第 1 章　软 件 工 程

导读：

提到工程时，大家都会不约而同地想到土木工程等传统工业，为什么软件也和工程挂钩了呢？

首先看看工程的基本定义，工程是科学和数学的某种应用，通过这一应用，自然界的物质和能源的特性能够通过各种结构、机器、产品、系统和过程，是以最短的时间和精力做出高效、可靠且对人类有用的东西。软件行业的高速发展，规模越来越大，复杂程度不断增加，使软件的开发和维护遇到一系列的严重问题，导致 20 世纪 60 年代末期“软件危机”的出现，人们便想如何避免软件开发过程中遇到的问题，于是软件工程的概念被提出来。软件工程就是试图用工程、科学和数学的原理与方法研发、维护计算机软件的有关技术及管理方法。

自从软件工程概念提出以来，经过三十多年的研究与实践，虽然“软件危机”没有得到彻底解决，但人们借助工程的思想，在软件开发方法和技术方面已经有了很大的进步。把需求计划、可行性研究、工程审核、质量监督等工程化的概念引入到软件生产当中，以期达到工程项目的三个基本要素：进度、成本和质量的目标。

随着互联网的迅速发展和普遍应用，对软件工程产生广泛而深远的影响，促进软件工程不断创新、不断变革和不断发展，以适应新形势下的软件开发、运行和维护的需求，最显著的两个例子就是开源软件运动和软件即服务。

根据调查，中国的现状几乎和美国十多年前的情况一样，软件开发过程没有明确规定，文档不完整，也不规范，软件项目的成功往往归功于软件开发组的一些杰出个人或小组的努力。这种依赖于个别人员的成功并不能为全组织的软件生产效率和质量的提高奠定有效的基础，只有通过建立全组织的过程改善，采用严格的软件工程方法和管理，并且坚持不懈地付诸实践，才能取得全组织的软件过程能力的不断提高。

软件工程是计算机专业的一门重要的专业基础课，它对于培养学生的软件素质，提高学生的软件开发能力与软件项目管理能力具有重要的意义。

软件工程这门科目，本来是纯理论的知识学习，大家学习之后可能只掌握一些理论的知识，对于实际的软件项目怎么实施可能还不清楚。本书以理论为指导，结合一个实际的软件项目，详细地介绍软件项目是如何设计、实施、部署和维护的，同时根据软件的基本流程，也就是 ERCM，详细介绍在每个阶段需要从上个阶段得到什么样的输入，然后有什么样的成果输出，而本阶段的输出将成为下一个阶段的输入，环环相扣。

下面我们就一起来揭开软件工程的神秘面纱吧！

今天，计算机应用充斥在人们生活、工作、娱乐的各个角落，如网上购物、网络游戏、微博、视频会议、网络视频、电子邮件、财务软件、ATM 自动存取款、银行网络系统等。这些飞速发展的计算机应用不仅改变了人们的生活方式，提高了人们的生活质量，同时也改变了人们的思维，让人们不禁时常发出感慨，计算机还有这样的应用。

五花八门的计算机应用背后都有着不同的复杂性和研发周期，其研发过程中经历的波折、困难也不尽相同，所以需要有科学的方法来进行跟踪管理，以保证可以开发出具有有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性、可互操作性、可兼容性，且满足用户需求的产品。软件工程正是这样一种包含系统学的思想、工程管理的方法和传统工业成功经验的学科。

1.1　软件工程概述

1.1.1　软件工程的定义

软件工程（Software Engineering，SE）是一门研究用工程化方法构建和维护有效、实用和高质量的软件的学科。它涉及程序设计语言、数据库、软件开发工具、系统平台、标准、设计模式等方面。软件工程包括三个要素，即方法、工具和过程。方法是完成软件工程项目的技术手段；工具支持软件的开发、管理、文档生成；过程是支持软件开发的各个环节的控制、管理。

软件工程概念的提出最早可以追溯到 20 世纪 60 年代末的软件危机。所谓软件危机泛指在计算机软件开发和维护过程中遇到的所有严重问题。在软件开发的开始，由于软件本身简单、规模小，可以很迅速地发现开发和维护过程中的严重问题，并快速地解决问题。但是，随着计算机技术的发展，计算机应用领域和规模的不断扩大，软件复杂度的不断增加，涉及人员的增多，一方面引起了软件开发成本的提高，开发效率的降低；另一方面，使用老的方法已经无法快速地发现和解决问题，软件质量问题逐步显现和不可控，最终成为计算机发展过程中最大的“瓶颈”，并且引发了软件危机。软件危机的出现极大地阻碍了计算机应用技术的发展。

在软件开发和维护过程中，软件危机主要表现在质量、成本和效率方面，如：

- 软件需求的增长和无法快速实现。
- 软件开发成本和进度的不可控。
- 软件质量难以保证。
- 软件可维护性差。
- 软件开发效率的提高赶不上硬件的发展和应用需求的增长。

为了解决软件危机，1968 年召开了北大西洋公约组织会议（NATO 会议），会议上讨论了摆脱软件危机的办法。德国人 Fritz Bauer 认为：“软件工程是建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法”。在这次会议上，首次提出了软件工程（Software Engineering）的概念，这在软件技术发展史上是一件大事。1993 年，IEEE（Institute of Electrical＆Electronic Engineers，电气和电子工程师

学会）对软件工程给出了一个更加综合的定义：“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。”这些思想主要都是强调在软件开发过程中需要应用工程化的原则、科学和数学的原理与方法、维护计算机软件的技术及管理方法。此后，随着软件技术的发展，各种有关软件工程的技术、思想、方法和概念不断地被提出，软件工程逐步发展成为一门独立的科学，开辟了工程学的新兴领域——软件工程学。

从软件工程的发展过程可以看出，软件工程的核心思想是把软件的开发看做是生产一个工程产品，引进工程产品生产过程中的科学管理方法：需求计划、可行性研究、工程审核、全面质量管理等，以期达到对工程项目的三个基本要素：进度、成本和质量的可控。此外，针对软件项目不同于工程产品的特点，软件工程提出了一些特有的技术方法，其中具有代表性的有结构化方法、面向对象方法、软件开发模型构建、软件开发过程、软件生命周期等。

软件工程的使用，使得软件开发摆脱了小作坊似的模式，走上了工业化的开发模式。

1.1.2　软件工程的目标

软件工程的目标是在付出相对较低的开发成本、给定进度的前提下，按时开发和交付出具有有效性、可靠性、可理解，可维护性、可重用性、可兼容性、可适应性、可移植性、可追踪性、可互操作性，且满足用户最终需求的产品。

软件工程的理论和技术性研究的内容主要包括：软件开发技术和软件工程管理。

1．软件开发技术

软件开发技术包括：软件开发方法学、开发过程、开发工具和软件工程环境，其主体内容是软件开发方法学。软件开发方法学是根据不同的软件类型，按不同的方法和原则，对软件开发中应遵循的策略、原则、流程和必须生成的所有文档资料都做出规定，从而使软件的开发能够进入规范化和工程化的阶段，以克服早期的作坊式开发中的随意性和非规范性做法。

2．软件工程管理

软件工程管理包括：软件管理学、软件工程经济学、软件心理学等内容。

软件工程管理是软件工程化开发过程中的重要环节，它要求软件的开发必须按照预先制定的计划、进度和预算执行，以实现预期的经济效益和社会效益。统计数据表明，多数软件开发项目的失败，并不是由于软件开发技术方面的原因，而是由于不适当的管理造成的。目前国内很多软件公司过于追求软件的开发技术，忽略了软件工程管理的重要性，在开发过程中不是没有使用软件工程管理，就是没有正确地执行软件工程管理，因此提高人们对软件工程管理重要性的认识，从而保证在软件开发过程中能够严格按照软件工程管理来执行也是撰写本书的主旨之一。

软件管理学包括人员组织、进度安排、质量保证、配置管理、项目计划等。

软件工程经济学是研究软件开发中成本的估算、成本效益分析的方法和技术，用经济

学的基本原理来研究软件工程开发中的经济效益问题。

软件心理学是软件工程领域具有挑战性的一个全新的研究视角，它是从个体心理、人类行为、组织行为和企业文化等角度来研究软件管理和软件工程的。

1.1.3　软件工程的基本原则

软件工程的基本原则包括：

- 选择适宜的开发模型。该原则与系统设计有关。在系统设计中，需要根据软件项目的实际情况权衡各种相互制约、相互影响的因素，如软件需求、硬件需求等；权衡后采用最适宜的开发模型，对各种需求、变化进行控制，以保证软件产品最大化满足用户的要求。
- 采用合适的设计方法。在软件设计中，合适的设计方法有助于实现软件的模块化、抽象与信息隐蔽、局部化、一致性以及适应性，以达到软件工程的目标。
- 高质量的工程支持。在软件工程中，软件开发、测试、管理工具与环境对软件工程的支持颇为重要。对软件工程所提供支持的质量和效用直接决定了软件工程项目的质量与开销。
- 有效管理软件工程过程。有效地管理软件工程的各个过程，对可用资源的有效利用，软件产品的质量提高，软件组织的生产能力的提高等问题有直接的影响。因此，只有当软件过程得到有效管理时，才能实现有效的软件工程。

软件工程的基本原则的每项都是非常重要的，它要求既重视软件技术的应用，又重视软件工程的支持和管理，最重要的是要在实践中实施，这样才能高效地开发出高质量的软件。如果在实践中，忽视软件工程的支持和管理，任何好的开发方法和技术都不会起到所期望的作用；同理，如果在实践中忽视好的开发方法和技术，任何好的工程支持和管理，也是不能获得高效率和高质量的软件的。

1.1.4　软件工程的作用

自从软件工程概念提出以来，经过三十多年的研究与实践，虽然“软件危机”没有得到彻底解决，但在软件开发方法和技术方面已经有了很大的进步。尤其应该指出的是，自 20 世纪 80 年代中期，美国工业界和政府部门开始认识到，在软件开发中，最关键的问题是软件开发组织不能很好地定义和管理其软件过程，从而使一些好的开发方法和技术都起不到所期望的作用。也就是说，在没有很好定义和管理软件过程的软件开发中，开发组织不可能在好的软件方法和工具中获益。

根据调查，中国的现状几乎和美国十多年前的情况一样，软件开发过程没有明确规定，文档不完整，也不规范，软件项目的成功往往归功于软件开发组的一些杰出个人或小组的努力。这种依赖于个别人员的成功并不能为全组织的软件生产效率和质量的提高奠定有效的基础，只有通过建立全组织的过程改善，采用严格的软件工程方法和管理，并且坚持不懈地付诸实践，才能取得全组织的软件过程能力的不断提高。

这一事实告诉我们，只有坚持软件工程的 4 条基本原则，既重视软件技术的应用，又

重视软件工程的支持和管理，并在实践中贯彻实施，才能高效地开发出高质量的软件。

1.1.5　软件工程基本流程 ERCM

软件工程的基本流程，也就是 ERCM（Engineering Release Cycle Methodology），把软件工程划分成不同的阶段，并规定了在每个阶段里，不同角色的成员需要做些什么事情，需要达到什么样的目标。

在实际的软件工程过程中，一般把项目划分成如下的几个阶段。

- PRD 生成阶段。PRD（Product Requirements Document）是产品需求文档，它决定了产品需要做什么，要实现哪些功能，它对整个项目具有指导作用，是软件开发的基准。PRD 通常由 PM（产品经理）根据客户的实际需求设计完成。PRD 在生成阶段，EM（工程部经理）、DEV（开发工程师）就会参与进来，阅读 PRD，并且提交发现的问题。QA（Quality Assurance）工程师会在 PRD 生成之后，参与 PRD 的阅读，提交发现的问题；PM 会与 DEV 和 QA 对问题进行讨论，并根据讨论结果修改 PRD。在 PRD 审阅完毕后，PM、EM、DEV 和 QA 对产品需求的理解应该是一致的。
- SPEC 设计阶段。SPEC（Specification）是产品规格说明书，当 PRD 确定之后，EM 就要根据 PRD 设计 SPEC。在 SPEC 中，将根据 PRD 细化客户的每个需求，详细设计产品的每个功能、逻辑关系、产品界面风格等。当 SPEC 设计完之后，PM、DEV、QA 必须共同对 SPEC 进行审阅，从各自的角度检查 SPEC 是否有设计不合理的、遗漏的地方，并与 EM 共同讨论，并按照讨论结果进行修改。SPEC 设计完成后，DEV 就要开始根据 SPEC 设计开发文档，QA 开始进行 Test Plan 和 Test Case 的设计。
- Test Plan 设计阶段。测试计划是 QA 工程师完成的，当 SPEC 中的内容最终确定之后，QA 工程师就要开始制定测试计划。在这个阶段，开发工程师就要开始写代码。
- Test Case 设计阶段。测试用例同样也是 QA 工程师完成的，它也是基于 SPEC 设计的，和 Test Plan 几乎在同一个阶段完成。开发工程师在这个阶段，需要对自己写的代码进行单元测试。
- 产品代码 CC（Code Complete）阶段。在产品 SPEC 确定之后，开发工程师就开始写代码，到 CC 阶段，就需要完成所有代码设计，并且完成代码的单元测试。
- 产品 CF（Code Freeze）阶段。当代码完成之后（CC），测试工程师开始进入测试周期，一般经过两到三轮测试之后，产品中存在的问题基本上全部被发现，再也找不到比较严重的产品缺陷，而且开发工程师把所有找到的产品缺陷修复，就可以达到 CF 标准。在 CF 之后，一般情况下，不允许轻易地改动代码，即使需要改动，也必须经过一定的流程控制，以确保没有 Regression 问题出现。
- 产品 ER（Engineer Release）阶段。当 CF 之后，测试工程师需要经过一到两轮的验证测试，确定没有较严重的缺陷存在时，产品就可以对外发布，或与客户一起进行验收测试。

详细的 ERCM 流程图，如图 1-1 所示。

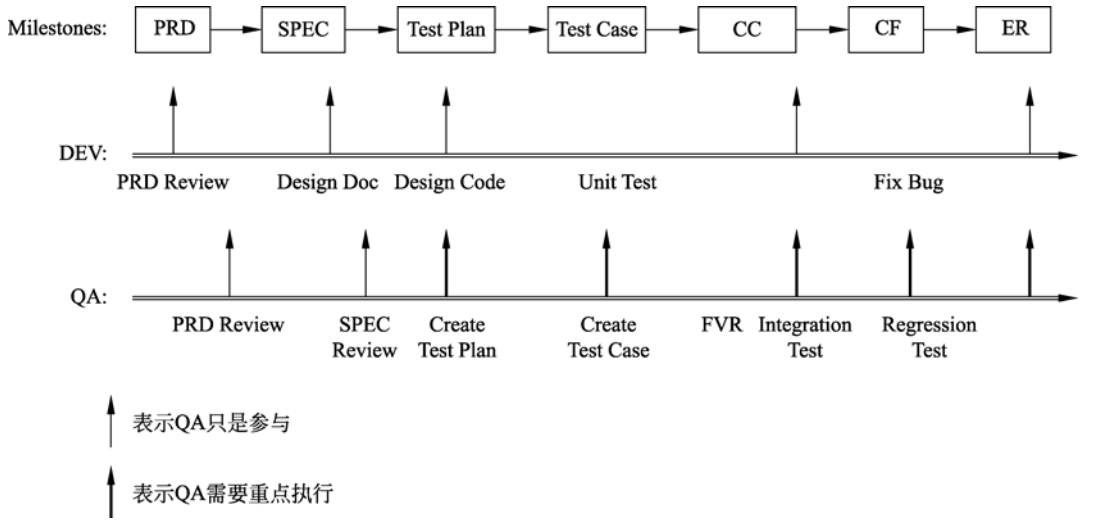

图 1-1　软件工程基本流程 ERCM

Milestone——里程碑

PRD——Product Requirements Document（产品需求文档）

SPEC——Specification（产品规格说明书）

CC——Code Complete（代码完成）

CF——Code Freeze（代码冻结）

ER——Engineering Release（工程交付）

PM——Product Manager（产品经理）

EM——Engineer Manager（工程部经理）

DEV——Developer（开发者）

QA——Quality Assurance（质量测试工程师）

FVR——Feature Validation Report（功能验证测试）

Integration Test——集成测试

Regression Test——衰退测试

1.2　现代软件工程

软件工程是为了解决软件危机而提出的，目的是规范软件开发和维护的行为，将类似于工业化的管理方法引入到软件工程中，从 1968 年第一次提出“软件工程”概念至今，经历了 40 多年的发展，获得了大量的研究成果和实践经验。但是，软件技术和方法随着计算机的高速发展，各种新技术、新模式、新方法和新工具层出不穷，令人眼花缭乱，软件工程不断受到各种各样的挑战。从早期的结构化方法到现在的面向对象方法、面向构件方法和面向服务方法等，从 CMM 这样的重型方法到极限编程（XP）的敏捷方法，软件工程方法也在不停地变化和发展，以适应软件领域出现的新的需求。尤其是互联网的迅速发展和

普遍应用，对软件工程产生了广泛而深远的影响，同时也使软件工程面临更大的挑战，促进软件工程不断创新、不断变革和不断发展，以适应新形势下的软件开发、运行和维护的需求。最显著的两个例子就是开源软件运动和软件即服务（Software as a Services，SaaS）。

1.2.1　开源软件运动

现在的开源软件可以说无处不在，其中最典型的代表是 LAMP，即 Linux+Apache+MySQL+PHP，它涵盖了操作系统、Web 服务器、数据库和编程语言等。开源软件运动促进了思想的开放、软件经验的共享等，也促进了软件过程的不断变革，对一些传统软件工程理念带来了冲击，对未来的软件工程实践产生了极其深远的影响。

开源软件的开发和传统软件的开发在组织、方式上发生了很大的变化。许多软件的开发是基于互联网的环境展开的，参与项目开发的团队成员分布在世界各地，人们往往采用同步开发和异步开发相结合的方式进行。

一方面，由于互联网的存在和软件开发协作的需求，来自世界各地的很多人可能同时完成同一个模块或组件的开发工作，同步开发是必要的。从开源软件开发过程来看，许多不同阶段的任务是同时进行的，而不像传统软件工程那样按严格规定的次序进行。另一方面，由于时区的差异，不能保证所有的人在同一时间工作，需要采用异步开发来协调这种关系。如果很好地利用异步开发模式，可以使项目在 24 小时每时每刻都有人在上面工作，可以大大提高项目进展速度。开源软件的开发，从本质上来讲，是以互联网为基础的支撑平台，一种由社会技术过程、开发条件和动态产生的各种关联事件组成的、复杂的网络开发。其特点表现在以下几方面。

- 日常管理成本最小化。版本控制需要强大灵活的支持，能让更多并行的开发人员同时在相互重叠的源代码文件上工作；需要为需求、设计、测试等各类文档制定标准模板，并在一个组织良好的 Web 站点及时发布。
- 开源软件的设计，努力提取其共性而形成参考体系结构，并使其易于移植。通常不会针对特定平台发行某个软件版本，而是将各种平台的共性抽象到一个发行版本中，通过配置工具来帮助构建系统。
- 人员组织。开源项目的核心小组成员及其责任分配是自发形成的，并不是硬性指派产生的。项目成员之间没有像企业中等级清楚且严厉的上下级关系，软件开发组织的管理方式和过去有很大的不同。
- 非正式交流（如邮件、论坛等）在开源软件开发的活动中发挥着积极的重要作用，协调项目成员间的任务及其关系。例如，其中一个基本原则是技术交流应当在公共论坛中进行。
- 多数商业软件公司在进入售后支持阶段时就可能面临悲惨的失败命运。而开源项目的用户支持工作却有良好成绩，因为大量的用户愿意提供关于开源产品的反馈信息，积极参与功能设计，用户的参与度比商业软件要高得多。
- 对于许多开源项目而言，开发人员并不刻意遵循特定的软件工程方法和过程。有些项目甚至没有特定的用户和发布截止期限，出现了许多新的概念，诸如“经常发布”、“简化设计”、“永远的 Beta”、“代码集体拥有”和“编码标准”等。

- 采取独特的、灵活的方式来解决标准、资源配置和进度安排等问题，如共享的 TO-DO（等待处理的任务）列表用于跟踪那些需要完成的任务，个人的 TO-DO 列表帮助开发人员保持进度，里程碑列表则基于用户和开发者的反馈设置了灵活的截止期限。

有关开源软件协同开发原理的经典论述，可以追溯到开源社区领袖之一的 Eric Redmond 写的《大教堂与集市》（The Cathedral and the Bazaar，1997，http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar）文章当中。作者形象地将传统的严格管理软件开发活动比喻为建造大教堂的行为，而将分布于世界各地、借助互联网的开源社区的协同开发活动看做是比较自由散漫的“集市”行为。作者根据亲身经历系统地论述了集市型开发的参考价值。开源软件开发积累了许多成功的经验，例如：

- 早发布、常发布、听取用户的建议。
- 把用户当作协作开发者和测试人员。
- 精妙的数据结构和笨拙的代码所构成的组合（软件）肯定好于笨拙的数据结构和精妙的代码。
- 最好的设计是最精简的设计，即再也没有什么内容可以去掉，而不是“再也没有什么内容可以添加了”。
- 好的程序员知道如何写代码，伟大的程序员知道重用或重构什么代码。

1.2.2　SaaS

SaaS（Software-as-a-Service）是通过 Internet 提供软件服务的模式，用户不用再购买软件、无须维护软件，而是直接使用基于互联网的软件，即享受软件供应商所提供的软件服务，这些服务可能是免费的，也可能需要付费，如许可费、月租费等。SaaS 如同煤气、电等服务一样，软件服务提供商负责软件系统的安装、管理和维护，而使用者只要支付相对低廉的服务费用即可使用该软件提供的服务。由 SaaS 延伸出各种类似的概念，如按需软件服务（On-demand Software）、应用服务提供商（Application Service Provider，ASP）和托管软件服务（Hosted Software Service）。对于许多中小企业来说，SaaS 是企业信息化的有效途径，可以免除一次性的 IT 投资，可以大大减少计算机机房建设、雇用 IT 人员、软硬件购买和维护等各项成本，而且所享受的服务成熟、稳定和可靠。

与软件产品模式不同，软件服务模式在产品发布程序、软件部署和维护等多个方面有较大差别。首先，软件服务模式的产品发布要复杂得多，涉及软件产品部署和实施的前期活动和后期活动。

- 前期活动：这些活动与软件需求分析、设计、编程和测试过程并行进行，在测试结束前完成，如软件产品的部署（Deployment）规划、部署设计、部署设计的验证等。
- 后期活动：这些活动发生在产品完成测试之后，如软件产品的部署（实施）、软件产品运行监控的设立，也可以看作从软件开发到软件维护的过渡阶段。

其次，软件服务模式的部署是一个必不可少的、关键的环节。软件部署涉及软件系统的技术要求、业务要求，要认真分析这些要求对系统部署的影响，从而决定系统运行环境的设计，包括业务分析、技术要求的确定、逻辑设计、部署的详细设计、设置优化等，最后根据部署的设计来实施。

再者，软件服务模式的软件维护需要考虑版本的网站或数据的迁移、多种升级方式的错开和验证，并要处理好与客户的关系，对于功能变化较大的新版本升级，一般要事先得到用户的许可或同意。

对于软件服务模式，当产品发布到运行环境（服务器）中，在用户开始使用之前，还要进一步验证。所以，对软件服务模式的产品发布中最后实施阶段，其时间性非常强，一般放在周末或晚上时间（9:00PM～6:00AM）。如果提供 7 × 24h 不间断的软件服务，就需要采用 DNS、服务器、目录等快速切换方式来实现无缝升级。

1.3　学习笔记

读书笔记　　*Name:*　　*Date:*

励志名句：*People With Passion Can Change The World—Steve Jobs*

充满激情的人可以改变世界。——乔布斯