国家示范性软件学院系列教材

杜育根 编著

软件工程教程

IBM RUP方法实践

Software Engineering

The Practice of IBM RUP Method

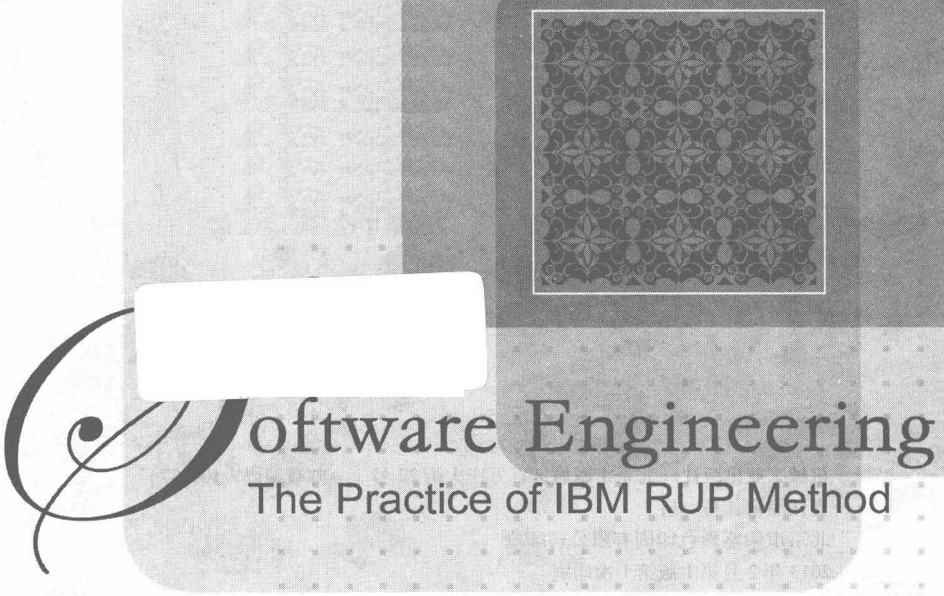
机械工业出版社
China Machine Press

国家示范性软件学院系列教材

杜育根 编著

软件工程教程

IBM RUP方法实践



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件工程教程：IBM RUP方法实践／杜育根编著．—北京：机械工业出版社，2013.2
（国家示范性软件学院系列教材）

ISBN 978-7-111-40681-5

Ⅰ．软… Ⅱ．杜… 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（2012）第293350号

本书采用IBM RUP框架，通过一个完整的一体化案例讲解RUP业务建模、需求、分析设计、实现、测试、部署等规程，主要内容包括：第1章介绍软件工程的基本知识和概念；第2章介绍面向对象和UML的基本知识和概念；第3～6章应用RUP思想，详细介绍业务建模、需求、分析与设计（分析建模、体系结构设计和详细设计）、实现、测试、部署相关的知识；附录中的“一体化案例介绍”给出本书第3～6章中所涉及的案例的背景和详细说明。

本书可以作为高等院校相关专业本科生和研究生的教材，同时也可以作为相关技术人员的参考用书。

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：马 超
北京市荣盛彩色印刷有限公司印刷
2013年2月第1版第1次印刷
185mm×260mm·15印张
标准书号：ISBN 978-7-111-40681-5
定 价：29.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前言

国外有一些比较好的关于 UML 和面向对象的教材，其中不乏好的案例，但这些教材不是从整个软件工程的开发角度来描述的，也较少应用 RUP 的框架阐述软件工程实践。国内目前有关 RUP 软件过程案例分析的教材也较少。

本书采用工业界通用的 IBM RUP 框架，通过一个完整的一体化案例讲解 RUP 业务建模、需求、分析设计、实现等规程，使读者认识和领会 RUP 软件开发过程的规范及方法。书中所有的一体化案例都取材于真实的软件项目，应用面向对象的开发方法，通过 UML 进行建模（建模使用的是 Rational Rose 工具），并参考 RUP 过程模型，采用 MVC 的体系结构进行设计实现。通过案例 + 理论捆绑式的讲解，能够使读者全面了解软件开发的完整过程，并掌握开发标准的软件开发文档和代码的方法及规范。

本书分为 6 章。第 1 章主要讲述软件工程的基本知识和概念。第 2 章介绍面向对象和 UML 的基本知识和概念。第 3 ~ 6 章应用 RUP 思想，详细介绍业务建模、需求、分析与设计（分析建模、体系结构设计和详细设计）、实现相关的知识，每一个操作环节都配有实践中的案例，帮助读者加深对理论的理解。这些案例是连贯的，相互之间是有联系的。附录中的“一体化案例介绍”给出本书第 3 ~ 6 章中所涉及的案例的背景和详细说明，便于读者对书中的案例有一个全面的了解。此外，各章节的相关术语、一些关键知识点标注了英文注解，便于开展双语教学的学校采用。

本书结合和吸收了作者在一体化案例教学、面向工业的软件工程专业课程教学实训体系研究与实践等的成果，在软件工程理论知识上有一定深度，并配套有较强操作性的一个连贯案例做说明，适用于软件工程的相关课程教学和实验。在阅读本书时，读者需要有 UML 及面向对象分析与设计的先导知识，此外案例的实现应用了 J2EE 框架技术，还要求读者具有一定的 J2EE 编程基础。

本书提供 PPT、案例建模技术文档及模板等相关内容，有需要的教师可到华章网站（www. hzbook. com）下载。另外，本书配套的一体化教学系统网站为 www. sei-ecnu. org。

最后，感谢 IBM 大学合作项目在书籍出版方面的资助。另外，也要感谢已毕业的学生李诗琦和喻桃阳，他们在案例文档的整理上给了我很大帮助。

编　者

2012 年 7 月于上海丽娃河畔

目录

第 1 章

■ 软件工程导论

导读

本章主要介绍软件工程的基本概念和基本知识，包括软件的概念、特点，软件危机的缘由，软件工程的诞生和基本内容，软件开发的过程模型，传统软件过程规范。本章还重点介绍 Rational 统一过程的概念和方法。

1.1 软件和软件工程史

本节介绍软件和软件工程的发展历史。

1.1.1 软件的产生

软件的历史可以追溯到 20 世纪 50 年代，伴随着第一台电子计算机（The Electronic Numerical Integrator And Computer，ENIAC，1946 年 2 月 14 日）的问世，软件也诞生了。然而第一个尝试写软件的人是 Ada（Augusta Ada Lovelace），在 19 世纪 60 年代她就尝试为 Babbage（Charles Babbage）的机械式计算机写软件，尽管他们的努力失败了。

在 1945 ~ 1949 年的计算机系统发展的初期，计算机硬件几乎每一两年更新一次，且硬件通常用来执行一个单一的程序，而这个程序又是为一个特定的目的而编制的。每当硬件更新，软件人员不得不重写所有的程序以运行在这些新机器上。那时编程人员通过登记到“机房”或者由机房的工作人员进行编程工作。而程序是打孔的卡，通过读卡机输入并把运行结果输出到打印机上。当时，软件业没有时间进度管理的概念，要预测一个项目的竣工日期几乎是不可能的。

大多数软件是由使用该软件的机构或个人研制的，1949 年后，出现了独立于卖主的早期专业软件服务公司，他们为个人客户开发定制解决方案。

先是由美国政府，后来是由几家美国大公司认购的几个大软件项目为第一批独立的美国软件公司提供了重要的学习机会，并使美国在软件业中成了早期的主角。开发于 1949 ~ 1962 年间的 SAGE 防空项目系统，是第一个极大的计算机项目，总开支最终达到了 80 亿美

元。1959 年，兰德公司（Rand）建立了一个独立的公司——系统开发公司（SDC），以进一步开发这个据估计需要 100 万行代码的软件。当时美国程序员的数目大约为 1200 名，有 700 人为 SAGE 项目工作。而开发于 1954～1964 年的 SABRE 机票预订系统，由美国航空公司要求 IBM（国际商业机器公司）开发，是第一个工业资助的软件项目，这是个雇用了大约 200 名软件工程师、耗资 3000 万美元的项目。该系统于 1964 年完成，并逐渐发展成了一个拥有 3 万多家旅行社、300 万在线客户的网络。SAGE 和 SABRE 系统都成了“程序员的大学”。此后，许多曾参与开发的程序员散布全美国，他们用在这些大项目上学到的知识创立了自己的公司。这些项目的实施奠定了美国在软件业的领先地位。

20 世纪中叶，当时的计算机十分昂贵，仅仅应用于少数的几个特殊领域，如国防、科研等。计算机软件则基本由计算机硬件厂商开发并打包在其硬件产品内免费赠送，或者由计算机用户自行编制应用软件。虽然大型计算机生产商为它们的大客户承接大的软件项目，但它们没有足够的资源为中等规模的客户开发软件。第一批编程企业“冲”进去填满的正是这个市场真空。1955 年，被认为是世界上第一家独立于卖主的软件（编程）服务公司——计算机惯用法公司（CUC）成立，由两位前 IBM 同事用 4 万美元创业资金创立，他们开始为不止一个平台提供软件服务，此时“software”这个术语尚未被发明（它首次被使用是在 1959 年）。“独立软件开发商”（Independent Software Vendors，ISV），特指专门从事软件的开发、生产、销售和服务的企业。

时至 20 世纪 60 年代，随着 COBOL（COmmon Business- Oriented Language）和 Fortran（Formula Translator）等高级编程语言的面世，计算机编程变得更为容易。一些新兴职业如程序员、分析员和计算机系统专家等应运而生。1964 年，硬件生产商 RCA 公司找到于 1959 年由 7 名程序员创立的 ADR 公司，要他们开发一个可以在一个程序里形象地代表设备的逻辑流程的流程图程序 Autoflow。最终，由于 RCA 公司对这个程序没有表示一点兴趣，ADR 公司试着通过直接向 RCA 501 计算机的 100 个用户发放许可证来收回大约 1 万美元的最初投资，尽管只有两个用户以其销售价 2400 美元购买了这个程序。ADR 公司改变策略，为 IBM 1401 计算机重写了程序，后来又为 IBM/360 系统重写了程序，获得了很大成功，在几年里数千台 IBM 计算机使用了 ADR 软件，使得这个软件成了第一个真正的软件产品，不但一次又一次销售给许多客户，还出现了一家围绕软件产品的开发和营销而组织的公司。

与此同时，计算机应用的范围和复杂度不断增加，这使计算机硬件生产厂商免费提供软件的成本加大，软件编制的任务也不断加重。终于，IBM 公司在 1969 年宣布停止发送免费随机软件，并从 1970 年 1 月 1 日开始分别为硬件和软件定价。这个具有里程碑意义的决定加速了软件产业的蓬勃发展。从此，一大批独立于计算机生产厂商的专门从事软件开发、生产、销售和服务的独立软件开发商崛起。其中一家生产 Mark Ⅳ软件的 Informatics 公司成为第一个最大受益者。IBM 公司免费软件的竞争不存在了，Mark Ⅳ软件在两年里一共出售了近 500 套，软件史上第一个收入超过 1 亿美元的产品诞生。其中，我们所熟知的主要软件企业包括：创建于 1972 年的 SAP 公司，目前是全球第三大独立软件开发商；创建于 1975 年的微软公司，目前是世界个人和商用计算机软件行业的领袖；创建于 1977 年的 Oracle（甲骨文）公司，目前是世界第二大独立软件开发商。

1.1.2 软件危机

20世纪60年代中期~20世纪70年代中期是计算机系统发展的第二个时期，在这一时期软件开始作为一种产品被广泛使用，出现了“软件作坊”专职根据用户的需要编写软件。早期的软件开发没有什么系统的方法可以遵循，软件设计是在某个人的头脑中完成的一个隐藏的过程。而且，除了源代码外往往没有软件说明书等文档，软件带有强烈的个人色彩。但软件的数量急剧膨胀，软件需求日趋复杂，维护的难度越来越大，开发成本高得惊人，而失败的软件开发项目却屡见不鲜。“软件危机”就这样开始了！

最为突出的例子是美国IBM公司于1963~1966年开发的IBM/360系列机的操作系统。该软件系统花了大约5000人一年的工作量，最多时有1000人投入开发工作，写出近100万行的源程序。尽管投入了这么多的人力和物力，得到的结果却极其糟糕。据统计，这个操作系统每次发行的新版本都是从前一版本中找出1000个程序错误而修正的结果。可想而知，这样的软件质量糟糕到了什么地步。难怪该项目的负责人F·D·希罗克斯在总结该项目时无比沉痛地说：“……正像一只逃亡的野兽落到泥潭中做垂死挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难，……程序设计工作正像这样一个泥潭……一批批程序员被迫在泥潭中拼命挣扎，……谁也没有料到问题竟会陷入这样的困境……”IBM/360操作系统的历史教训已成为软件开发项目中的典型事例被记入历史史册。

另一个例子发生在1963年，美国研制的一枚发射火星探测器的火箭，由于程序员将程序中的一个“,”误写为“.”，导致火箭升空爆炸，造成高达数亿美元的损失。

“软件危机”使得人们开始对软件及其特性进行更深一步的研究，人们改变了早期对软件的不正确看法。早期那些被认为是优秀的程序常常很难被别人看懂，通篇充满了程序技巧。现在人们普遍认为优秀的程序除了功能正确、性能优良之外，还应该容易看懂、使用、修改和扩充。

1968年北大西洋公约组织（NATO）的计算机科学家在联邦德国召开的国际学术会议上第一次提出了“**软件危机**”（Software Crisis）这个名词。概括来说，软件危机包含两方面问题：一是如何开发软件，以满足不断增长、日趋复杂的需求，二是如何维护数量不断膨胀的软件产品。

人们寻找产生危机的内在原因，发现其原因可归纳为两方面，一方面是软件生产本身存在着复杂性，另一方面却是与软件开发所使用的方法和技术有关。

1968年秋季，NATO的科技委员会召集了近50名一流的编程人员、计算机科学家和工业界巨头，讨论和制定摆脱“软件危机”的对策。在那次会议上第一次提出了**软件工程**（Software Engineering）这个概念。

软件工程是为克服软件危机而提出的一种概念，并在实践中不断地探索它的原理、技术和方法。在此过程中，人们研究和借鉴了工程学的某些原理和方法，并形成了一门新的学科——软件工程学，但可惜的是，时至今日，人们并没有完全克服软件危机。

软件工程是一门研究如何用系统化、规范化、数量化等工程原则和方法去进行软件的开发和维护的学科。软件工程包括两方面内容：软件开发技术和软件项目管理。软件开发技术包括软件开发方法学、软件工具和软件工程环境。软件项目管理包括软件度量、项目估算、进度控制、人员组织、配置管理和项目计划等。

1.1.3 传统软件工程

为迎接软件危机的挑战，人们进行了不懈的努力。这些努力大致上是沿着两个方向同时进行的（见表 1-1）。

表 1-1 软件工程的两个方向

内容	软件开发技术	软件开发方法学	基于“瀑布”模型的结构化生命周期方法
			基于动态需求的快速原型法
			基于结构的面向对象的软件开发方法
		软件工具	用来开发软件的软件
		软件工程环境	支持软件开发的环境、软件工具及其相互间关系的总和
	软件项目管理	软件管理	人力管理、进度安排、质量保证、资源管理
		软件工程经济学	以经济学的观点研究开发过程中的经济效益，成本估算、效益分析的方法和技术

软件工程发展的第一个方向，侧重于对软件开发过程中分析、设计的方法的研究。这方面的重要成果就是在 20 世纪 70 年代风靡一时的结构化开发方法，即 PO（面向过程的开发或结构化方法）以及结构化的分析、设计和相应的测试方法。

第二个方向是从管理的角度，希望实现软件开发过程的工程化。这方面最为著名的成果就是提出了大家都很熟悉的“瀑布式”生命周期模型。它是在 20 世纪 60 年代末“软件危机”后出现的第一个生命周期模型，如下所示：

分析→设计→编码→测试→维护

后来，又有人针对该模型的不足，提出了快速原型法、螺旋模型、喷泉模型等对“瀑布式”生命周期模型进行补充。现在，它们在软件开发的实践中被广泛采用。

这方面的努力，还使人们认识到了文档的标准以及开发者之间、开发者与用户之间的交流方式的重要性。一些重要文档格式的标准被确定下来，包括变量、符号的命名规则及源代码的规范。

软件工程的目标是研制、开发并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达到用户要求的程度，软件质量是指该软件能满足明确的和隐含的需求能力有关特征和特性的总和。软件质量可用 6 个特性来进行评价，即功能性、可靠性、易使用性、效率、维护性和易移植性。

1.1.4 现代软件工程

软件不是纯物化的东西，其中包含着人的因素，于是就有很多变动的东西，不可能像理想的物质生产过程，基于物理学等的原理来做。早期的软件开发仅考虑人的因素，传统的软件工程强调物性的规律，然而现代软件工程最根本的就是重视人与物的关系，就是人和机器（工具、自动化）在不同层次的不断循环发展的关系。

面向对象的分析、设计方法（OOA 和 OOD）的出现使传统的开发方法发生了翻天覆地的变化。随之而来的是面向对象建模语言（以 UML 为代表）、软件复用、基于构件的软件开发等新的方法和领域。

与之相应的是从企业管理的角度提出的软件过程管理，即关注软件生命周期中所实施的

一系列活动并通过过程度量、过程评价和过程改进等涉及对所建立的软件过程及其实例进行不断优化的活动使得软件过程循环往复、螺旋上升地发展。其中最著名的软件过程成熟度模型是美国卡内基－梅隆大学软件工程研究所（SEI）建立的 CMM（Capability Maturity Model），即能力成熟度模型。此模型在建立和发展之初，主要目的是为大型软件项目的招投标活动提供一种全面而客观的评审依据，而发展到后来，又同时被应用于许多软件机构内部的过程改进活动中。

1.1.5 软件工程与计算机科学

软件的开发到底是一门科学还是一个工程，这是一个争论了很久的问题。在学术界，人们争论的焦点主要集中在软件工程与计算机科学之间的关系上，即二者是存在一定交集的两个独立部分，还是前者是后者的一个子集。实际上，软件开发兼有两者的特点，但这并不意味着它们可以被互相混淆。IEEE-CS/ACM 在 Computing Curricula 2005（CC2005）中对整个计算领域进行了不同学科的划分，清楚地表明软件工程与计算机科学是完全独立的两个学科，但是二者存在着大量的交叉，两者的差异对比见表 1-2。

表 1-2 软件工程与计算机科学的差异对比

	软件工程	计算机科学
目标	在时间、资源、人员这 3 个主要限制条件下构建满足用户需求的软件系统	探索正确的计算和建模方法，从而改进计算方法本身
产品	软件（如办公套件和编译器）	算法（如希尔排序法）和抽象的问题（如哲学家进餐问题）
关注点	软件工程关注如何为用户实现价值	软件理论关注的是软件本身运行的原理，如时间复杂度、空间复杂度和算法的正确性
变化程度	随着技术和用户需求的不断变化，软件开发人员必须时刻调整自己的开发以适应当前的需求，同时软件工程本身也处于不断的发展过程中	对于某种特定问题的正确解决方法将永远不会改变
需要的其他知识	相关领域的知识	数学

1.2 软件和软件工程

下面详细介绍软件和软件工程的相关内容。

1.2.1 软件

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据结构及其相关文档的完整集合。这 3 个方面的内容如下。

1）程序：在运行时，能提供所希望的功能和性能的指令集。

2）数据结构：使程序能够正确运行的数据结构。

3）文档：描述程序研制过程、方法及使用的图文材料。

1. 软件特点

软件产品具有如下一些特性。

1）**抽象**（Abstract）而非**具体**（Concrete）：软件是一种逻辑实体，而不是具体的物理

实体，因而它具有抽象性。

2）**开发**（Develop）而非**制造**（Manufacture）：软件是通过人们的智力活动，把知识与技术转化成信息的一种产品，是在研制、开发中被创造出来的。

3）**退化**（Deterioration）而非**磨损**（Wear Out）：在软件的运行和使用期间，没有硬件那样的机械磨损、老化问题，但随着时间推移，由于软件不断变更，软件的失效率会不断上升，导致软件退化。

4）**定制**（Custom Built）而非**基于构件**（Component-based）：硬件制造越来越标准化，通过标准件构造完成，而软件的开发至今未完全脱离定制开发的方式。软件的开发费用在软、硬件费用中所占比例越来越高。

5）**不可见**（Invisible）：不像硬件，软件本身及其开发进度是不能立即看得到的。

6）**复杂**（Complex）：不像硬件制品，很难计算软件产品每分钱花在哪里。

7）**易改变**（Flexible）：相比物质的系统，软件更容易改变。

8）**易复制**（Easy to Copy）：与开发相比，复制非常容易，成本非常低。

2. 软件分类

软件按其功能可分为**系统软件**（System Software）、**支持软件**（Support Software）和**应用软件**（Application Software）。

（1）系统软件

系统软件是为特定的计算机系统或一簇计算机系统所开发的软件，用以管理计算机系统资源，促进计算机系统及有关程序的运行和维护，包括操作系统、系统实用程序、系统扩充程序等。

（2）支持软件

支持软件是指所有用于帮助和支持软件开发的软件，包括软件开发工具、软件测评工具、界面工具、转换工具、软件管理工具、语言处理程序、数据库管理系统和网络支持软件等。

（3）应用软件

应用软件是指为使用一个计算机系统以得到某种功能而专门开发的软件，包括科学和工程计算软件、文字处理软件、数据处理软件、图形软件、图像处理软件、应用数据库软件、事务管理软件、辅助类软件、控制类软件、智能软件、仿真软件、网络应用软件、安全与保密软件、社会公益服务软件和游戏软件等。

1.2.2 软件危机的表现

软件危机是一种现象，是指由于软件复杂程度愈来愈高，在计算机软件开发和维护时所遇到的一系列问题，具体表现在：

- 软件开发成本高，成本难以控制。
- 研制周期长，软件开发进度难以控制。
- 正确性难以保证，软件质量差，从而可靠性难以保证。
- 软件维护困难，维护人员和维护费用不断增长。
- 软件发展跟不上硬件的发展和用户的要求。

软件危机主要表现在以下方面。

（1）对软件开发成本和进度的估计常常很不准确

实际成本比估计成本高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了开发组织的信誉。为赶进度和节约成本所采取的权宜之计往往又降低了软件产品的质量，从而不可避免地引起用户的不满。

（2）用户对“已完成的”软件系统不满意的现象经常发生

软件开发人员常常在对用户需求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就匆忙着手编写程序。软件开发人员和用户之间的交流往往很不充分，“闭门造车”必然导致最终产品不符合用户实际需要。

（3）软件产品的质量常常靠不住

由于软件可靠性和质量保证的确切定量概念刚刚出现，软件质量保证技术（审查、复审和测试）还没有完全应用到软件开发的全过程中，所以导致软件产品时常发生质量问题。

（4）软件常常是不可维护的

程序中的错误很难改正，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需求在原有程序中增加新的功能。

（5）软件通常没有适当的文档资料

软件不仅是程序，还应该有一整套文档资料。这些文档资料是在软件开发过程中产生出来的，而且应该是“最新的”（与代码完全一致）。缺乏文档必然会给软件的开发和维护带来许多困难和问题。

（6）软件成本在计算机系统总成本中所占比例逐年上升

随着微电子技术的进步和生产自动化程度的提高，硬件成本逐年下降，然而软件开发需要大量的人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大和增加而逐年上升。

1.2.3　软件工程

软件工程是一门将理论和知识应用于实践的学科，它借鉴了传统工程的原则和方法，以求高效地开发高质量软件。除此之外，软件工程还综合应用了计算机科学、数学、工程科学和管理科学。计算机科学和数学用于构造模型与算法，工程科学用于制定规范、分析与设计、评估成本及确定平衡，管理科学用于计划、资源、质量和成本的管理。

1. 软件工程定义

软件工程的定义很多，其中 IEEE 组织的定义（IEEE Standard 610.12—1990）比较经典。IEEE 对软件工程的定义如下：

1）将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中。

2）对 1）中所述方法的研究。

前文中提到，软件工程这一概念是在 1968 年 NATO 会议上针对“软件危机”而提出的。自这一概念提出以来，围绕软件项目，开展了有关开发模型、方法及支持工具的研究。其主要成果有：提出了“瀑布”模型，开发了结构化程序设计语言（如 PASCAL、Ada 语言），设计了结构化方法等。同时，围绕项目管理提出了费用估算、文档复审等方法和工具。20 世纪 60 年代末～20 世纪 80 年代初，软件工程发展的主要特征是，前期着重研究系统实现技术，后期开始强调开发管理和软件质量。

2. 软件工程要素

20 世纪 70 年代初，自“软件工程”这一概念提出以来，围绕软件过程及软件复用，主要开展了有关软件生产技术和软件生产管理的研究与实践。其主要成果有：提出了应用广泛的面向对象语言以及相关的面向对象方法，大力开展了计算机辅助软件工程的研究与实践。尤其是近几年来，针对软件复用及软件生产，软件构件技术以及软件质量控制技术、质量保证技术得到了广泛的应用。软件工程涉及的要素如图 1-1 所示。

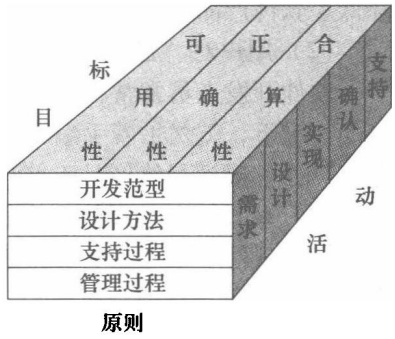


图 1-1 软件工程要素

通过这一框架可以看出：软件工程涉及了软件工程的目标、原则和活动。

软件工程的主要目标是：生产具有正确性、可用性以及合算性的产品。正确性是指软件产品达到预期功能的程度。可用性是指软件基本结构、实现及文档为用户可用的程度。合算性是指软件开发、运行的整个开销满足用户要求的程度。这些目标的实现不论在理论上还是在实践中均存在很多问题有待解决，它们形成了对过程、过程模型及工程方法选取的约束。

软件工程活动是“开发一个最终满足需求且达到工程目标的软件产品所需要的步骤”，主要包括需求、设计、实现、确认及支持等活动。需求活动包括问题分析和需求分析。设计活动一般包括概要设计和详细设计。**概要设计**建立整个软件体系结构，包括子系统、模块以及相关层次的说明、每一模块接口定义。**详细设计**产生程序员可用的模块说明，包括每一模块中数据结构说明及加工描述。实现活动把设计模型转换为可执行的程序代码。确认活动贯穿于整个开发过程，实现完成后的确认，保证最终产品满足用户的要求。支持活动包括修改和完善。伴随以上活动，还有管理过程、支持过程、培训过程等。

3. 软件工程——一种层次化技术

软件工程是一种层次化的技术（Layered Technology），如图 1-2 所示，分 4 个层次，由底向上分别是**质量关注**（Quality Focus）、**过程模型**（Process Model）、**方法**（Method）、**工具**（Tool）。

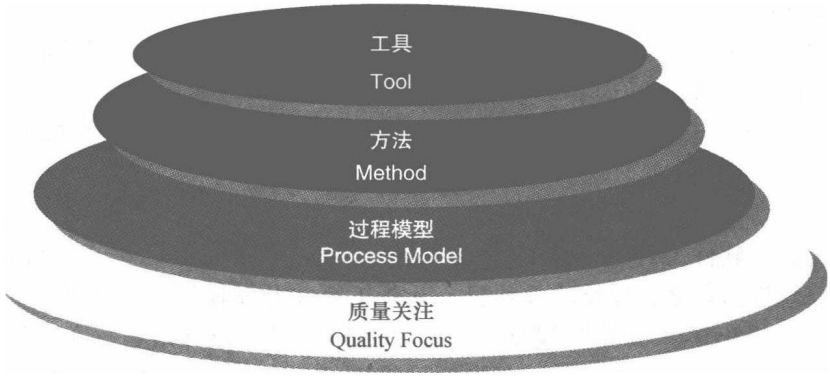


图 1-2 软件工程层次图

（1）质量关注

任何工程方法（包括软件工程）必须以有组织的质量保证为基础。全面的质量管理和类似的理念促使过程不断地改进，正是这种改进导致不断地出现更加成熟的软件工程方法。支持软件工程的根基就在于对质量的关注。

（2）过程模型

软件工程的基层是过程层。软件工程过程是将技术层结合在一起的凝聚力，使得计算机软件能够被合理和及时地开发出来。过程定义了一组关键过程区域（Key Process Area，KPA）的框架，这对于软件工程技术的有效应用是必需的。关键过程区域构成了软件项目的管理控制的基础，并且确立了上、下各区域之间的关系，其中规定了技术方法的采用、工程产品（模型、文档、数据、报告和表格等）的产生、里程碑的建立、质量的保证及变化的适当管理。

（3）方法

软件工程的方法层提供了开发软件在技术上需要“如何做”。方法涵盖了一系列的任务：需求分析、设计、编程、测试和维护。软件工程方法依赖于一组基本原则，这些原则控制了每一个技术区域，且包含建模活动和其他描述技术。

（4）工具

软件工程的工具层对过程和方法提供了自动或半自动的支持。当这些工具被集成起来使得一个工具产生的信息可被另外一个工具使用时，一个支持软件开发的系统就建立了，称为**计算机辅助软件工程**（CASE）。CASE 集成了软件、硬件和一个软件工程数据库（一个仓库，其中包含了关于分析、设计、编程和测试的重要信息），从而形成了一个软件工程环境，它类似于用于硬件设计的 CAD/CAE（计算机辅助设计/工程）软件。

1.2.4　软件过程及通用过程框架

软件有一个孕育、诞生、成长、成熟、衰亡的生存过程，这个过程即为计算机软件的生命周期。

正如前面所说，软件工程的基础是软件过程。软件过程模型（也称软件生命周期模型，软件工程范型）是跨越整个生存期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。简单地说，是软件产品或软件系统从设计、投入使用到被淘汰的全过程。所谓模型就是一种开发策略，这种策略针对软件工程的各个阶段提供了一套范型，使工程的进展达到预期的目的。

软件过程的**通用过程框架**（Generic Process Framework）是指不考虑软件系统规模和复杂性而适用于所有软件项目的活动，如图 1-3 所示，它包括两类活动，一类是框架活动（Framework Activity），还有一类是保护性活动（Umbrella Activity）。

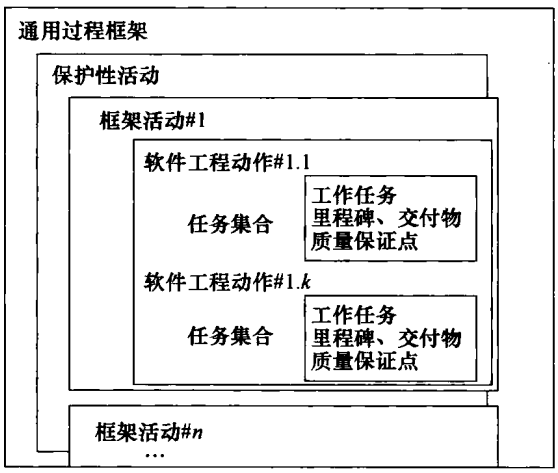


图 1-3　通用过程框架