HZ BOOKS
华章教育

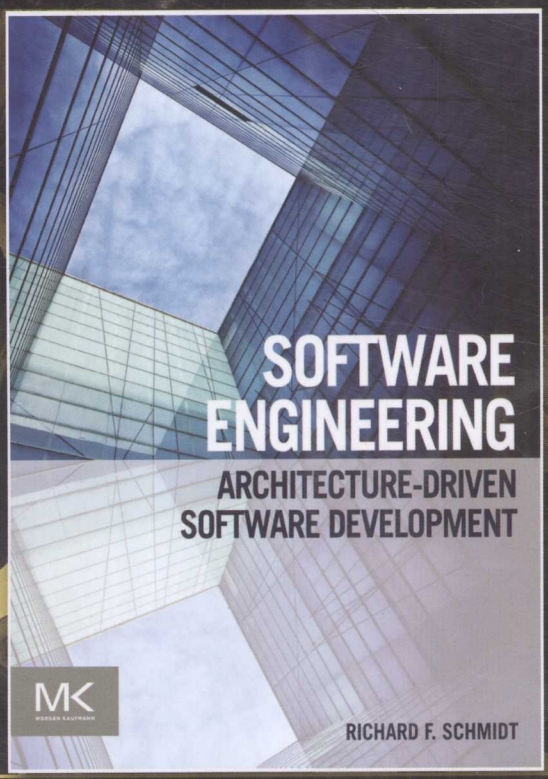
计 算 机 科 学 丛 书

ELSEVIER
爱思唯尔

软件工程
架构驱动的软件开发

[美] 理查德 F. 施密特（Richard F. Schmidt）著
江贺 李必信 周颖 等译

Software Engineering
Architecture-Driven Software Development



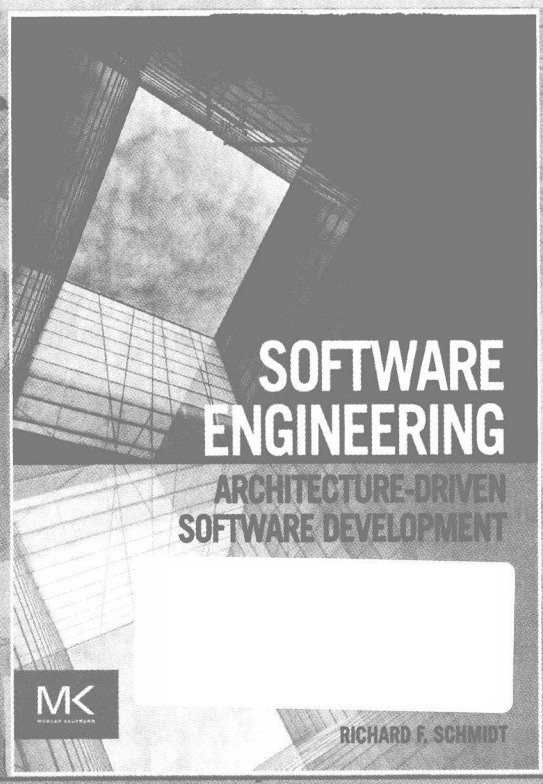
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

软件工程
架构驱动的软件开发

[美] 理查德 F. 施密特（Richard F. Schmidt） 著
江贺 李必信 周颖 等译

Software Engineering
Architecture-Driven Software Development



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件工程：架构驱动的软件开发 /（美）理查德 F. 施密特（Richard F. Schmidt）著；江贺等译 . —北京：机械工业出版社，2016.7
（计算机科学丛书）
书名原文：Software Engineering：Architecture-Driven Software Development

ISBN 978-7-111-53314-6

I. 软… II. ① 理… ② 江… III. 软件工程 IV. TP311.5

中国版本图书馆 CIP 数据核字（2016）第 140294 号

本书版权登记号：图字：01-2013-7842

本书比较全面地介绍软件工程学科，展示软件工程原则与基于系统工程的软件实践，阐明与软件工程所用的严格方法相关的实践活动、原则、任务和工件。本书共分三部分：第一部分讨论在软件工程体系下的软件开发框架和项目构建；第二部分通过 6 项技术惯例传达一种理念——利用计算技术，应用科学原则以及激活设计软件产品结构的灵活性；第三部分讨论软件工程团队在软件开发项目中扮演的角色，以便建立和控制软件产品架构。

本书适合作为高等院校软件工程及相关课程的教材，也可作为软件开发人员和软件技术人员的参考书。

出版发行：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：谢晓芳　　责任校对：殷　虹
印　　刷：北京瑞德印刷有限公司　　版　　次：2016 年 7 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：14.75
书　　号：ISBN 978-7-111-53314-6　　定　　价：69.00 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

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译 者 序

Software Engineering: Architecture-Driven Software Development

自从1968年提出“软件工程”这一术语以来，研究软件工程的专家、学者们陆续提出了100多条关于软件工程的准则或信条。然而，软件工程项目失败、进度落后、预算超支等问题仍屡见不鲜。部分原因是目前的教育更关注于编程以及模块级的设计，而缺乏对软件架构设计的理解。没有一个完整的设计概念，编码就会很低效，进而造成软件产品生命周期中的设计基础不稳定。软件行业项目成功率不理想正是作者Richard F. Schmidt出版并推广本书的动机所在，他希望借此对未来的软件工程师有所启发。

Richard F. Schmidt在航空航天系统工程和软件工程方面拥有30多年的研发经验。他曾参与防御系统软件开发的美国国防部标准2167A（DoD-STD-2167A）的修订和防御系统软件质量保证的美国国防部标准2168（DoD-STD-2168）的制定。Richard还负责了IEEE 1220标准（系统工程过程的应用和管理）的发布。本书阐述了那些开发政府或企业系统项目的公认的软件工程实践。本书介绍的是跨学科的软件开发方法。本书的内容与IEEE计算机协会的软件工程知识体系（SWEBOK）相对应，重点是整合各种软件开发方法和对开发有效且高效的软件产品必不可少的架构化设计实践。

全书分为三个部分，共20章。第一部分主要介绍软件工程基础，包括软件开发框架、软件架构、项目环境和软件集成以及软件设计的阻碍；第二部分侧重于软件工程实践，这涉及软件需求分析、需求管理、功能架构制定、功能分析与分配实践、物理架构配置、软件设计综合实践、软件分析实践、软件验证和确认实践、软件控制实践等；第三部分重点分析软件工程应用的各个阶段，从软件需求定义到软件验收测试。

希望广大读者可以通过本书对架构驱动的软件开发方式多一些了解，对软件工程过程多一些思考。受译者水平所限，书中如有错漏和不当，欢迎大家指正讨论。

江贺

本书提出了几个颇有争议的话题。这些争议性的话题涵盖了“软件工程”的范围，并且是作者出版本书的动机核心。如果软件工程学科很好地建立起来，并且被证明取得了成功的结果，那么就不再需要出版和推广本书了。然而，并非如此。在过去 20 年中，软件行业项目的成功率都徘徊在 30% 左右。这些项目的失败都可以和两个几乎在软件开发项目或方法中随处可见的主要问题联系在一起。第一个问题涉及对于什么是软件产品设计以及如何开发一个完整的设计描述几乎完全的误解。第二个问题涉及缺乏一套可用于为软件工程学科建立合适范围的软件工程原则和实践的标准。

本书提供了一套全面、集成并且紧密耦合的实践。然而，这些内容背离了当下流行的“最佳实践”，因为它们缺乏设计软件的完美方法。其中的一些观点可能有些批判性；当提出一个方法去修复缺陷系统时，批评是不可避免的。这样做的目的是通过建立一套重要的软件工程原则和实践来鼓励软件社区进行广泛交流。

我希望读者能够保留自己关于软件工程中主流概念的观点。不要让这些有争议的话题分散你的注意力。本书为软件产品的设计提供了一个严谨的、严格的方法。全体软件社区是时候采取行动来提升惨淡的业绩了。我希望这本书能让未来几代专业软件工程师受益。

——理查德·F·施密特（Richard F. Schmidt）

前　言

Software Engineering: Architecture-Driven Software Development

本书旨在比较全面地介绍软件工程学科，展示软件工程原则与基于系统工程的软件实践。本书详细地解释了基本的软件工程体系理念，即强调使用严格规范的方法来设计软件产品。为达到此目的，第一部分讨论了在软件工程体系下的软件开发框架和项目构建。第二部分展示了 6 项技术惯例，它们传达了这样一种理念：利用计算技术，应用科学原则以及激活设计软件产品架构（即设计）的灵活性。第三部分讨论了软件工程团队在软件开发项目中扮演的角色，以便建立和控制软件产品架构。典型软件开发项目的每个阶段都会讨论的重点是软件工程团队如何与其他技术和项目相关的团体协作来影响架构设计和软件产品实现。这几部分阐明了与软件工程所用的严格方法相关的实践、原则、任务和工件。

本书的基础概念基于系统工程实践来达到表 1 确定的目标。这些目标通过应用一系列来源于系统工程学科中 50 多年来成功应用于开发复杂系统的原则和实践来实现。它强调完整软件架构的建立，这使得产品的每个元素都要明确，以便制造、组装、集成和测试。将这些实践应用到软件工程领域，为解决表 1 中列出的那些挑战提供了基础。

表 1　软件工程挑战与目标

软件工程挑战	目标
在编码之前必须先做设计	在提高成本效率和进度准确性前弄清楚正在构建什么
	减少产品在设计细节和精度上的复杂性
	成本管理、进度安排和风险控制
交付软件技术数据包	完整的设计图表和软件实现（构建）的说明文档
分配设计配置元素间的需求	软件组件和单元间的需求分解与分配
	需求可追踪性
集成产品和过程开发（IPPD）	产品维护性能的并行设计与开发
	生命周期成本控制
准备软件集成策略	架构设计活动中计划的软件组件集成开发
	高效的软件实现规划
控制软件复杂性	降低软件维护 / 支持成本
	高效、用户友好的交互
使变更同化	涉众 / 用户满意度
	产品竞争力
权衡分析	成本管理和进度控制
	设计优化
	产品演变 / 增量发布的稳定性
	项目成功率的增强
预先计划的产品提升	将某些功能延迟发布来保证产品按时交付

软件分析与设计的当前实践基于计算机编程语言和这些语言处理数据使用的逻辑概念。这驱动了诸如面向对象设计的软件设计方法，它并不是用来处理先进软件产品的复杂性的。通过适应系统工程实践，本书建立了严格的软件工程原则和实践，从而提供了一种全面的方法来设计软件产品。这些软件工程实践必须清晰说明以保证它们对软件开发的重要性和适用性是确定的。将这些实践应用于一个软件开发过程演练中，以便可以控制、修正和管理贯

穿整个软件开发项目上下文的软件架构。本书的内容与软件工程知识体系[⊖]（SWEBOK）的主要过程领域（如表 2 所示）相对应。与 SWEBOK 的对应说明了本书中的主题是如何根据 SWEBOK 中主题进行安排和关联的。然而，SWEBOK 是基于现在的软件开发实践，严格从技术上来讲，它并不包括系统工程实践。

表 2　SWEBOK 关键过程领域

关键过程领域	本书范围
软件需求知识领域	第一部分：第 3 章
	第二部分：第 7 章、第 9 章
	第三部分：第 17 章
软件设计知识领域	第一部分：第 3 章、第 6 章
	第二部分：第 10 ～ 14 章
	第三部分：第 18 章
软件构造知识领域	第三部分：第 19 章
软件测试知识领域	第三部分：第 19 章、第 20 章
软件维护知识领域	第一部分：第 5 章
	第三部分：第 17 ～ 20 章
软件配置管理知识领域	第二部分：第 9 章、第 16 章
	第三部分：第 20 章，提到配置审核（FCA/PCA）
软件工程管理知识领域	第一部分：第 4 章
	第二部分：第 9 章、第 16 章（提到项目和技术规划，提到工作包）
	第三部分，提到项目和技术方案，提到工作包
软件工程过程知识领域	第二部分
	第三部分
软件工程方法知识领域	第二部分：第 13 章（提到软件设计综合实践面向对象的方法）、第 14 章（提到建模和原型）
软件质量知识领域	第三部分，在测试和评估子部分确定软件质量保证任务

本书内容

接下来会简要介绍书中各章的内容。这三部分将本书内容分成三个连贯的主题，希望读者能增加他们对原则（第一部分）、实践（第二部分）以及软件工程的应用（第三部分）的认识和理解。通过将系统工程实践应用到软件工程领域，本书旨在提供一种创新的、规范的、技术上具有挑战性的方法来开发软件产品。

第一部分：软件工程基础

这一部分讨论的是软件工程相关的基本原则以及这些原则在软件开发场景中的执行。这些基本的原则、实践和理论用于将软件工程建立成一个专业学科。通过讨论软件产品特点和软件开发策略来强调软件开发项目面对的挑战。作为一个组织实体，软件工程弥补了存在于技术专家和项目管理专家之间的外观和感觉上的显著不同。因此，这一部分陈述了软件工程实践与项目管理责任和其他软件开发角色的集成。

⊖ SWEBOK由IEEE计算机学会制定，具体内容参见*http://www.computer.org/portal/web/swebok*。

第 1 章概述了软件工程概念、原则和实践，它是解决设计、开发复杂软件产品面临的挑战所必需的。通过调查软件工程实践和工具来确定它们与项目管理机制的关系。

第 2 章讨论了那些描述软件产品如何定义、设计和实现的软件开发活动的进度。本章通过一系列被项目里程碑和评审分离的连续开发阶段，来追踪典型的软件开发工作，还陈述了软件技术与项目管理控制领域的关系。

第 3 章确定了软件架构的组成，包括以下几个方面：软件产品、计算环境和那些能满足产品维护的开发后的过程。它涉及架构设计表示、模型和文档对技术和项目相关机制和必要性，以保证软件开发工程对在预算内按期交付是必不可少的。需要讨论建立软件需求规格的技术，功能和物理架构要与软件开发阶段一致。本章讨论了软件产品架构如何为软件实现（编程设计、编码、集成和测试）提供结构化基础以及产品生命周期支持。

第 4 章使读者了解那些导致软件开发变得复杂并且难以理解的软件产品特征。它解决了软件产品复杂性挑战，并且将这些与那些已被证明有利于成功完成软件开发工作的项目构建和实践相关联。其中的真知灼见将帮助减少项目的障碍、变动、作废和失败。

第 5 章提出了 IPPD 理论和它对于项目范围的影响以及开发后过程的考虑。它试图证明需要严密构思和结构化的软件架构来确保可以延长产品的使用寿命，这是由于在开发过程中软件工程关注了生命周期问题。同时检查软件开发后过程的工程可以发现，早期的架构决策可以影响生命周期和拥有成本。

第 6 章检查了导致“设计”实践变得非常规并且更加难以理解的软件底层特点。作为挑战传统工程审查的设计和构建材料，讨论软件特征。本章提出了管理软件产品设计的软件工程原则。最后，本章介绍了软件设计歧义来计划一项允许对软件产品进行设计的决议。

第二部分：软件工程实践

这部分确定了 6 个有助于专业化软件工程的实践：1）软件需求分析，2）功能分析与分配，3）软件设计综合，4）软件分析，5）软件验证和确认，6）软件控制。每个实践都由一定数量的任务来表示，每个软件工程的专业人员都应该理解。这些实践建立了一套清晰的任务，集中于设计和细化软件产品架构。

第 7 章提出了一种方法来开发源自涉众需求和期望，以及有助于决定软件开发工作范围的软件需求规约。软件规约驱使软件架构的定义，但不应该推断出任何架构设计方案。软件需求作为派生软件功能和物理架构的起点。架构设计是由表示功能架构和配置物理架而完成的。架构中的每个元素都必须能在软件规约中指定和追踪。要检查软件需求、软件工程任务，以及项目和技术计划之间的关系。

第 8 章确定了必须选择性地应用特定任务来建立软件产品和开发后的过程规约。这种实践涉及软件架构中低级功能和结构元素之间性能限额的分配。这种实践从征求涉众需求和期望的工作开始，以建立软件产品需求基线结束。

第 9 章讨论了以积极主动的方式控制软件架构的重要性，这有利于对已提议的变更进行评估。通过考虑软件需求管理工具和实践以使软件工程团队可以实际考核变更对软件架构的影响和项目资源的自由度来适应一项期望的变更。意图是使开发团队有能力机智地回应得到授权的变更，并且在不扰乱项目范围、计划和成功结束的进度情况下将改进融合到软件架构。

第 10 章讨论了功能架构的本质以及它如何将明确的需求分解为连续的功能元素。每个功能元素在不断改进的方法中是明确的，这些改进会在识别出未完成的功能并且保证它们能实现时告终。功能架构提供了有逻辑性的、连贯的软件产品行为表示法，以回应那些计算环境内出现的刺激、事件和状况。

第 11 章确定了为确保得到完整、一致和可追踪的功能架构而必须考虑的具体任务。通过分析理解业务和软件产品行为，方式包括检查、分解、分类和指定那些来源于需求规约的顶级功能。在功能间分配性能需求来确立低级功能元素有效性和性能的度量标准。

第 12 章描述了安排和确定软件产品物理架构的目的和策略。该物理架构确定了软件单元设计、编码和测试的基本构建快。制定软件集成策略来确定产品结构并规定软件单元和组件如何增量地组合、集成和测试，进而形成完整的软件产品。

第 13 章确定了必须考虑的特定任务，以确保生成完整、一致并且可追踪的物理架构。为了从纯粹的产品功能表示向物理架构过渡，设计综合是一个经过验证的系统工程实践。它涉及“制造或者购买”的权衡，这对应于软件“实现或再利用”决策。

第 14 章确定了必须执行的特定任务，进行设计备选方案的权衡分析和风险评估。进行架构设计决策必须在抑制应用复杂性和软件生命周期成本上有足够的洞察力。与进行权衡分析和风险评估相关的任务可以为理解架构设计决策的本质提供基础，并且对软件开发工作产生影响。

第 15 章确定了必须执行的特定任务，以确保软件架构元素保持一致，并且与授权的变更提议和请求一致。必须执行验证任务以确保软件实现、测试和评估工作与软件架构规约以及设计文档是同步的。

第 16 章确定了选择性应用的特定任务，以确保软件产品架构反映了当前设计思想并包含授权的变更提议、请求和设计决策。需求追踪必须嵌入到软件架构中去，并且与文档关联，这样技术团队才能迅速并且有效地响应变更控制委员会的决策。此外，将授权的变更提议和请求反映在项目和技术计划、进度、预算以及工作包描述中是十分必要的。

第三部分：软件工程应用阶段

这部分讨论了在软件开发项目中，分配给技术组织的角色和职责。强调技术组织在软件工程集成产品团队（IPT）中的参与。

第 17 章确定了由软件工程 IPT 生成的软件需求规约的方式。将参与的组织代表的贡献确定为软件产品的需求，并且建立了开发后的过程。

第 18 章确定了在概要和详细架构阶段定义的软件功能和物理架构的方式。这些阶段关注 IPPD 方法来促进软件实现、测试和开发后的过程必要的基础设施的建立，以推动项目目标的实现。

第 19 章确定了软件实现组织要执行的任务，以编程方式来设计、编码和测试软件单元，并且进行软件集成和测试。在这个阶段同时实现开发后的过程，来支持验收测试和部署准备评审。

第 20 章确定了在软件产品验收测试过程中软件测试和评估组织要执行的任务。参与的组织代表的职责在于监督验收测试、应对测试失败，并且回应由验收测试导致的软件问题报告。此外，开发后的过程必须有资格确认它们已经做好准备来支持软件产品发布、培训和维护业务。

目　录

Software Engineering: Architecture-Driven Software Development

第二部分 软件工程实践

第一部分

Software Engineering: Architecture-Driven Software Development

软件工程基础

本部分概述了软件工程学科，以此使读者熟悉用来描述软件工程原理、实践和任务的词汇。从根本上讲，软件是一种制作软件产品的独特材料。软件作为制作材料的特色对软件专业人士而言是一个谜。通过研究与软件产品工程和设计相关的挑战来减少各种软件工程方法的混乱。通过建立软件工程的基本理论来提供一系列创建软件工程学科的原则和实践。

首先，本部分介绍一组术语，用来讨论健全的软件工程技术的应用。其中的术语汇聚了各个专业，尤其是系统工程、项目管理和配置管理中公认的词汇。这些术语旨在抵制一些软
1 件社区中使用的不合法术语，因为它们会造成混淆，增加建立标准实践的难度。对软件行业而言，至关重要的是稳定该行业同其他工程专业人士、涉众和消费者讨论该行业专业实践的方式。在尝试阐明技术挑战或设计独创性的优点时，出现方言是很不应该的。

Fred Brook（布鲁克斯）教授通过讨论巴别塔阐述了多个软件词典引发的混淆。他的书《人月神话》（*The Mythical Man-Month: Essays on Software Engineering*[㊀]）确认爆发的泥潭是软件方言泛滥的结果。自首次出版之后，该软件文化便在最短的时间内传播了更多编程语言、技术和方法，比人类历史上任何其他行业都快。这个趋势继续分散、弥漫，进而增加了传统工程行业与软件工匠和软件工程印象派之间的沟通障碍。软件工程是一项要求苛刻的职业，它不允许散漫的技术和方法。软件术语和方言必须基于已经建立的工程方法合并为一个统一的词典。源于多种软件语言和方法的泥潭预示着那些尝试与层出不穷的新颖软件战略同步的产业和学术社区有被淹没的威胁。

因此，本部分会耐心地探讨软件产品开发的各种影响。软件开发工作必须借助这些影响或者环境变量才能成功。因此，本部分讨论被软件产品工程的任何方法接受的各种技术和商业动机。从根本上讲，意识到软件产品旨在对采用它们并提供这一独特产品的企业的经济繁荣做出贡献这一点是很必要的。

软件产品的一大类旨在通过自动化常规的、劳动密集的任务来促进企业多产。这些软件产品降低了企业对人工数据收集、分析、操作的依赖，尤其是在计算错误可能导致减少企业盈利的昂贵错误方面。开发软件产品的公司想要从软件产品开发的投资中享受到有益的补偿。因此，软件开发者和使用软件的企业双方都有兴趣确保软件开发项目得到一个专业的、用户友好的、能正常工作的产品。然而，根据 Standish Group 这 20 年的 Chaos 报告[㊁]，软件开发项目的成功率却徘徊在 25% ～ 30%。显然，企业正在对软件行业按时在预算内成功交
2 付产品的能力失去信心。失败！混乱！但这并没有困扰软件专业人员，使他们的声誉太过狼狈不堪。

别怕，因为当绝望的专家跳上另一辆开往某处的花车时，总存在另一个自称软件专家的人具有新奇的软件开发解决方案。另一种方法、扣人心弦的新术语和另一套纷繁的活动准备上场。软件开发产业正拼命地寻找一个解决方案，可以显著地为其辩护增加可信度。软件专家无法接受其他工程学科支持的基本原则和实践，因为软件不同于其他产品。千真万确！然而，可能会从其他工程实践中获得一些适用于软件产品工程的价值。

本部分包括 6 个介绍性章节，它们确定了检查软件工程原则和实践的各个阶段。这些内容基于作者过去 20 年的系统工程实践应用经验。1989 年，题为“程序中的漏洞”（Bugs in the Program[㊂]）的美国国会报告激起了作者对系统工程的研究。这份报告中的发现和建议如下：

㊀ Brooks, F. (1975). *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley.

㊁ 参见 *http://blog.standishgroup.com/pmresearch*.

㊂ Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation, Staff Study by the Subcommittee on Investigations and Oversight, Congress, Sept. 1989.