第 2 章

Software Engineering: A Practitioner's Approach, Eighth Edition

软件工程

要点浏览

概念：软件工程包括过程、一系列方法（实践）和大量工具，专业人员借由这些来构建高质量的计算机软件。

人员：软件工程师应用软件工程过程。

重要性：软件工程之所以重要是因为它使得我们可以高效、高质量地构建复杂系统。它使杂乱无章的工作变得有序，但也允许计算机软件的创建者调整其工作方式，以更好地适应要求。

步骤：开发计算机软件就像开发任何成功的产品一样，需采用灵活、可适应的软件开发过程，完成可满足使用者要求的高质量的软件产品。这就是软件工程化方法。

工作产品：从软件工程师的角度来看，工作产品是计算机软件，包括程序、内容（数据）和其他工作产品；而从用户的角度来看，工作产品是可以改善生活和工作质量的最终信息。

质量保证措施：阅读本书的后面部分，选择切合你所构建的软件特点的思想，并在实际工作中加以应用。

关键概念
框架活动
通用原则
原则
问题解决
SafeHome
软件工程
　定义
　层次
　实践
软件神话
软件过程
普适性活动

要构建能够适应 21 世纪挑战的软件产品，我们必须认识到以下几个简单的事实：

- 软件实际上已经深入到我们生活的各个方面，其结果是，对软件应用所提供的特性和功能感兴趣的人显著增多。[⊖]因此，在确定软件方案之前，需要共同努力来理解问题。
- 年复一年，个人、企业以及政府的信息技术需求日臻复杂。过去一个人可以构建的计算机程序，现在需要由一个庞大的团队共同实现。曾经在可预测、独立的计算环境中实现的复杂软件，现在要将其嵌入任何产品中，从消费性电子产品到医疗器械再到武器系统。因此，设计已经成为关键活动。
- 个人、企业和政府在进行日常运作管理以及战略战术决策时越来越依赖于软件。软件失效会给个人和企业带来诸多不便，甚至是灾难性的失败。因此，软件应该具有高质量。
- 随着特定应用系统感知价值的提升，其用户群和软件寿命也会增加。随着用户群和使用时间的增加，其适应性和增强型性需求也会同时增加。因此，软件需具备可维护性。

由这些简单事实可以得出一个结论：各种形式、各个应用领域的软件

关键点 在建立解决方案之前应理解问题。

关键点 质量和可维护性都是良好设计的产物。

⊖ 在本书的后续部分，将这些人统称为“利益相关者”。

都需要工程化。这也是本书的主题——软件工程。

2.1 定义软件工程学科

对于软件工程，美国电气与电子工程师学会（IEEE）[IEE93a] 给出了如下定义：

软件工程是：（1）将系统化的、规范的、可量化的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件；（2）对（1）中所述方法的研究。

提问 如何定义软件工程？

然而，对于某个软件开发队伍来说可能是“系统化的、规范的、可量化的”方法，对于另外一个团队却可能是负担。因此，我们需要规范，也需要可适应性和灵活性。

图 2-1 软件工程层次图

软件工程是一种层次化的技术，如图 2-1 所示。任何工程方法（包括软件工程）必须构建在质量承诺的基础之上。全面质量管理、六西格玛和类似的理念[⊖]促进了持续不断的过程改进文化，正是这种文化最终引导人们开发出更有效的软件工程方法。支持软件工程的根基在于质量关注点（quality focus）。

软件工程的基础是过程（process）层。软件过程将各个技术层次结合在一起，使得合理、及时地开发计算机软件成为可能。过程定义了一个框架，构建该框架是有效实施软件工程技术必不可少的。软件过程构成了软件项目管理控制的基础，建立了工作环境以便于应用技术方法、提交工作产品（模型、文档、数据、报告、表格等）、建立里程碑、保证质量及正确的管理变更。

关键点 软件工程包括过程、管理和构建软件的方法和工具。

软件工程方法（method）为构建软件提供技术上的解决方法（如何做）。方法覆盖面很广，包括沟通、需求分析、设计建模、程序构造、测试和技术支持。软件工程方法依赖于一组基本原则，这些原则涵盖了软件工程所有技术领域，包括建模活动和其他描述性技术等。

网络资源

《Cross Talk》杂志提供了关于过程、方法和工具的许多实践信息，网站地址是 www.stsc.hill.af.mil。

软件工程工具（tool）为过程和方法提供自动化或半自动化的支持。这些工具可以集成起来，使得一个工具产生的信息可被另外一个工具使用，这样就建立了软件开发的支撑系统，称为计算机辅助软件工程（computer-aided software engineering）。

提问 软件过程的元素是什么？

2.2 软件过程

软件过程是工作产品构建时所执行的一系列活动、动作和任务的集合。活动（activity）主要实现宽泛的目标（如与利益相关者进行沟通），与应用领域、项目大小、结果复杂性或者实施软件工程的重要程度没有直接关系。动作（action，如体系结构设计）包含了主要工作产品（如体系结构设计模型）生产过程中的一系列任务。任务（task）关注小而明确的目标，能够产生实际产品（如构建一个单元测试）。

引述 过程定义了活动的时间、人员、工作内容和达到预期目标的途径。

Ivar Jacobson, Grandy Booch, and James Rumbaugh

⊖ 质量管理和相关方法的内容在本书第三部分进行讨论。

在软件工程领域，过程不是对如何构建计算机软件的严格的规定，而是一种具有可适应性的调整方法，以便于工作人员（软件团队）可以挑选适合的工作动作和任务集合。其目标
16 通常是及时、高质量地交付软件，以满足软件项目资助方和最终用户的需求。

2.2.1 过程框架

过程框架（process framework）定义了若干个框架活动（framework activity），为实现完整的软件工程过程建立了基础。这些活动可广泛应用于所有软件开发项目，无论项目的规模和复杂性如何。此外，过程框架还包含一些适用于整个软件过程的普适性活动（umbrella activity）。一个通用的软件工程过程框架通常包含以下5个活动。

提问　五个最基本的过程框架活动是什么？

沟通。在技术工作开始之前，和客户（及其他利益相关者[⊖]）的沟通与协作是极其重要的，其目的是理解利益相关者的项目目标，并收集需求以定义软件特性和功能。

引述　爱因斯坦认为必然存在着一个对自然界的简单解释，因为上帝既不专制也不喜怒无常。然而软件工程师却无法抱侥幸心理，多数情况下，他必须面对毫无规律的复杂性。
Fred Brooks

策划。如果有地图，任何复杂的旅程都可以变得简单。软件项目好比是一个复杂的旅程，策划活动就是创建一个“地图”，以指导团队的项目旅程，这个地图称为软件项目计划，它定义和描述了软件工程工作，包括需要执行的技术任务、可能的风险、资源需求、工作产品和工作进度计划。

建模。无论你是庭园设计师、桥梁建造者、航空工程师、木匠还是建筑师，你每天的工作都离不开模型。你会画一张草图来辅助理解整个项目大的构想——体系结构、不同的构件如何结合，以及其他一些特性。如果需要，可以把草图不断细化，以便更好地理解问题并找到解决方案。软件工程师也是如此，需要利用模型来更好地理解软件需求，并完成符合这些需求的软件设计。

构建。必须要对所做的设计进行构建，包括编码（手写的或者自动生成的）和测试，后者用于发现编码中的错误。

部署。软件（全部或者部分增量）交付给用户，用户对其进行评测并给出反馈意见。

17 上述五个通用框架活动既适用于简单小程序的开发，也可用于WebApp的建造以及基于计算机的大型复杂系统工程。不同的应用案例中，软件过程的细节可能差别很大，但是框架活动都是一致的。

对许多软件项目来说，随着项目的开展，框架活动可以迭代应用。也就是说，在项目的多次迭代过程中，沟通、策划、建模、构建、部署等活动不断重复。每次项目迭代都会产生一个软件增量（software increment），每个软件增量实现了部分的软件特性和功能。随着每一次增量的产生，软件将逐渐完善。

2.2.2 普适性活动

软件工程过程框架活动由很多普适性活动来补充实现。通常，这些普适性活动贯穿软件项目始终，以帮助软件团队管理和控制项目进度、质量、变更和风险。典型的普适性活动包括如下活动。

⊖ 利益相关者（stakeholder）就是可在项目成功中分享利益的人，包括业务经理、最终用户、软件工程师、支持人员等。Rob Thomsett曾开玩笑说：“stakeholder就是掌握巨额投资（stake）的人……如不照看好你的stakeholder，就会失去投资。”

软件项目跟踪和控制——项目组根据计划来评估项目进度，并且采取必要的措施保证项目按进度计划进行。

风险管理——对可能影响项目成果或者产品质量的风险进行评估。

软件质量保证——确定和执行保证软件质量的活动。

技术评审——评估软件工程产品，尽量在错误传播到下一个活动之前发现并清除错误。

测量——定义和收集过程、项目以及产品的度量，以帮助团队在发布软件时满足利益相关者的要求。同时，测量还可与其他框架活动和普适性活动配合使用。

软件配置管理——在整个软件过程中管理变更所带来的影响。

可复用管理——定义工作产品复用的标准（包括软件构件），并且建立构件复用机制。

工作产品的准备和生产——包括生成产品（如建模、文档、日志、表格和列表等）所必需的活动。

上述每一种普适性活动都将在本书后续部分详细讨论。

关键点 普适性活动贯穿整个软件过程，主要关注项目管理、跟踪和控制。

关键点 对软件过程的适应性调整是项目成功的关键。

2.2.3 过程的适应性调整

在本节前面部分曾提到，软件工程过程并不是教条的法则，也不要求软件团队机械地执行；而应该是灵活可适应的（根据软件所需解决的问题、项目特点、开发团队和组织文化等进行适应性调整）。因此，不同项目所采用的项目过程可能有很大不同。这些不同主要体现在以下几个方面：

- 活动、动作和任务的总体流程以及相互依赖关系。
- 在每一个框架活动中，动作和任务细化的程度。
- 工作产品的定义和要求的程度。
- 质量保证活动应用的方式。
- 项目跟踪和控制活动应用的方式。
- 过程描述的详细程度和严谨程度。
- 客户和利益相关者对项目的参与程度。
- 软件团队所赋予的自主权。
- 队伍组织和角色的明确程度。

本书第1部分将详细介绍软件过程。

18

引述 我认为食谱只是指导方法，一个聪明的厨师每次都会变化出不同的特色。
Madame Benoit

2.3 软件工程实践

在2.2节中，曾介绍过一种由一组活动组成的通用软件过程模型，建立了软件工程实践的框架。通用的框架活动——**沟通、策划、建模、构建和部署**——和普适性活动构成了软件工程工作的体系结构的轮廓。但是软件工程的实践如何融入该框架呢？在以下几节里，读者将会对应用于这些框架活动的基本概念和原则有一个基本了解。⊖

网络资源 软件工程实践方面各种深刻的想法都可以在以下网址获得：www.literate programming.com。

⊖ 在本书的后面对于特定软件工程方法和普适性活动的讨论中，你应该重读本章中的相关章节。

2.3.1　实践的精髓

在现代计算机发明之前，有一本经典著作《How to Solve it》，在书中，George Polya[Pol45]列出了解决问题的精髓，这也正是软件工程实践的精髓：

1. 理解问题（沟通和分析）。
2. 策划解决方案（建模和软件设计）。
3. 实施计划（代码生成）。
19 4. 检查结果的正确性（测试和质量保证）。

建议　你可能会觉得Polya的方法只是简单的常识，的确如此。但是令人惊奇的是，在软件世界里，很多常识常常不为人知。

在软件工程中，这些常识性步骤引发了一系列基本问题（与[Pol45]相对应）：

建议　理解问题最重要的是倾听。

理解问题。虽然不愿承认，但生活中的问题很多都源于我们的傲慢。我们只听了几秒钟就断言，好，我懂了，让我们开始解决这个问题吧。不幸的是，理解一个问题不总是那么容易，需要花一点时间回答几个简单问题：

- 谁将从问题的解决中获益？也就是说，谁是利益相关者？
- 有哪些是未知的？哪些数据、功能和特性是解决问题所必需的？
- 问题可以划分吗？是否可以描述为更小、更容易理解的问题？
- 问题可以图形化描述吗？可以建立分析模型吗？

策划解决方案。现在你理解了要解决的问题（或者你这样认为），并迫不及待地开始编码。在编码之前，稍稍慢下来做一点点设计：

- 以前曾经见过类似问题吗？在潜在的解决方案中，是否可以识别一些模式？是否已经有软件实现了所需要的数据、功能和特性？
- 类似问题是否解决过？如果是，解决方案所包含元素是否可以复用？
- 可以定义子问题吗？如果可以，子问题是否已有解决方案？
- 能用一种可以很快实现的方式来描述解决方案吗？能构建出设计模型吗？

实施计划。前面所创建的设计勾画了所要构建的系统的路线图。可能存在没有想到的路径，也可能在实施过程中会发现更好的解决路径，但是这个计划可以保证在实施过程中不至迷失方向。需要考虑的问题是：

引述　在任何问题的解决方案中都会有所发现。
George Polya

- 解决方案和计划一致吗？源码是否可追溯到设计模型？
- 解决方案的每个组成部分是否可以证明正确？设计和代码是否经过评审？或者采用更好的方式，算法是否经过正确性证明？

20 **检查结果**。你不能保证解决方案是最完美的，但是可以保证设计足够的测试来发现尽可能多的错误。为此，需回答：

- 能否测试解决方案的每个部分？是否实现了合理的测试策略？
- 解决方案是否产生了与所要求的数据、功能和特性一致的结果？是否按照项目利益相关者的需求进行了确认？

不足为奇，上述方法大多是常识。但实际上，有充足的理由可以证明，在软件工程中采用常识将让你永远不会迷失方向。

2.3.2　通用原则

原则这个词在字典里的定义是“某种思想体系所需要的重要的根本规则或者假设”。在

本书中，我们将讨论一些不同抽象层次上的原则。一些原则关注软件工程的整体，另一些原则考虑特定的、通用的框架活动（比如沟通），还有一些关注软件工程的动作（比如架构设计）或者技术任务（比如编制用例场景）。无论关注哪个层次，原则都可以帮助我们建立一种思维方式，进行扎实的软件工程实践。因此，原则非常重要。

David Hooker[Hoo96] 提出了 7 个关注软件工程整体实践的原则，这里复述如下。⊖

第 1 原则：存在价值

建议 在开始一个软件项目之前，应首先确保软件具有商业目标并且让用户体会到它的价值。

一个软件系统因能为用户提供价值而具有存在价值，所有的决策都应该基于这个思想。在确定系统需求之前，在关注系统功能之前，在决定硬件平台或者开发过程之前，问问你自己：这确实能为系统增加真正的价值吗？如果答案是不，那就坚决不做。所有的其他原则都以这条原则为基础。

第 2 原则：保持简洁

提问 简洁比所有巧妙的措词更加美妙。

Alexander Pope (1688-1744)

软件设计并不是一种随意的过程，在软件设计中需要考虑很多因素。所有的设计都应该尽可能简洁，但不是过于简化。这有助于构建更易于理解和易于维护的系统。这并不是说有些特性（甚至是内部特性）应该以“简洁”为借口而取消。的确，优雅的设计通常也是简洁的设计，但简洁也不意味着“快速和粗糙”。事实上，它经常是经过大量思考和多次工作迭代才达到的，这样做的回报是所得到的软件更易于维护且错误更少。

21

第 3 原则：保持愿景

清晰的愿景是软件项目成功的基础。没有愿景，项目将会由于它有“两种或者更多种思想”而永远不能结束；如果缺乏概念的一致性，系统就好像是由许多不协调的设计补丁、错误的集成方式强行拼凑在一起……如果不能保持软件系统体系架构的愿景，就会削弱甚至彻底破坏设计良好的系统。授权体系架构师，使其能够持有愿景，并保证系统实现始终与愿景保持一致，这对项目开发的成功至关重要。

第 4 原则：关注使用者

关键点 如果软件有价值，那么其价值在其生命周期中将发生变更。因此，软件必须构建成可维护的。

有产业实力的软件系统不是在真空中开发和使用的。通常软件系统必定是由开发者以外的人员使用、维护和编制文档等，因此必须要让别人理解你的系统。在需求说明、设计和实现过程中，牢记要让别人理解你所做的事情。对于任何一个软件产品，其工作产品都可能有很多用户。需求说明时应时刻想到用户，设计中始终想到实现，编码时想着那些要维护和扩展系统的人。一些人可能不得不调试你所编写的代码，这使得他们成了你所编写代码的使用者，尽可能地使他们的工作简单化会大大提升系统的价值。

第 5 原则：面向未来

生命周期持久的系统具有更高的价值。在现今的计算环境中，需求规格说明随时会改变，硬件平台几个月后就会淘汰，软件生命周期都是以月而不是以年来衡量的。然而，真正具有“产业实力”的软件系统必须持久耐用。为了成功地做到这一点，系统必须能适应各种变化，能成功做到这一点的系统都是那些一开始就以这种路线来设计的系统。永远不要把自己的设计局限于一隅，经常问问“如果出现……应该怎样应对”，构建可以解决通用问题的

⊖ 这里的引用得到了作者的授权 [Hoo96].Hooker 定义这些原则的模式请参见：http://c2.com/cgi/wiki?SevenPrinciplesOfSoftwareDevelopment。

系统，为各种可能的方案做好准备，[⊖]这很可能会提高整个系统的可复用性。

第 6 原则：提前计划复用

22 复用既省时又省力[⊜]。软件系统开发过程中，高水平的复用是很难实现的一个目标。曾有人宣称代码和设计复用是面向对象技术带来的主要好处，然而，这种投入的回报不会自动实现。为达到面向对象（或是传统）程序设计技术所能够提供的复用性，需要有前瞻性的设计和计划。系统开发过程中的各个层面上都有多种技术可实现复用……*提前做好复用计划将降低开发费用，并增加可复用构件以及构件化系统的价值。*

第 7 原则：认真思考

*这最后一条规则可能是最容易被忽略的。在行动之前清晰定位、完整思考通常能产生更好的结果。*仔细思考可以提高做好事情的可能性，而且也能获得更多的知识，明确如何把事情做好。如果仔细思考过后还是把事情做错了，那么，这就变成了很有价值的经验。思考就是学习和了解本来一无所知的事情，使其成为研究答案的起点。把明确的思想应用在系统中，就产生了价值。使用前六条原则需要认真思考，这将带来巨大的潜在回报。

如果每位工程师、每个开发团队都能遵从 Hooker 这七条简单的原则，那么，开发复杂计算机软件系统时所遇到的许多困难都可以迎刃而解。

2.4　软件开发神话

软件开发神话，即关于软件及其开发过程的一些被人盲目相信的说法，这可以追溯到计算技术发展的初期。神话具有一些特点，让人觉得不可捉摸。例如，神话看起来是事实的合理描述（有时的确包含真实的成分），它们符合直觉，并且经常被那些知根知底的有经验的从业人员拿来宣传。

今天，大多数有见地的软件工程师已经意识到软件神话的本质——它实际上误导了管理者和从业人员对软件开发的态度，从而引发了严重的问题。然而，由于习惯和态度的根深蒂固，软件神话遗风犹在。

管理神话。像所有领域的经理一样，承担软件职责的项目经理肩负着维持预算、保证进度和提高质量的压力。就像溺水人抓住稻草一样，软件经理经常依赖软件神话中的信条，只要它能够减轻以上的压力（即使是暂时性的）。

网络资源　软件项目经理网有助于人们澄清这些神话：www.spmn.com。

神话：我们已经有了一本写满软件开发标准和规程的宝典，难道不能提供我们所需要了解的所有信息吗？

23 **事实：**这本宝典也许的确已经存在，但它是否在实际中采用了？从业人员是否知道这本书的存在呢？它是否反映了软件工程的现状？是否全面？是否可以适应不同的应用环境？是否在缩短交付时间的同时还关注产品质量的保证？在很多情况下，问题的答案是否定的。

神话：如果我们未能按时完成计划，我们可以通过增加程序员人数而赶上进度（即所谓的“蒙古游牧”概念）。

事实：软件开发并不是像机器制造那样的机械过程。Brooks 曾说过 [Bro95]：“在软件工

⊖ 把这个建议发挥到极致会非常危险，设计通用方案会带来性能损失，并降低特定的解决方案的效率。

⊜ 尽管对于准备在未来的项目中复用软件的人而言，这种说法是正确的，但对于设计和实现可复用构件的人来说，复用的代价会很昂贵。研究表明，设计和开发可复用构件比直接开发目标软件要增加 25% ～ 200% 的成本，在有些情况下，这些费用差别很难核实。

程中，为赶进度而增加人手只能使进度更加延误。”初看，这种说法似乎与直觉不符。然而，当新人加入到一个软件项目后，原有的开发人员必须要牺牲本来的开发时间对后来者进行培训，因此减少了本应用于高效开发的时间。只有在有计划且有序进行的情况下，增加人员对项目进度才有意义。

神话：如果决定将软件外包给第三方公司，就可以放手不管，完全交给第三方公司开发。

事实：如果开发团队不了解如何在内部管理和控制软件项目，那么将无一例外地在外包项目中遇到困难。

客户神话。软件产品的客户可能是隔壁的某个人、楼下的一个技术团队、市场/销售部门或者签订软件合同的某个外部公司。多数情况下，客户之所以相信所谓的软件神话，是因为项目经理和从业人员没有及时纠正他们的错误信息。软件神话导致客户错误的期望，最终导致对开发者的不满。

神话：有了对项目目标的大概了解，便足以开始编写程序，可以在之后的项目开发过程中逐步充实细节。

建议 在项目开始之前，尽可能努力了解工作内容。也许难以明确所有细节，但你了解得越多，所面临的风险就越低。

事实：虽然通常很难得到综合全面且稳定不变的需求描述，但是对项目目标模糊不清的描述将为项目实施带来灾难。要得到清晰的需求描述（经常是逐步变得清晰的），只能通过客户和开发人员之间的持续有效的沟通。

神话：虽然软件需求不断变更，但是因为软件是弹性的，因此可以很容易地适应变更。

事实：软件需求的确在随时变更，但随变更引入的时机不同，变更所造成的影响也不同。如果需求变更提出得较早（比如在设计或者代码开发之前），则对费用的影响较小[⊖]；但是，随着时间的推移，变更的代价也迅速增加——因为资源已经被分配，设计框架已经建立，而变更可能会引起的剧变，需要添加额外的资源或者修改主要设计。 24

从业者神话。在60多年的编程文化的滋养下，软件开发人员依然深信着各种神话。在软件业发展早期，编程被视为一种艺术。旧有的方式和态度根深蒂固。

神话：当我们完成程序并将其交付使用之后，我们的任务就完成了。

事实：曾经有人说过，对于编程来说，开始得越早，耗费的时间就越长。业界的一些数据显示，60%～80%的工作耗费在软件首次交付顾客使用之后。

建议 每当你认为没有时间采用软件工程方法时，就再问问自己：“是否有时间重做整个软件?”

神话：直到程序开始运行，才能评估其质量。

事实：最有效的软件质量保证机制之一——技术评审，可以从项目启动就开始实行。软件评审（第20章）作为“质量过滤器”，已经证明其可以比软件测试更为有效地发现多种类型的软件缺陷。

神话：对于一个成功的软件项目，可执行程序是唯一可交付的工作成果。

事实：软件配置包括很多内容，可执行程序只是其中之一。各种工作产品（如模型、文档、计划）是成功实施软件工程的基础，更重要的是，为软件技术支持提供了指导。

⊖ 许多软件工程师采纳了“敏捷”(agile)开发方法，通过增量的方式逐步纳入变更，以便控制变更的影响范围和成本。本书第5章讨论敏捷方法。

神话：软件工程将导致我们产生大量无用文档，并因此降低工作效率。

事实：软件工程并非以创建文档为目的，而是为了保证软件产品的开发质量。好的质量可以减少返工，从而加快交付时间。

目前，大多数软件专业人员已经认识到软件神话的谬误。对于软件开发真实情况的正确
25 理解是系统阐述如何使用软件工程方法解决实际问题的第一步。

2.5　这一切是如何开始的

每个软件工程项目都来自业务需求——对现有应用程序缺陷的纠正，改变遗留系统以适应新的业务环境，扩展现有应用程序功能和特性，或者开发某种新的产品、服务或系统。

在软件项目的初期，业务需求通常是在简短的谈话过程中非正式地表达出来的。以下这段简短谈话就是一个典型的例子。

SafeHome[⊖] 如何开始一个软件项目

[场景] CPI公司的会议室里。CPI是一个虚构的为家庭和贸易应用生产消费产品的公司。

[人物] Mal Golden，产品开发部高级经理；Lisa Perez，营销经理；Lee Warren，工程经理；Joe Camalleri，业务发展部执行副总裁。

[对话]

Joe: Lee，我听说你们那帮家伙正在开发一个产品——通用的无线盒？

Lee: 哦，是的，那是一个很棒的产品，只有火柴盒大小。我们可以把它放在各种传感器上，比如数码相机，总之任何东西里。采用802.11n无线网络协议，可以通过无线连接获得它的输出。我们认为它可以带来全新的一代产品。

Joe: Mal，你觉得怎么样呢？

Mal: 我当然同意。事实上，随着这一年来销售业绩的趋缓，我们需要一些新的产品。Lisa和我已经做了一些市场调查，我们都认为该系列产品具有很大的市场潜力。

Joe: 多大，底线是多少？

Mal（避免直接承诺）：Lisa，和他谈谈我们的想法。

Lisa: 这是新一代的家庭管理产品，我们称之为“SafeHome”。产品采用新型无线接口，给家庭和小型商务从业人士提供一个由电脑控制的系统——住宅安全、监视，仪表和设备控制。例如，你可以在回家的路上关闭家里的空调，或者如此这类的应用。

Lee（插话）：Joe，工程部已经作了相关的技术可行性研究。它可行且制造成本不高。大多数硬件可以在市场购买产品，不过软件方面是个问题，但也不是我们不能做的。

Joe: 有意思！我想知道底线。

Mal：在美国，70%的家庭拥有电脑。如果我们定价合适，这将成为一个十分成功的产品。到目前为止，只有我们拥有这一无线控制盒技术。我们将在这方面保持两年的领先地位。收入吗，在第二年大约可达到3000万到4000万。

Joe（微笑）：我很感兴趣，让我们继续讨论一下。

⊖ SafeHome项目将作为一个案例贯穿本书，以便说明项目组在开发软件产品过程中的内部工作方式。公司、项目和人员都是虚构的，但场景和问题是真实的。

除了一带而过地谈到软件，这段谈话中几乎没有提及软件开发项目。然而，软件将是SafeHome产品线成败的关键。只有SafeHome软件成功，该产品才能成功。只有嵌入其中 26
的软件产品满足顾客的需求（尽管还未明确说明），产品才能被市场所接受。我们将在后面的几章中继续讨论SafeHome中软件工程的话题。

2.6 小结

软件工程包含过程、方法和工具，这些工具使得快速构建高质量的复杂计算机系统成为可能。软件过程包括五个框架活动：沟通、策划、建模、构建和部署，这些活动适用于所有软件项目。软件工程实践遵照一组核心原则，是一项解决问题的活动。

尽管我们关于构建软件所需的软件知识和技能增长了，但仍有大量的软件神话将管理者和从业人员诱入歧途。随着对软件工程理解的深化，你就会逐渐明白，为什么无论何时遇到这些神话，都要不遗余力地揭露。

习题与思考题

2.1 图2-1中，将软件工程的三个层次放在了“质量关注点”这层之上。这意味着在整个开发组织内采用质量管理活动，如“全面质量管理”。仔细研究并列出全面质量管理活动中关键原则的大纲。

2.2 软件工程对构建WebApp是否适用？如果适用，需要如何改进以适应WebApp的独特特点？

2.3 随着软件的普及，由于程序错误所带来的公众风险已经成为一个愈加重要的问题。设想一个真实场景：由于软件错误而引起“世界末日”般的重大危害（危害社会经济或是人类生命财产安全）。

2.4 用自己的话描述过程框架。当我们谈到框架活动适用于所有的项目时，是否意味着对于不同规模和复杂度的项目可应用相同的工作任务？请解释。

2.5 普适性活动存在于整个软件过程中，你认为它们均匀分布于软件过程中，还是集中在某个或者某些框架活动中？

2.6 在2.4节所列举的神话中，增加两种软件神话，同时指出与其相对应的真实情况。

扩展阅读与信息资源

软件工程及软件过程的当前发展状况可以参阅一些期刊，如《IEEE Software》《IEEE Computer》《CrossTalk》和《IEEE Transactions on Software Engineering》。《Application Development Trends》和《Cutter IT Journal》等行业期刊通常包含一些关于软件工程的文章。每年，IEEE和ACM资助的 27
研讨会论文集《Proceeding of the International Conference on Software Engineering》都是对当年学术成果的总结，并且在《ACM Transactions on Software Engineering and Methodology》《ACM Software Engineering Notes》和《Annals of Software Engineering》等期刊上有进一步深入讨论。当然，在互联网上有很多关于软件工程和软件过程的网页。

近年出版了许多关于软件过程和软件工程的书籍，有些是关于整个过程的概要介绍，有些则深入讨论过程中一些重要专题。下面是一些畅销书（除本书之外）：

《SWEBOK: Guide to the Software Engineering Body of Knowledge》[⊖]，IEEE, 2013，见http:www.computer.org/portal/web/swebok。

Andersson, E. 等，《Software Engineering for Internet Applications》，MIT Press, 2006。

Braude, E. 和 M. Bernstein,《Software Engineering: Modern Approaches》，2nd ed., Wiley,

⊖ 可从http://www.computer.org/portal/web/swebok/htmlformat免费下载。

2010。

Christensen, M. 和 R. Thayer,《A Project Manager's Guide to Software Engineering Best Practices》, IEEE-CS Press (Wiley), 2002。

Glass, R.,《Fact and Fallacies of Software Engineering》, Addison-Wesley, 2002。

Hussain, S.,《Software Engineering》, I K International Publishing House, 2013。

Jacobson, I.,《Object-Oriented Software Engineering: A Use Case Driven Approach》, 2nd ed., Addison-Wesley, 2008。

Jalote, P.,《An Integrated Approach to Software Engineering》, 3rd ed., Springer, 2010。

Pfleeger, S.,《Software Engineering: Theory and Practice》, 4th ed., Prentice Hall, 2009。

Schach, S.,《Object-Oriented and Classical Software Engineering》, 8th ed., McGraw-Hill,2010).

Sommerville, I.,《Software Engineering》, 9th ed., Addison-Wesley, 2010。

Stober, T., 和 U. Hansmann,《Agile Software Development: Best Practices for Large Development Projects》, Springer, 2009。

Tsui, F., 和 O.karam,《Essentials of Software Engineering》, 2nd ed., Jones&Bartlett Publishers, 2009。

Nygard(《Release it!: Design and Deploy Production-Ready Software》, Pragmatic Bookshelf, 2007)、Richardson 和 Gwaltney (《Ship it! A Practical Guide to Successful Software Projects》, Pragmatic Bookshelf, 2005) 以及 Humble 和 Farley (《Continues Delivery: Reliable Software Releases through Build, Test, and Deployment Automation》, Addison-Wesley, 2010) 的书给出了大量有用的指导原则，可用于部署活动。

在过去的几十年里，IEEE、ISO 以及附属其下的标准化组织发布了大量软件工程标准。Moore (《The Road Map to Software Engineering: A Standards-Based Guide》, IEEE Computer Society Press[Wiley], 2006) 对相关标准进行了调查并指出了这些标准应如何应用到实际工程中。

网上有很多有关软件工程和软件过程相关问题的信息资源，与软件过程相关的最新参考文献可以在 SEPA 网站 www.mhhe.com/pressman 找到。

28