第 1 章

Software Engineering: A Practitioner's Approach, Eighth Edition

软件的本质

要点浏览

概念：计算机软件是由专业人员开发并长期维护的软件产品。完整的软件产品包括：可以在各种不同容量及系统结构的计算机上运行的程序、程序运行过程中产生的各种结果以及各种描述信息，这些信息可以以硬拷贝或是各种电子媒介形式存在。

人员：软件工程师开发软件并提供技术支持，产业界中几乎每个人都间接或直接地使用软件。

重要性：软件之所以重要是因为它在我们的生活中无所不在，并且日渐深入到商业、文化和日常生活的各个方面。

步骤：客户和利益相关者表达对计算机软件的要求，工程师构建软件产品，最终用户应用软件来解决特定的问题或者满足特定的要求。

工作产品：在一种或多种特定环境中运行并服务于一个或多个最终用户要求的计算机软件。

质量保证措施：如果你是软件工程师，就要应用本书中包含的思想。如果你是最终用户，应确保理解了自己的要求和环境，然后选择能很好地满足两者的应用软件。

关键概念

应用领域
云计算
失效曲线
遗留软件
移动 App
产品线
软件定义
软件问题
软件的本质
磨损
WebApp

在给我演示了最新开发的世界上最流行的第一人称射击视频游戏之后，这位年轻的开发者笑了。

“你不是一个玩家，对吗？”他问道。

我微笑道：“你是怎么猜到的？”

这位年轻人身着短裤和 T 恤衫。他的腿像活塞那样上下跳动，燃烧着神经能量，这在他的同事中看起来是很平常的。

“因为如果你是玩家，”他说，“你应该会更加兴奋。你已经看到了我们的下一代产品，我们的客户会对它着迷……这不是开玩笑。”

我们坐在开发区，这是地球上最成功的游戏开发者之一。在过去几年中，他演示的前几代游戏售出了 5000 万份，收入达几亿美元。

“那么，这一版什么时候上市？”我问道。

他耸耸肩，“大约在 5 个月以内，我们还有很多工作要做。”

他负责一个应用软件中的游戏和人工智能功能，该软件包含的代码超过了 300 万行。

“你们使用任何软件工程技术吗？”我问道，估计他会笑笑并摇头。

他停顿了一下，想了一会儿，然后缓慢地点点头。“我们让软件工程技术适应我们的需求，但是，我们确实使用。”

“在什么地方使用？”我试探地问道。

“我们的问题是经常将需求翻译成创意。”

“创意?”我打断了他的话。

“你知道，那些设计故事、人物及所有游戏素材的家伙，他们想的是游戏大卖。而我们不得不接受他们抛给我们的这些，并形成一组技术需求，从而构建游戏。”

“那么形成了需求之后呢?”

他耸耸肩，“我们不得不扩展并修改以前游戏版本的体系结构，并创建新的产品。我们需要根据需求创建代码，对每日构建的代码实施测试，并且做你书中建议的很多事情。”

“你知道我的书?”老实说，我非常惊讶。

“当然，在学校使用。那里有很多。”

“我已经与你的很多同事谈了，他们对我书中的很多东西持怀疑态度。”

他皱了皱眉，“你看，我们不是IT部门或航空公司，所以我们不得不对你所提倡的东西进行取舍。但是底线是一样的——我们需要生产高质量的产品，并且以可重复方式完成这一目标的唯一途径是改写我们自己的软件工程技术子集。”

“那么这个子集是如何随着时间的推移变更的?”

他停顿了一下，像是在思考着未来。“游戏将变得更加庞大和复杂，那是肯定的。随着更多竞争的出现，我们的开发时间表将会收缩。慢慢地，游戏本身会迫使我们应用更多的开发规范。如果我们不这样做，我们就会死掉。”

引述 创新观念和科技发现是经济增长的推进器。
华尔街日报

计算机软件仍然是世界舞台上最为重要的技术，并且也是“意外效应法则”的典型例子。60年前，没有人曾预料到软件科学会成为今天商业、科学和工程所必需的技术。软件促进了新科技的创新（例如基因工程和纳米科技）、现代科技的发展（例如通信）以及传统技术的根本转变（例如媒体行业），软件技术已经成为个人计算机革命的推动力量，消费者使用智能手机就可以购买软件产品。软件还将由产品逐渐演化为服务，软件公司随需应变，通过Web浏览器发布即时更新功能，软件公司几乎可以比所有工业时代的公司都更大、更有影响力。在大量应用软件的驱动下，互联网将迅速发展，并逐渐改变人们生活的诸多方面——从图书馆搜索、消费购物、政治论战到年轻人和（不很年轻的）成年人的约会行为。

2

没有人曾想到软件可嵌入各种系统中：交通运输、医疗、通信、军事、工业、娱乐以及办公设备……不胜枚举。如果笃信“意外效应法则”的话，那么还有很多结果和影响是我们尚未预料到的。

没有人曾想到，随着时间的推移，将有数百万的计算机程序需要进行纠错、适应性调整和优化，这些维护工作将耗费比开发新软件更多的人力和物力。

随着软件重要性的日渐凸现，软件业界一直试图开发新的技术，使得高质量计算机程序的开发和维护更容易、更快捷、成本更低廉。一些技术主要针对特定应用领域（例如网站设计和实现），另一些着眼于技术领域（例如面向对象系统、面向方面的程序设计），还有一些覆盖面很宽（例如像LINUX这样的操作系统）。然而，我们尚未开发出一种可以实现上述所有需求的软件技术，而且未来能够产生这种技术的可能性也很小。人们也尚未将其工作、享受、安全、娱乐、决策以及全部生活都完全依赖于计算机软件。这或许是正确的选择。

本书为需要构建正确软件的计算机软件工程师提供了一个框架。该框架包括过程、一系列方法以及我们称为软件工程的各种工具。

1.1　软件的本质

关键点　软件既是产品也是交付产品的载体。

现在的软件具有产品和产品交付载体的双重作用。作为产品，软件显示了由计算机硬件体现的计算能力，更广泛地说，显示的是由一个可被本地硬件设备访问的计算机网络体现的计算潜力。无论是安装在移动电话、手持平板电脑、台式机还是大型计算机中，软件都扮演着信息转换的角色：产生、管理、获取、修改、显示或者传输各种不同的信息，简单如几个比特的传递，复杂如从多个独立的数据源获取的多媒体演示。而作为产品生产的载体，软件提供了计算机控制（操作系统）、信息通信（网络）以及应用程序开发和控制（软件工具和环境）的基础平台。

引述　软件是播撒梦想和收获噩梦的地方，是一片恶魔和神仙相竞争的抽象而神秘的沼泽，是一个狼人和银弹共存的矛盾世界。
Brad J.Cox

软件提供了我们这个时代最重要的产品——信息。它转换个人数据（例如个人财务交易），从而使信息在一定范围内发挥更大的作用；它通过管理商业信息提升竞争力；它为世界范围的信息网络提供通路（例如因特网），并为各类格式的信息提供不同的查询方式。软件还提供了可以威胁个人隐私的载体，并给那些怀有恶意目的的人提供了犯罪的途径。

在最近半个世纪里，计算机软件的作用发生了很大的变化。硬件性能的极大提高、计算机结构的巨大变化、存储容量的大幅度增加以及种类繁多的输入和输出方法都促使基于计算机的系统更加先进和复杂。如果系统开发成功，那么“先进和复杂”可以产生惊人的效果，但是同时复杂性也给系统的开发人员和防护人员带来巨大的挑战。

现在，一个庞大的软件产业已经成为了工业经济中的主导因素。早期的独立程序员也已经被专业的软件开发团队所代替，团队中的不同专业技术人员可分别关注复杂应用系统中的某一部分技术。然而同过去的独立程序员一样，开发现代计算机系统时，软件开发人员依然面临同样的问题：⊖

- 为什么软件需要如此长的开发时间？
- 为什么开发成本居高不下？
- 为什么在将软件交付顾客使用之前，我们无法找到所有的错误？
- 为什么维护已有的程序要花费如此多的时间和人工？
- 为什么软件开发和维护的进度仍旧难以度量？

种种问题显示了业界对软件以及软件开发方式的关注，这种关注导致了业界对软件工程实践方法的采纳。

1.1.1　定义软件

今天，绝大多数专业人员和许多普通人认为他们对软件大概了解。真的是这样吗？

提问　如何定义软件？

来自教科书的关于软件的定义也许是：

软件是：（1）指令的集合（计算机程序），通过执行这些指令可以满足预期的特性、功能和性能需求；（2）数据结构，使得程序可以合理利用信息；（3）软件描述信息，它以硬拷贝和虚拟形式存在，用来描述程序的操作和使用。

⊖ 在一本优秀的关于软件业务的论文集中，Tom DeMarco[DeM95] 提出了相反的看法。他认为：“我们更应该总结使得当今的软件开发费用低廉的成功经验，而不是不停地质问为何软件开发成本高昂。这会有助于我们继续保持软件产业的杰出成就。”

4　当然，还有更完整的解释。但是一个更加正式的定义可能并不能显著改善其可理解性。为了更好地理解“软件”的含义，有必要将软件和其他人工产品的特点加以区分。软件是逻辑的而非物理的系统元素。因此，软件和硬件具有完全不同的特性：软件不会“磨损”。

图 1-1 描述了硬件的失效率，该失效率是时间的函数。这个名为“浴缸曲线”的关系图显示：硬件在早期具有相对较高的失效率（这种失效通常来自设计或生产缺陷）；在缺陷被逐个纠正之后，失效率随之降低并在一段时间内保持平稳（理想情况下很低）；然而，随着时间推移，因为灰尘、振动、不当使用、温度超限以及其他环境问题所造成的硬件组件损耗累积的效果，使得失效率再次抬高。简而言之，硬件开始磨损了。

建议　若希望降低软件退化，则需要改进软件的设计（第 12～18 章）。

而软件是不会被引起硬件磨损的环境问题所影响的。因此，从理论上来说，软件的失效率曲线应该呈现为图 1-2 的“理想曲线”。未知的缺陷将在程序生命周期的前期造成高失效率。然而随着错误的纠正，曲线将如图中所示趋于平缓。“理想曲线”只是软件实际失效模型的粗略简化。曲线的含义很明显——软件不会磨损，但是软件退化的确存在。

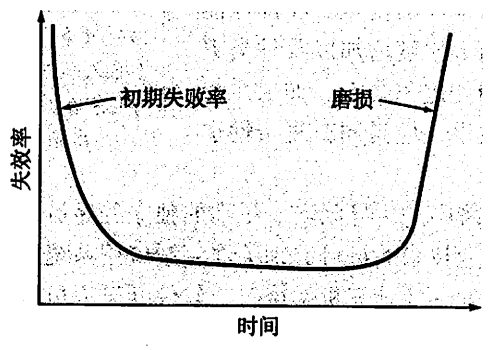

图 1-1　硬件失效曲线图

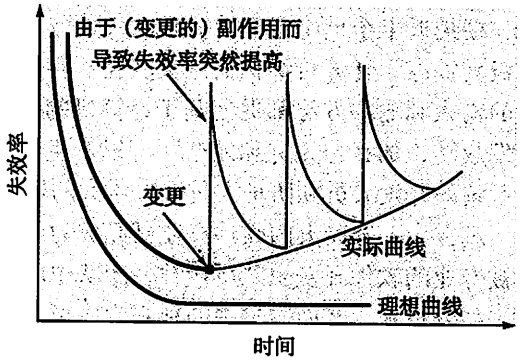

图 1-2　软件失效曲线图

5　这个似乎矛盾的现象用图 1-2 所示的“实际曲线”可以很好地解释。在完整的生命周期里，⊖软件将会面临变更，每次变更都可能引入新的错误，使得失效率像“实际曲线”（图 1-2）那样陡然上升。在曲线回到最初的稳定失效率状态前，新的变更会引起曲线又一次上升。就这样，最小的失效率点沿类似于斜线的形状逐渐上升，可以说，不断的变更是软件退化的根本原因。

关键点　软件工程方法的目的是降低图 1-2 中曲线向上突变的幅度及实际失效曲线的斜率。

磨损的另一方面同样说明了软硬件的不同。磨损的硬件部件可以用备用部件替换，而软件却不存在备用部件。每个软件的缺陷都暗示了设计的缺陷或者在从设计转化到机器可执行代码的过程中产生的错误。因此，软件维护要应对变更请求，比硬件维护更为复杂。

1.1.2　软件应用领域

今天，计算机软件可分为七个大类，软件工程师正面临持续的挑战。

系统软件——一整套服务于其他程序的程序。某些系统软件（例如编译器、编辑器、文

⊖ 事实上，在软件开发开始，在第一个版本发布之前的很长时间，许多利益相关者可能提出变更要求。

件管理软件）处理复杂但确定的[⊖]信息结构，另一些系统应用程序（例如操作系统构件、驱动程序、网络软件、远程通信处理器）主要处理的是不确定的数据。

应用软件——解决特定业务需要的独立应用程序。这类应用软件处理商务或技术数据，以协助业务操作或协助做出管理或技术决策。

网络资源　目前最大的共享软件/免费软件库之一：shareware.cnet.com。

6

工程/科学软件——“数值计算”（number crunching）类程序涵盖了广泛的应用领域，从天文学到火山学，从自动压力分析到轨道动力学，从计算机辅助设计到分子生物学，从遗传分析到气象学。

嵌入式软件——嵌入式软件存在于某个产品或者系统中，可实现和控制面向最终用户和系统本身的特性和功能。嵌入式软件可以执行有限的和内部的功能（例如微波炉的按键控制），或者提供重要的功能和控制能力（例如汽车中的燃油控制、仪表板显示、刹车系统等汽车电子功能）。

产品线软件——为多个不同用户的使用提供特定功能。产品线软件关注有限的及内部的市场（例如库存控制产品）或者大众消费品市场。

Web/移动 App——以网络为中心，其概念涵盖了宽泛的应用软件产品，包括基于浏览器的 App 和安装在移动设备上的软件。

人工智能软件——利用非数值算法解决计算和直接分析无法解决的复杂问题。这个领域的应用程序包括机器人、专家系统、模式识别（图像和语音）、人工神经网络、定理证明和博弈等。

引述　对于我来说，电脑是我们能够想出的最重要的工具。它相当于思想的自行车。

Steve Jobs

全世界成百万的软件工程师在为以上各类软件项目努力地工作着。有时是建立一个新的系统，而有时只是对现有应用程序的纠错、适应性调整和升级。一个年轻的软件工程师所经手项目本身的年限比他自己的年龄还大是常有的事。对于上述讨论的各类软件，上一代的软件工程师都已经留下了遗留系统。我们希望这代工程师留下的遗留系统可以减轻未来工程师的负担。

1.1.3　遗留软件

成千上万的计算机程序都可以归于在前一小节中讨论的 7 大类应用领域。其中某些是当今最先进的软件——最近才对个人、企业和政府发布。但是另外一些软件则年代较久，甚至过于久远了。

这些旧的系统——通常称为**遗留软件**（legacy software）——从 20 世纪 60 年代起，就成为人们持续关注的焦点。Dayani-Fard 和他的同事 [Day99] 这样描述遗留软件：

7

遗留软件系统……在几十年前诞生，它们不断被修改以满足商业需要和计算平台的变化。这类系统的繁衍使得大型机构十分头痛，因为它们的维护代价高昂且系统演化风险较高。

Liu 和他的同事 [Liu98] 进一步扩展了这个描述，指出“许多遗留软件系统仍然支持核心的商业功能，是业务必不可少的支撑。”因此，遗留软件具有生命周期长以及业务关键性的特点。

然而不幸的是，遗留软件常常存在另一个特点——质量差[⊜]。遗留软件通常拥有数不清的

⊖ 软件的确定性是指系统的输入、处理和输出的顺序及时间是可以预测的，软件的不确定性是指系统的输入、处理和输出的顺序和时间是无法提前预测的。

⊜ 所谓“质量差”是基于现代软件工程思想的，这个评判标准对遗留系统有些不公平，因为在遗留软件开发的年代里，现代的软件工程一些概念和原则可能还没有被人们完全理解。

问题：遗留系统的设计难以扩展，代码令人费解，文档混乱甚至根本没有，测试用例和结果并未归档，变更的历史管理混乱……然而，这些系统仍然支撑着“核心的业务功能，并且是业务必不可少的支撑”。该如何应对这种情况？

提问 如果遇到质量低下的遗留软件该怎么办？

最合理的回答也许就是什么也不做，至少在其不得不进行重大变更之前什么也不做。如果遗留软件可以满足用户的需求并且能可靠运行，那么它就没有失效，不需要修改。然而，随着时间的推移，遗留系统经常会由于下述原因而发生演化：

提问 对遗留软件都进行了哪些类型的改变？

- 软件需要进行适应性调整，从而可以满足新的计算环境或者技术的需求。
- 软件必须升级以实现新的商业需求。
- 软件必须扩展以使之具有与更多新的系统和数据库的互操作能力。
- 软件架构必须进行改建以使之能适应不断演化的计算环境。

建议 所有软件工程师都需认识到变更是不可避免的。不要反对变更。

当这些变更发生时，遗留系统需要经过再工程（第36章）以适应未来的多样性。当代软件工程的目标是“修改在进化论理论上建立的方法论”，即“软件系统不断经历变更，新的软件系统从旧系统中建立起来，并且……新旧所有系统都必须具有互操作性和协作性。”[Day99]。

8

1.2 软件的变更本质

四大类软件不断演化，在行业中占有主导地位。这四类软件在十几年前还处于初级阶段。

1.2.1 WebApp

万维网（WWW）的早期（大约从1990年到1995年），Web站点仅包含链接在一起的一些超文本文件，这些文件使用文本和有限的图形来表示信息。随着时间的推移，一些开发工具（例如XML、Java）扩展了HTML（超级文本标记语言）的能力，使得Web工程师在向客户提供信息的同时也能提供计算能力。基于Web的系统和应用软件[⊖]（我们将这些总称为WebApp）诞生了。

引述 对于任何类型的稳定性，Web都会变成完全不同的东西。
Louis Monier

今天，WebApp已经发展成为成熟的计算工具，这些工具不仅可以为最终用户提供独立的功能，而且已经同公司数据库和业务应用系统集成在一起了。

十年前，WebApp“演化为一种混合体，介于印刷出版和软件开发之间、市场和计算之间内部通信和外部关系之间以及艺术和技术之间。”[Pow98]，但是，如1.1.2节所述，当前WebApp在很多应用类型中提供了丰富的计算能力。

在过去的十多年中，语义Web技术（通常指Web 3.0）已经演化为成熟的企业和消费者应用软件，包括“提供新功能的语义数据库，这些新功能需要Web链接、灵活的数据表示以及外部访问API（应用编程接口）。”[Hen10]成熟的关系型数据结构导致了全新的WebApp，允许以多种方式访问不同的信息，这在以前是不可能做到的。

⊖ 在本书中，WebApp这个术语包含了很多事物，从一个简单的帮助消费者计算汽车租借费用的网页，到为商务旅行和度假提供全套旅游服务的大型复杂的Web站点。其中包括完整的Web站点、Web站点的专门功能以及在Internet、Intranet或Extranet上的信息处理应用软件。

1.2.2　移动 App

术语 App 已经演化为在移动平台（例如 iOS、Android 或 Windows Mobile）上专门设计的软件。在大多数情况下，移动 App 包括用户接口，用户接口利用移动平台所提供的独特的交互机制，基于 Web 资源的互操作性提供与 App 相关的大量信息的访问，并具有本地处理能力，以最适合移动平台的方式收集、分析和格式化信息。此外，移动 App 提供了在平台中的持久存储能力。

9

提问　WebApp 和移动 App 之间的差别是什么？

认识到移动 WebApp 与移动 App 之间的微妙差异是非常重要的。移动 WebApp 允许移动设备通过针对移动平台的优点和弱点专门设计的浏览器获取基于 Web 内容的访问。移动 App 可以直接访问设备的硬件特性（例如加速器或者 GPS 定位），然后提供前面所述的本地处理和存储能力。随着时间的推移，移动浏览器会变得更加成熟，并可获取对设备级硬件和信息的访问，这将使得移动 WebApp 和移动 App 之间的区别变得模糊。

1.2.3　云计算

云计算包括基础设施或“生态系统”，它能使得任何用户在任何地点都可以使用计算设备来共享广泛的计算资源。云计算的总体逻辑结构如图 1-3 所示。

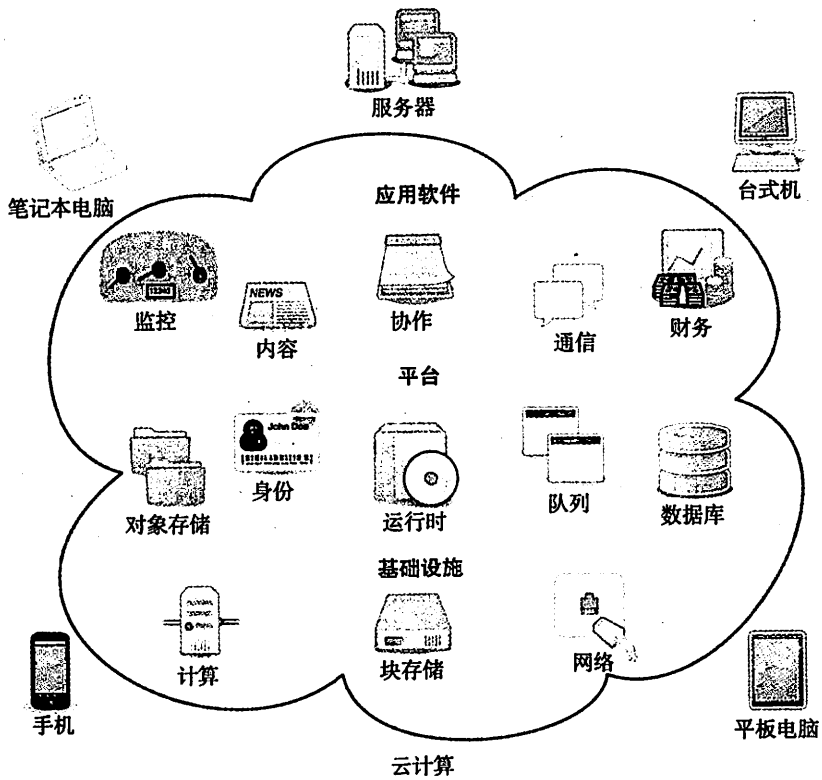

图 1-3　云计算的逻辑结构 [Wik13]

10

如图所示，计算设备位于云的外部，可以访问云内的各种资源。这些资源包括应用软件、平台和基础设施。最简单的形式是，外部计算设备通过 Web 浏览器或类似的软件访问云。云提供对存储在数据库或其他数据结构中的数据的访问。此外，设备可访问可执行的应

用软件，可以用这种应用程序代替计算设备上的 App。

云计算的实现需要开发包含前端和后端服务的体系结构。前端包括客户（用户）设备和应用软件（如浏览器），用于访问后端。后端包括服务器和相关的计算资源、数据存储系统（如数据库）、服务器驻留应用程序和管理服务器。通过建立对云及其驻留资源的一系列访问协议，管理服务器使用中间件对流量进行协调和监控。[Str08]

可以对云体系结构进行分段，以提供不同级别的访问，从公共访问到只对授权用户提供访问的私有云体系结构。

1.2.4　产品线软件

美国卡内基·梅隆大学软件工程研究所（SEI）将软件产品线定义为“一系列软件密集型系统，可以共享一组公共的可管理的特性，这些特性可以满足特定市场或任务的特定需求，并以预定的方法从一组公共的核心资源开发出来。”[SEI13] 以某种方式使软件产品相互关联，可见，软件产品线的概念并不是新概念。但是，软件产品线的思想提供了重要的工程影响力，软件产品线都使用相同的底层应用软件和数据体系结构来开发，并使用可在整个产品线中进行复用的一组软件构件来实现。

软件产品线共享一组资源，包括需求（第 8 章）、体系结构（第 13 章）、设计模式（第 16 章）、可重用构件（第 14 章）、测试用例（第 22、23 章）及其他的软件工程工作产品。本质上，软件产品线是对很多产品进行开发的结果，在对这些产品进行工程设计时，利用了产品线中所有产品的公共性。

1.3　小结

软件是以计算机为基础的系统和产品中的关键部分，并且是世界舞台上最重要的技术之一。在过去的 50 年里，软件已经从解决特定问题和信息分析的工具发展为独立的产业。然而，如何在有限的时间内利用有限的资金开发高质量的软件，这仍然是我们所面对的难题。

软件——程序、数据和描述信息——覆盖了科技和应用的很多领域。遗留软件仍旧给维护人员带来了特殊的挑战。

软件的本质是变更。基于 Web 的系统和 App 已经从简单的信息内容集合演化为能够展示复杂功能和多媒体信息的复杂系统。尽管 WebApp 具有独特的特性和需求，但它们仍然属于软件范畴。由于 App 已迁移到很多的平台上，因此移动 App 展示出了新的挑战。云计算将转变软件交付的方式及软件存在的环境。产品线软件提供了构建软件的潜在效率。

习题与思考题

1.1. 举出至少 5 个例子来说明“意外效应法则”在计算机软件方面的应用。

1.2. 举例说明软件对社会的影响（包括正面影响和负面影响）。

1.3. 针对 1.1 节提出的 5 个问题给出你的答案，并与同学讨论。

1.4. 在交付最终用户之前，或者首个版本投入使用之后，许多现代 App 程序都会有频繁的变更。为防止变更引起软件退化，请提出一些有效的解决措施。

1.5. 思考 1.1.2 节中提到的 7 个软件分类。请问能否将一个软件工程方法应用于所有的软件分类？并就你的答案加以解释。

扩展阅读与信息资源[⊖]

在数千本关于计算机软件的书中，大多数讨论的是程序设计语言和软件应用系统，很少有涉及软件本身的。Pressman 和 Herron（《Software Shock》，Dorset House，1991）最早讨论了软件和专业开发方法的问题（针对门外汉）。Negroponte 的畅销书（《Being Digital》，Alfred A. Knopf，Inc.，1995）提供了关于计算及其在 21 世纪的发展和影响的观点。Demarco（《Why does Software Cost So Much？》，Dorset House，1995）就软件和开发过程发表了一系列惊人且见解独到的论文。Ray Kurzweil（《How to Create a Mind》，Viking，2013）讨论了软件如何在不久的将来就会模仿人类思想，并带来人类和机器进化的“奇异性”。

Keeves（《Catching Digital》，Business Infomedia Online，2012）讨论了商业领导者应该如何适应以不断增大的步伐进行演化的软件。Minasi 在著作（《The Software Conspiracy: Why Software Companies Put Out Faulty Products，How They Can Hurt You，and What You Can Do》，McGraw-Hill，2000）中认为，现在由于软件缺陷引起的“现代灾难”将被消除并提出了解决的方法。Eubanks（《Digital Dead End: Fighting for Social Justice in the Information Age》，MIT Press，2011）和 Compaine（《Digital Divide: Facing a Crisis or Creating a Myth》，MIT Press，2001）的书认为，在 21 世纪的第一个十年里，信息（如 Web 资源）富有者和信息贫困者之间的数字鸿沟将越来越小。Kuniavsky（《Smart Things: Ubiquitous Computing User Experience Design》，Morgan Kaufman，2010）、Greenfield（《Everyware: The Dawning Age of Ubiquitous Computing》，New Riders Publishing，2006）和 Loke（《Context-Aware Pervasive Systems: Architectures for a New Breed of Applications》，Auerbach，2006）的著作介绍了“开放世界”软件的概念，并指出在无线网络环境中软件必须能够进行适应性调整，以满足实时涌现的需求。

网上有很多讨论软件本质的信息资源，与软件过程相关的最新参考文献可在 SEPA 网站 www.mhhe.com/pressman 找到。

12
~
13

⊖ 在每章小结之后，“扩展阅读与信息资源”一节简单介绍了本章相关资料，便于读者扩展阅读和深入理解本章内容。针对本书，我们已经建立了网站 www.mhhe.com/pressman。网站涉及软件工程的很多主题，逐章列出了相关的软件工程网站资料信息以作为本书的补充，并给出了每一本书在 Amazon.com 的链接。