 James Martin	200
习题与思考	201
实验：PowerDesigner 入门	201
▶ 第十章 统一建模语言 UML	213
第一节 UML 概述	213
第二节 PowerDesigner 的 OOM	214



主要术语	217
阅读: 软件开发的教父 Martin Fowler	217
习题与思考	218
实验: PowerDesigner 面向对象模型	218
▶ 第十一章 软件文件	247
第一节 目的和作用	247
第二节 软件生存周期与各种文件的编制	248
第三节 文件编制中考虑的因素	249
第四节 文件编制的质量要求	251
第五节 文件的管理和维护	251
主要术语	254
阅读: 软件工程的 7 条基本原理	254
习题与思考	255
实验: 软件产品开发文件编制指南	256
▶ 第十二章 软件维护	263
第一节 概述	263
第二节 软件的可维护性	266
第三节 软件维护的管理	274
主要术语	275
阅读: 软件工程学科的内涵	276
习题与思考	277
实验: 软件项目管理 Project 初步	279
▶ 第十三章 质量评价与软件管理	284
第一节 软件项目的特点与软件管理职能	284
第二节 对软件质量的需求	285
第三节 软件质量度量	286
第四节 软件质量评估指标体系	288
第五节 软件评价	290
第六节 CMM: 软件能力成熟度模型	293
主要术语	294
阅读: 软件工程学科的相关学科	294
习题与思考	295
实验: 软件配置管理 VSS	297
▶ 第十四章 软件工程实验总结	307
第一节 实验的基本内容	307

第二节 实验的基本评价	309
第三节 课程学习能力测评	310
第四节 软件工程实验总结	311
第五节 实验总结评价(教师)	312
▶ 附录一 WinRunner 功能测试实践	313
▶ 附录二 Project 项目管理实践	331
▶ 参考文献	351

第一章 软件工程概述

随着计算机系统的迅速发展和应用范围的日益广泛,计算机软件的规模越来越大,其复杂程度也在不断增加。进入 20 世纪 60 年代以来,人们开始逐渐认识到确实存在着“软件危机”这样一个事实。例如:

- (1) 软件生产不能满足日益增长的需要。
- (2) 对软件开发成本和开发进度的估计往往不准确,实际成本有时高出预计成本好几倍,预计完工的时间往往推迟几个月,甚至更长时间。
- (3) 软件开发人员和用户之间信息交流不充分,用户对完成的软件满意度很低。
- (4) 软件价格昂贵,软件成本在整个计算机系统中所占的比例急剧上升,软件已成为许多计算机系统中花钱最多的项目。
- (5) 软件质量难以保证,质量保证技术还没有真正应用到软件开发的全过程。
- (6) 软件可维护性差,程序中的错误很难改正,或者当硬件环境发生变化时,想要进行适应性或完善性维护极其困难。

导致这一系列问题产生的一个重要原因就是,由于软件的研制和维护本身是工程性的任务,而软件人员所采取的方式却未能工程化。

为了克服“软件危机”,人们开始考虑采用工程化方法和工程途径来研制和维护软件。20 世纪 60 年代末至 70 年代初开始,逐渐发展起一组总称为“软件工程”的技术。这些技术把软件作为一个工程产品来处理,它需要计划、分析、设计、实现、测试以及维护。

第一节 计算机系统及其软件的发展

硬件工程和软件工程都可以看成是一门更广义的学科——计算机系统工程的一部分。

一、计算机系统工程

用于计算机硬件的工程技术是由电子设计技术发展起来的。今天,硬件设计技术已经达到了比较成熟的水平。虽然制造方法仍在不断地改进,但可靠性已是一种可以期待的现实,而不再是一种愿望了。

但是,计算机软件工程却还处于某种困境之中。一方面,在以计算机为基础的系统,软



件已经取代硬件成为系统中设计起来最困难、最不容易成功(按时完成和不超过预计的成本),而且是最不易管理的部分。另一方面,随着以计算机为基础的系统在数量、复杂程度和应用范围上的不断增长,对软件的需求却仍然有增无减。

计算机系统工程的主要内容是对系统所要求的功能加以揭示、分析,并把它们分配给系统的各个部分,如图 1-1 所示。

在大多数新系统创建时,对系统所要求的功能往往只有模糊的概念。系统分析和定义的目的在于明确项目的范围,这就要对需要进行处理的信息、所要求的功能、所期望的性能以及设计的约束和检验的标准等进行系统、详细的分析。

在范围确定之后,计算机系统工程师必须考虑多种能潜在的满足项目范围的、可供选择的配置。在综合考虑各项因素之后,选择其中的一种配置,并将系统的功能分配给系统的各个部分(例如硬件和软件)。

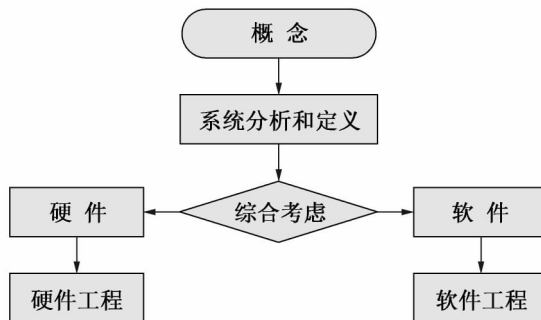


图 1-1 计算机系统工程

二、计算机软件的发展

软件发展的历史是与计算机系统发展紧密相关的。

在计算机系统发展的初期(20 世纪 50~60 年代),硬件经历了不断的变化,而软件则被多数人作为一种事后工作来看待,几乎没有什么系统的方法可以遵循。在此期间,多数系统采用批处理工作方式,硬件通常仅用来执行一个单一的为某个特定的应用目的而编制的程序。

当通用硬件已经成为平常的事情时,软件则仍然是为每一种用途而分别设计的,它们的通用性有限。大多数软件是由使用该软件的人或机构研制的,软件设计是在某个人的头脑中完成的一种隐含的过程。

计算机系统发展的第二个时期跨越了从 20 世纪 60 年代中期到 70 年代中期这 10 年,这个时期以产品化软件的使用和“软件车间”的出现为特征。多道程序设计、多用户系统引入了人机对话的新概念。人机对话技术为计算机的应用开辟了新的天地,并使硬件和软件提高到新的水平;实时系统能够在几毫秒内收集、分析和变换来自多个信息源的数据,进而控制处理过程并产生输出;联机辅助存储设备的发展导致了数据库管理系统的出现;人们开发软件以广泛销售为目的。

随着以计算机为基础的系统日益增多,计算机软件库开始膨胀。所有这些程序,当检测出故障、用户要求改变时,或者要使软件在新的硬件上运行时,都必须加以维护和修改,软件维护开销越来越大。更坏的是,许多软件所带有的个人色彩使得它们实际上是不可维护的。“软件危机”开始了。

计算机系统发展的第三个时期从 20 世纪 70 年代初期开始,分布式系统(多个计算机、各机器并行执行和相互通信)极大地增加了以计算机为基础的系统复杂性。由于微处理机和

有关部件的功能越来越强而价格越来越低,因此在最常用的计算机应用领域中,具有“嵌入智能”的产品取代了较大的计算机。此外,微处理机的出现也使得人们可能以极低的成本实现复杂的逻辑功能。

硬件的迅速发展超过了人们提供支持软件的能力。在第三个发展时期,软件危机日益严重。维护软件的费用迅速增长,而软件开发的生产率又跟不上对软件需求的步伐。为了对付不断增长的软件危机,软件工程开始得到认真的对待。如今,计算机系统的发展正在向第四个时期过渡,即从技术性应用向消费性市场过渡。计算机软件发展的3个时期详见表1-1。

表 1-1 计算机软件发展的 3 个时期

时 期 特 点	程 序 设 计	程 序 系 统	软 件 工 程
软件所指	程序	程序及说明书	程序、文档及数据
主要程序设计语言	汇编及机器语言	高级语言	软件语言*
软件工作范围	程序编写	包括设计和测试	软件生存周期
需求者	程序设计者本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大、中型软件开发机构
软件规模	小型	中、小型	大、中、小型
决定质量的因素	个人程序设计技术	小组技术水平	管理水平
开发技术和手段	子程序、程序库	结构化程序设计	数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络和分布式开发、面向对象技术、软件过程与过程改进
维护责任者	程序设计者	开发小组	专职维护人员
硬件特征	价格高、存储容量小、工作可靠性差	降价,速度、存储容量及工作可靠性有明显提高	向超高速、大容量、微型化及网络化方向发展
软件特征	完全不受重视	软件技术的发展不能满足需求,出现软件危机	开发技术有进步,但未获突破性进展,价格高,未完全摆脱软件危机

* 软件语言包括需求定义语言、软件功能语言、软件设计语言、程序设计语言等。

第二节 软件、软件生存周期和软件生存周期过程

2006年出版的《中国大百科全书》给软件下的定义是:软件是“计算机系统中的程序和有关的文件。程序是计算任务的处理对象和处理规则的描述;文件是为了便于了解程序所需的资料说明。程序必须装入机器内部才能工作,文件一般是给人看的,不一定装入机器。程序作为一种具有逻辑结构的信息,精确而完整地描述计算任务中的处理对象和处理规则。这一描述还必须通过相应的实体才能体现”。



也就是说,软件不仅仅是指程序,在软件研制过程中按一定规格产生的各种文件也是软件不可缺少的组成部分。

《软件工程术语》(GB/T11457—1995)定义了软件生存周期,即:从设计软件产品开始到产品不能再使用时为止的时间周期。亦即:一个计算机软件,从出现一个构思之日起,经过开发成功投入使用,在使用中不断增补修订,直到最后决定停止使用,并被另一款软件替代时止,被认为是该软件的一个生存周期(或称生命周期、生存期)。

一个软件产品的生存周期可以划分成若干个互相区别而又有联系的阶段,每个阶段中的工作均以上一阶段工作的结果为依据,并为下一阶段的工作提供了前提。经验表明,失误造成的差错越是发生在生存周期的前期,在系统交付使用时造成的影响和损失就越大,要纠正它所花费的代价也越高。因此,在前一阶段工作没有做好之前,决不要草率地进入下一阶段。

《软件生存周期过程》(GB/T8566—2001)则根据软件工程的实践和软件工程学科的发展,进一步完善了软件生存周期的定义,即:软件生存周期是从概念形成直到退役,并且由获取和供应软件产品及服务的各个过程所组成。该标准把软件生存周期中可以开展的活动分为5个基本过程(获取过程、供应过程、开发过程、运作过程、维护过程)、8个支持过程(文档编制过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程、审核过程、问题解决过程)和4个组织过程(管理过程、基础设施过程、改进过程、培训过程)。

软件生存周期过程中阶段的划分有助于软件研制管理人员借用传统工程的管理方法(重视工程性文件的编制、采用专业化分工方法、在不同阶段使用不同的人员等),从而有利于明显提高软件质量、降低成本、合理使用人才,进而提高软件开发的劳动生产率。

由于工作的对象和范围的不同以及经验的不同,对软件生存周期过程中各阶段的划分也不尽相同。但是,这些不同划分中有许多相同之处。《计算机软件开发规范》(GB/T8566—1988)(《软件生存周期过程》GB/T8566—2001的前身)将软件生存周期划分为以下8个阶段,即:可行性研究与计划、需求分析、概要设计、详细设计、实现(包括单元测试)、组装测试、确认测试、使用和维护。如图1-2所示为软件生存周期的瀑布模型。

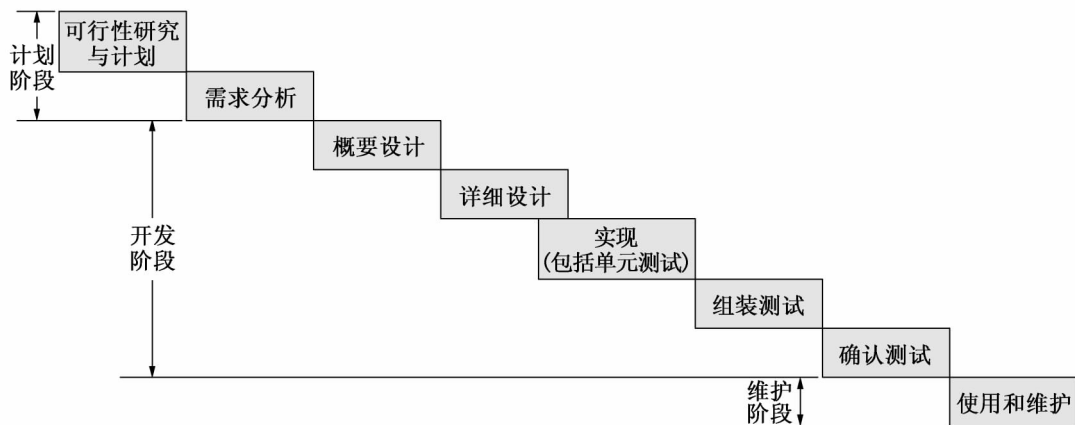


图 1-2 软件生存周期的瀑布模型

软件生存周期是对软件的一种长远发展的看法,这种看法把软件开始开发之前和软件交付使用之后的一些活动都包括在软件生存周期之内。

应当注意的是,软件系统的实际开发工作不可能直线地通过分析、设计、编程和测试等阶段,出现各阶段间的回复是不可避免的。

软件生存周期的每个阶段都要产生一定规格的软件文件移交给下一阶段,使下一阶段在所提供的软件文件的基础上继续开展工作。《计算机软件产品开发文件编制指南》(GB/T8567—1988,已升级为GB/T8567—2006《计算机软件文档编制规范》)建议在软件的开发过程中编制下述14种文件,即:可行性研究报告、项目开发计划、软件需求说明书、数据要求说明书、概要设计说明书、详细设计说明书、数据库设计说明书、用户手册、操作手册、模块开发卷宗、测试计划、测试分析报告、开发进度月报以及项目开发总结报告。而《计算机软件需求说明编制指南》(GB/T9385—1988)、《计算机软件测试文件编制规范》(GB/T9386—1988)等有关软件工程的国家标准对软件文件的编制提出了更为详尽的要求,《软件文档管理指南》(GB/T16680—1996)则明确了对软件文件的管理要求。

图1-3和图1-4分别说明了在典型的情况下,软件生存周期各阶段的工作量所占的比重。图1-3说明运行维护工作量要占整个生存周期工作量的一半以上,图1-4则说明了测试阶段(组装测试和确认测试)的工作量约占整个开发期工作量的近一半。

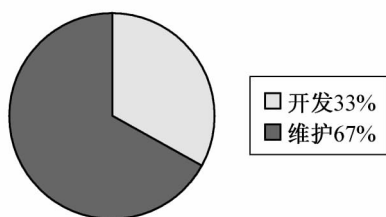


图 1-3 软件生存周期工作量分配

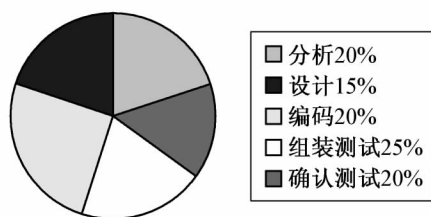


图 1-4 开发阶段工作量分配

第三节 软件生存周期模型

软件生存周期模型(又称软件开发模型)是软件工程的一个重要的概念,它可以定义为一个框架,含有遍历系统从确定需求到终止使用这一生存周期的软件产品的开发、运行和维护中需实施的过程、活动和任务。

软件生存周期模型清晰、直观地表达了软件开发的全过程,明确规定了开发工作各阶段所要完成的主要活动和任务,以作为软件项目开发工作的基础。对于不同的软件系统,可以采用不同的开发方法、使用不同的程序设计语言、挑选各种不同技能的人员参与工作、运用不同的管理方法和手段,以及允许采用不同的软件工具和不同的软件工程环境等。软件生存周期模型是稳定有效和普遍适用的。

在软件生存周期过程中,软件生存周期模型仅对软件的开发、运作和维护过程有意义,在信息技术国际标准 ISO12207 和 ISO9000-3 中都提到软件生存周期模型,包括瀑布模型、渐增模型、演化模型、螺旋模型、喷泉模型和智能模型等。



一、瀑布模型

瀑布模型(Waterfall Model)是 1970 年 W. Royce 提出的最早的软件开发模型(如图 1-2 所示),它将软件开发过程中的各项活动规定为依固定顺序连接的若干阶段工作,形如瀑布流水,最终得到软件系统或软件产品。换句话说,它将软件开发过程划分成若干个互相区别而又彼此联系的阶段,每个阶段中的工作都以上一个阶段工作的结果为依据,同时为下一个阶段的工作提供了前提。

瀑布模型之所以能广泛流行,一方面,由于它在支持开发结构化软件、控制软件开发复杂度、促进软件开发工程化方面起到了显著作用;另一方面,它为软件开发和维护提供了一种当时较为有效的管理模式,根据这一模式制订开发计划、进行成本预算、组织开发人员以阶段评审和文档控制为手段有效地对软件开发过程进行指导,从而使软件质量在一定程度上得到保证。1988 年曾依据该开发模型制定并公布了《软件开发规范》(GB8567—1988),对我国软件开发起到了较大的促进作用。

瀑布模型在大量的实践中也暴露出了不足和问题。例如,由于固定顺序,前期阶段工作中所造成的差错,越到后期阶段,所造成的损失和影响也越大,为了纠正它而花费的代价也越高,尽管技术人员小心翼翼,但这种情况还是会经常发生。因此,在评价瀑布模型时,应考虑相关风险,例如:

- (1) 需求未被充分理解。
- (2) 系统太大而一次不能做完所有的事。
- (3) 事先打算采用的技术迅速发生变化。
- (4) 需求迅速发生变化。
- (5) 有限的资源。
- (6) 无法利用某一中间产品。

二、渐增模型

渐增模型(Incremental Model)是指从一组给定的需求开始,通过构造一系列可执行的中间版本来实施开发活动。第一个中间版本纳入一部分需求,下一个中间版本纳入更多的需求,依此类推,直到系统完成。每个中间版本都要执行必要的过程、活动和任务,如需求分析和体系结构设计需要执行一次,而详细设计、编码和测试、软件组装和验收测试在每个中间版本构造过程中都要执行,如图 1-5 所示。

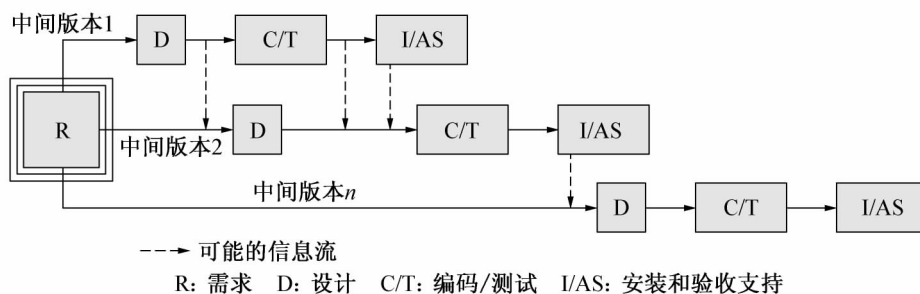


图 1-5 渐增模型示意

在开发每个中间版本时,开发过程中的活动和任务顺序地或部分平行地使用。当相继的中间版本在部分并行开发时,开发过程中的活动和任务可以在各中间版本间平行地采用。

在评价该模型时,需要考虑的风险因素是:

- (1) 需求未被很好地理解。
- (2) 突然提出一些功能。
- (3) 事先打算采用的技术迅速发生变化。
- (4) 需求迅速发生变化。
- (5) 长时期内仅有有限的资源保证(工作人员/资金)。

三、演化模型

演化模型(Evolutionary Model,又称进化模型)主要针对事先不能完整定义需求的软件开发。由于人们对软件需求的认识模糊,许多软件开发项目很难一次开发成功,返工再开发也就难以避免。因此,人们对需要开发的软件先给出基本需求,做第一次试验开发,其目标仅在于探索可行性和弄清需求,取得有效的反馈信息,以支持软件的最终设计和实现。通常,我们把第一次试验开发出来的软件称为原型(Prototype),这种开发模型可以减少由于需求不明给开发工作带来的风险。

演化模型也有多种形式,例如:

- (1) 丢弃型。原型开发后,已获得了更为清晰的需求反馈信息,原型无需保留而被丢弃,开发的原型仅以演示为目的,这往往用在软件用户界面的开发上。
- (2) 样品型。原型规模与最终产品相似,只是原型仅供研究之用。
- (3) 渐增式演化型。原型作为最终产品的一部分,它可满足用户的部分需求,经用户试用后提出精化系统、增强系统能力的需求,开发人员根据反馈信息,实施开发的迭代过程。如果一次迭代过程中有些需求还不能满足用户的要求,可在下一代中予以修正,整个过程实现后软件才可最终交付使用。

演化模型(如图 1-6 所示)也是通过构造系统的各个可执行的中间版本来开发系统的,但是它与渐增模型的区别是,承认需求不能被完全了解,且不能在初始时就确定。在该模型中,需求一部分被预先定义,然后在每个相继的中间版本中逐步完善。该模型中,每个中间版本在开发时,开发过程中的活动和任务顺序地或部分重叠平行地被采用。

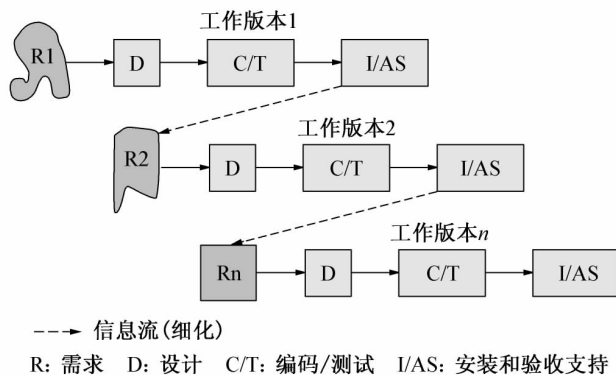


图 1-6 演化模型示意



对所有的中间版本,开发过程中的活动和任务通常按同一顺序被重复使用。维护过程和运作过程可以与开发过程平行地使用,获取过程、供应过程、支持过程和组织过程通常与开发过程平行地使用。

第四节 软件工程定义

人们对软件工程的定义和内涵有一个广泛讨论、研究和认识的过程,从而对软件工程有了各种各样的定义。例如:

(1) P. Wegner 和 B. Boehm 认为软件工程可定义为:科学知识在设计和构造计算机程序,以及开发、运作和维护这些程序所要求的有关文档编制中的实际应用。

(2) F. L. Bauer 认为软件工程的定义是:为了经济地获得软件,这个软件是可靠的并且能在实在的计算机上工作所需要的健全的工程原理(方法)的确立和使用。

(3) 1983 年 IEEE(国际电气与电子工程师协会)的软件工程术语汇编中,将软件工程定义为:对软件开发、运作、维护、退役的系统研究方法。1990 年新版的 IEEE 软件工程术语汇编又将定义更改为:对软件开发、运作、维护的系统化的、有纪律的、可量化的方法之应用,即是对软件的工程化应用。

(4) 《软件工程术语》(GB/T11457—1995)则把软件工程定义为:软件开发、运行、维护和引退的系统方法。

从以上对软件工程的定义中,我们可以看到对其内容的理解是逐步深入的。发展到今天,软件工程已是一门交叉性学科,它是解决软件问题的工程,对它的理解不应是静止的和孤立的。软件工程是应用计算机科学、数学及管理科学等原理,借鉴传统工程的原则、方法来创建软件,从而达到提高质量、降低成本的目的。其中,计算机科学和数学用于构造模型、分析算法;工程科学用于制定规范、明确风格、评估成本、确定权衡;管理科学用于进度、资源、质量、成本等的管理。

对于软件工程所包含的内容也不是一成不变的。随着人们对于软件系统的研制开发和生产的理解,也应该用发展的眼光来看待它,只有这样,才能客观、公正地反映出它的内涵和外延。

软件工程的目的是明确的,就是研制开发与生产出具有良好的软件质量和费用合算的产品。费用合算是指软件开发运行的整个开销能满足用户要求的程度,软件质量是指该软件能满足明确的和隐含的需求能力的有关特征和特性的总和。

软件工程的基础是一些指导性的原则,例如:

(1) 必须认识软件需求的变动性,并采取适当措施来保证结果产品能忠实地满足用户要求。在软件设计中,通常要考虑模块化、抽象与信息隐蔽、局部化、一致性等原则。

(2) 稳妥的设计方法可以大大地方便软件开发。为达到软件工程的目标,软件工具与环境对软件设计的支持来说颇为重要。

(3) 软件工程项目的质量与经济开销直接取决于对它所提供的支撑的质量与效用。

(4) 有效的软件工程只有在对软件过程进行有效管理的情况下才能实现。

一、软件工程学的内容

对软件应用的各个领域,认识问题与解决问题的技巧和手段大不相同。然而,可以使用一些与具体应用无关的技术来研究如何设计软件。这些技术就构成了软件工程学的基础。

从软件的生产管理技术来看,软件工程学大致可分为:软件的需求定义和分析技术;软件设计技术和设计审查技术;软件设计表现技术;软件测试技术;软件可靠性的理论及其评价方法;软件扩充和维护;软件成本估算等。

从内容上划分,软件工程学可分为理论、结构、方法、工具等部分。理论与结构是方法论与工具的基础及前提。在方法论与工具方面存在着这样的关系:为了找出行之有效的方法,就应该具备合适的工具;反之,为了制造、改进软件工具,也应先有合适的方法。

软件工程学主要研究如何以较少的代价获得高质量的软件。要达到这个目的,就必须研究软件开发方法和软件开发工具。

软件开发方法就是开发软件的规范化方法。一般来说,一个软件开发方法必须在以下 3 个方面作出规定:

- (1) 工作步骤,即明确软件系统的基本开发步骤及每一步的条件、目的、结果等。
- (2) 软件文件格式。
- (3) 评价标准,即确定对各种不同的软件开发方案的评价标准。

二、软件工程学的基本目标和原则

软件工程学研究的基本目标是:

(1) 一种定义良好的方法学,该方法学是面向包括计划、开发和维护等阶段的软件生存周期的。

(2) 一组确定的软件成分,它对软件生存周期的每一步记录软件文件资料,并且具有按步显示轨迹的能力。

(3) 一组可以预测的里程碑,在整个软件生存周期中,每隔一定时间可以对它们进行复审。

为了达到“以较少的投资获得易维护、可靠、高效率 and 易理解的软件产品”这个最终目的,各种软件工程技术所遵循的基本原则是:

(1) 分解。分解是分析解决复杂问题的重要手段,它的基本思想是将一个复杂的问题分成若干个较小的、相对独立的、较易解决的子问题,然后分别加以解决。将软件开发过程划分成几个阶段、结构化分析方法的分解、结构化设计方法和 Jackson 方法的模块化等都体现了分解的原则。

(2) 抽象和信息隐蔽。抽象和信息隐蔽也是解决复杂问题的重要手段,它的基本思想是在将复杂问题逐层分解时,将“怎么做”等大量细节隐蔽在下一层,从而使上一层突出“做什么”而得到简化。我们称上一层为下一层的抽象。

软件设计中常用的“模块化”和“局部性”方法就体现了这种抽象和信息隐蔽的原则。

(3) 一致性。一致性强调软件开发过程的标准化、统一化,包括软件文件格式的一致、工



作流程的一致等。研究软件方法的目的之一,就是使开发过程走上标准化的轨道。

(4) 确定性。确定性要求软件开发过程中用确定的形式将一些较含糊的概念表达出来。如用数据流程图来精确地表达用户需求,用易维护性、可靠性、易理解性、高效率等指标来具体度量软件的质量等。确定性也是保证软件质量的前提。

三、软件工程与一般工程的差异

软件工程是一门交叉性学科,又是工程学科家族中的新成员,在几十年的发展过程中,已经形成了其特有的形态。虽然软件产品在某些方面相似于其他工程中的有形产品,如桥梁、建筑物、机床、计算机等,但其间也确有一些重要的差别,不能简单地把一般工程方面的知识和技术直接应用到软件工程上来。

软件工程与一般工程的差别主要表现在以下 5 个方面:

(1) 软件是逻辑产品而不是实物产品,磁盘、集成电路块只是软件的载体,这就意味着费用集中在研制开发上而不是在生产上。当然,由于是逻辑产品,软件就不会用坏、磨损、老化,而且可以不断地改进、优化,其可靠性由逻辑性所确定。开发软件在许多方面更像进行数学证明。可是,软件产品的评价却主要决定于它们在问题求解中是否有用,而不是决定于抽象的正确性判定标准。换句话说,开发软件产品时主要使用的是工程标准,而不是数学标准。

(2) 由于软件是逻辑产品,使得它的功能只能依赖于硬件和软件的运行环境以及人们对它的操作才能得以体现。没有计算机及其相关硬件的支持,软件难以实现其实用价值。同样,没有软件支持的计算机硬件,也只是毫无使用价值的机器。软件与硬件的密切相关的程度是一般工程所没有的。

(3) 对软件产品的要求要比对一般的有形产品来得复杂。其一,软件产品要完成的多种多样的功能,用户难以清晰、准确地表达。仅此一项,软件系统的复杂性就可以比得上任何一个工程项目。其二,对软件产品的要求,如可靠性、易移植性、易使用性等是隐含的,也是难以表达的,而且也缺少度量的具体标准,与有形产品的质量检验的精度相距甚远。其三,软件设计不仅仅涉及技术复杂性,还涉及管理复杂性,即使在今天软件工程已有很大进展的情况下,领导一个庞大的项目组能像其他工程项目那样进行规模化生产也绝非易事。

(4) 在软件设计中存在的复杂性,使得软件具有以下四方面的特征:

① 功能的多样性。软件提供的功能比一般工程产品提供的功能更加多种多样,以致用户一时难以掌握。因此,使用软件产品,往往还会伴随着一个培训任务,而且软件产品的用户手册还不一定能全面描述其功能。

② 实现的多样性。对软件产品的要求不仅包括功能和性能要求,还必须包括在符合功能和性能要求的各种实现中做出选择,更有实现算法上的优化带来的不同,所以实现上的差异会带来使用上的差异。

③ 能见度低。看到软件进度要比看到有形产品的进度难得多,这里还将涉及如何使用文档表示的概念能见度,这又为软件管理的复杂性增加了难度。

④ 软件结构的合理性差。结构不合理使软件管理复杂性随着软件规模的增大而呈指数增长。换句话说,软件规模的增大所引起的管理复杂性成了软件设计的难题。

前两方面属于技术复杂性,后两方面属于管理复杂性。为了控制软件复杂性,人们进行