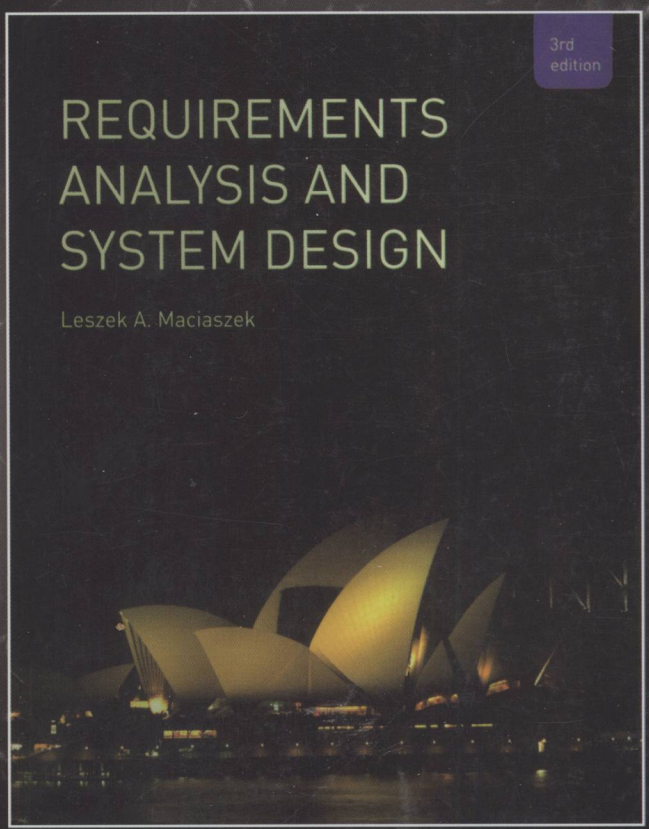
HZ BOOKS
华章教育

计 算 机 科 学 丛 书

原书第3版

需求分析与系统设计

(澳) Leszek A. Maciaszek
Macquarie大学 著 马素霞 王素琴 谢 萍 等译



Requirements Analysis and System Design

Third Edition

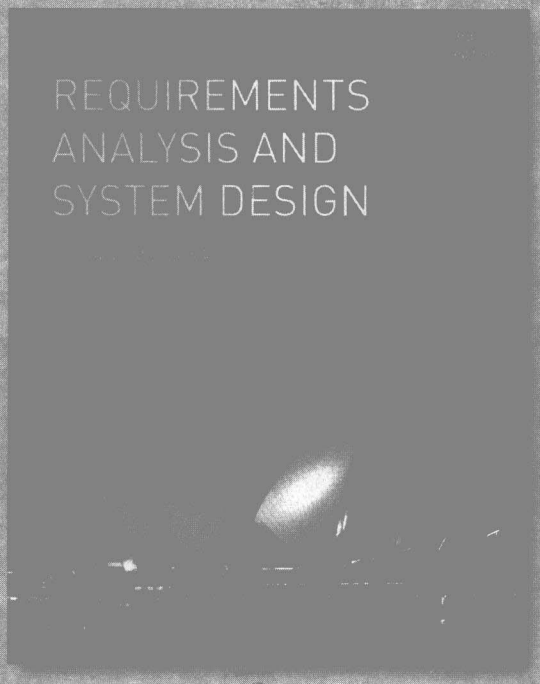
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第3版

需求分析与系统设计

(澳) Leszek A. Maciaszek
Macquarie大学 著 马素霞 王素琴 谢 萍 等译



Requirements Analysis and System Design

Third Edition

机械工业出版社
China Machine Press

本书论述软件分析与设计的原理、方法和技术，并特别关注设计阶段，对软件体系结构的内容进行了很大的扩充。本书强调对象技术及统一建模语言（UML）在企业信息系统开发中的应用，并讨论了使用Web技术和数据库技术进行开发的方法。

本书是大学本科生学习系统分析与设计、软件工程、软件项目管理、数据库和对象技术的理想教材和参考书；对于软件工程技术人员来说，本书也是很好的参考资料。

Leszek A. Maciaszek: Requirements Analysis and System Design, Third Edition (ISBN 978-0-321-44036-5).

译 者 序

随着计算机的日益普及和广泛应用，软件系统的规模和复杂程度与日俱增，软件开发技术面临新的挑战。大型复杂软件的开发是一项特殊的工程。它与传统工程的相同之处是需要按工程学的方法去组织和管理软件的开发。但与传统工程相比，软件工程还有其独特之处。软件开发本身就是一个迭代增量式的过程。在软件生命周期的前两个阶段，即分析和设计阶段更是如此，并且这两个阶段在软件的开发过程中占据至关重要的地位，在很大程度上直接影响了软件项目的成败。

本书论述了软件分析和设计的迭代增量式过程，讨论软件分析与设计的原理、方法和技术，并特别关注了设计阶段，对软件体系结构的内容进行了很大的扩充。本书强调对象技术及统一建模语言（UML）在企业信息系统开发中的应用，并讨论了使用Web技术和数据库技术进行开发的方法。

本书的最大特点是“实例教学”，以七个实例贯穿全书。由于软件分析和设计具有很强的实践性，所以实例对教学来说十分重要。本书对实例不断扩充的过程也体现了软件分析与设计的迭代增量式过程。对初学的学生和软件开发人员来说，这种教学方式是非常有效的。

参与本书翻译工作的主要是华北电力大学及河北经贸大学的教师，包括华北电力大学陈菲（第1章）、石敏（第2章）、马素霞（第3、6、10章、附录）、王素琴（第4、5章）和谢萍（第8、9章）及河北经贸大学林天华（第7章）。在翻译过程中，得到了华北电力大学计算机系张晶、万华的帮助，对他们的辛勤劳动表示感谢。最后，本人对译稿进行了审核与修改。限于水平，对内容的理解和中文表达难免有不当之处，在此敬请读者批评指正。

很高兴能将本书推荐给读者，希望读者能够从中受益。

马素霞

2009年5月于北京

前　言

本书概况

信息系统（information system，IS）的开发（从开始计划到部署给利益相关者）包括三个迭代增量式阶段：分析、设计和实现。本书论述了分析和设计阶段使用的方法和技术。实现方面的问题（包括代码实例）只在设计阶段需要考虑时才讲解，质量与变更管理在第9章单独讨论。

本书集中在面向对象软件开发上。统一建模语言（Unified Modeling Language，UML）用于捕捉建模的人工制品，主要论述用逐步细化的方式进行开发，并且在整个开发生命周期中都使用UML这种建模语言。系统分析师、设计师和程序员使用同一种语言和工具，但有时也会使用一些语言中的方言（配置文件）来满足各自的需要。

对象技术的早期应用主要针对图形用户界面（GUI），并关注开发新系统的速度和程序执行的速度。而在本书中，作者强调对象技术在企业信息系统（enterprise information system，EIS）开发中的应用。其中的挑战是数据量大，数据结构复杂，许多并发用户对信息进行共享式访问，事务处理，需求变更等。对象技术在EIS开发中的主要优势在于可以提高系统的适应性（可理解性、可维护性和可伸缩性）。

开发企业信息系统与进行大规模的分析和设计是同步的。如果不遵循严格的开发过程，不理解基本的软件体系结构，EIS项目就不可能成功。这种开发是大型的、面向对象的、迭代增量式的。

本书提出了用UML进行企业信息系统分析和设计的详细方法，确定了以下几方面的解决方法：

- 分析和建模业务过程。
- 控制大型系统模型的复杂性。
- 改进软件体系结构。
- 提高系统的适应性。
- 处理详细的设计问题。
- 理解图形用户界面。
- 了解数据库的重要性。
- 管理质量、管理变更等。

本书特点

本书的最大特点是“实例教学”。主要的讨论围绕七个实例研究和学习指导形式的复习巩固章节进行。这些例子是从七个应用领域抽取的，每个例子都有各自的特点和教学价值。涉及的领域有大学注册、音像商店、关系管理、电话销售、广告支出、时间记录和货币兑换。学习指导涉及在Internet上购买计算机的在线购物应用系统。

为了便于自学，本书用问题－答案及练习－解决方案的形式阐述了实例研究和学习指导。通过每章末给出的问题和练习，实践材料得到了进一步扩充和丰富。选择题（或练习）都提供了答案（或解决方案）。每章都包含带有答案的复习小测验和选择题，并且都给出了关键术

语的定义。

本书讨论好的分析与设计的原理、方法和技术，并特别关注设计阶段，而设计并不作为分析的直接转换。本书充分考虑大规模系统开发的困难和复杂性，并在许多方面提出了新颖独特的见解，包括“大粒度设计”、大型系统的迭代增量式开发以及在大型软件生产中工具和方法的能力和局限性。

本书只有一章将理论与实践相结合，这既避免了不必要的过度复杂，同时又不失其严谨性。本书从经验的角度进行讲解，与工业无关的或者只具有研究价值的主题并不讨论。

本书采用前沿信息技术，使用可视化系统建模的标准——UML，讨论Web技术和数据库技术的开发方法。本书介绍了Internet驱动的转变，从“胖客户机”（即大型台式计算机）回到了基于服务器的计算。本书讨论的分析和设计原理也适用于传统的客户机/服务器解决方案以及现代的基于构件的分布式应用系统。

软件开发并没有“黑－白”、“真－假”、“0－1”式的解决方案。好的软件解决方案出自好的业务分析员和系统设计师，而不是源于盲目应用的算法。本书的策略是提示读者那些提倡的方法并不能完全解决潜在困难，目的是希望读者能够认真地应用所学的知识，而不应认为这些方法是很容易应用的（从而可能导致更大的失败）。

本书具有以下特点：

- 理论联系实际——按“在领域中”应用该方法所必须解决的实际问题和限制的形式。
- 给予设计阶段特别的关注。本书并不是将设计看成分析的直接转换，而是承认大型企业信息系统开发的困难和复杂性。
- 本书包含了大量实例以及问题、练习、复习小测验和选择题，且大多数附有答案和解决方案。

目标读者

随着与工业实践更加相关的大学课程需求的日益增长，本书的目标就是面向这样的一些学生和实践者。这曾是一个机遇与挑战并存的任务，现在这个任务已经成功地实现了。为了确保继续教育的优势，本书采用了一种不是特定供应商专用的术语来讨论软件开发实现方面的问题（虽然在实例和解决方案中使用了商用的CASE工具）。

本书针对计算机科学和信息系统方面的课程。由于本书既包含“高层”系统建模的内容，也包含“低层”用户界面和数据库设计的问题，所以可作为系统分析与系统设计、软件工程、数据库和对象技术课程的教材，也可用于要求学生按照开发生命周期（从需求确定到用户界面和数据库实现）来开发系统的软件项目课程。本书是为一个学期的课程而设计的，但也可以用于两个学期的课程：一个学期学习需求分析，而另一个学期学习系统设计。

对软件从业人员来说，书中所给出的理论都与实际应用相关，很多问题领域、例子和练习都来自作者的咨询实践工作。本书采取的策略是提醒读者所提出方法的潜在困难或限制。下面的一些从业人员均能从本书获益：业务和系统分析师、设计人员、程序员、系统架构师、软件项目管理者、评审人员、测试人员、技术资料编写人员以及行业培训人员。

本书的组织结构

本书较全面地论述了企业信息系统的面向对象分析和设计，内容的次序与现代开发过程一致。全书包括10章和一个关于对象技术基础内容的附录，在分析和设计两个方面的内容比重均衡。

具有不同背景知识的读者都适合阅读本书。本书最后一章专门对整书内容进行复习和巩固，可以满足很多大学课程的需要。毕竟，复习和巩固原则是任何继续教育的基础。

第2版的变化

虽然本书第1版广受欢迎，并翻译成多种语言，但还是有很多方面需要改进，特别是受技术影响的领域，如系统开发。第2版的主要变化（希望这些变化带来了改进）如下：

- 去掉了子标题“使用UML开发信息系统”。现代系统开发本身就隐含了使用UML建模，因此没有必要向读者声明本书使用UML。
- 在每章的最后增加了奇数编号问题的答案及可选练习的解决方案，这使得本书更适合自学。
- 将以前位于第2章和第6章的“学习指导”抽取成单独的一章，即第10章，并进行了改进和扩充，作为复习和巩固，标题为“复习巩固指南”。
- 第2章和第3章进行了调换，原因是需求分析（现在位于第2章）不需要对象和对象建模方面的预备知识（现在位于第3章）。
- 将第3章的标题改成了“对象和对象建模”，以反映从通过指导概括地解释对象到以例子和UML解释对象的这种转变。
- 丰富了第4章（“需求规格说明”）的内容，讲解了系统体系结构框架对于所开发系统的可支持性是最重要的。因此，本章介绍了体系结构框架，并在后面章节的讨论中得到了实施。
- 更新了第6章的内容，主要是本章的一些内容移到其他章中，而其他章的内容又移到了本章。学习指导已经移到了第10章，某些程序设计主题是从第9章移过来的。系统体系结构方面的讨论进行了很大的扩充。因此，本章的新标题为“系统体系结构与程序设计”。
- 第7章（“用户界面设计”）已经在多方面得到了修改和丰富。使用Java Swing库的优点是对某些UI设计原则进行了解释。以前基于UML活动图的窗口导航模型已经被新的UML配置文件——用户体验（UX）故事情节——所代替。
- 第8章（现在的标题是“持久性与数据库设计”）已经得到了扩充，讨论了持久对象的设计，包括在所采纳的体系结构框架中管理持久性的模式。以前关于对象和对象－关系数据库的讨论已经去掉了（由于关系数据库在企业信息系统中会继续占主导地位以及对象－关系数据库的困难性），节省的空间补充了来自以前的第9章的数据库主题，如事务管理。
- 以前第9章的内容移到了前一章，以前的第10章变成了新的第9章。本章内容也进行了扩充，特别是，增加了测试驱动开发的内容。

第3版的变化

第3版在篇幅上比第2版多了大约25%。每一章都引入了复习小测验（有答案）、选择题（也有答案）和关键术语的定义。每一章（除了第8章）的内容都有修改和增加，对某些章的内部结构进行了改进。第3版的主要变化如下：

- 对第1章进行了非常大的扩充——增加了一些新的内容：解决方案管理框架（ITIL和COBIT）、面向方面的开发和系统集成（需要强调的是，目前的企业系统很少是独立的应用系统开发，大多数开发项目如果不是占优势的项目，就是集成项目）。
- 对第2章也进行了扩充——在开头增加了新的内容：过程层次建模、业务过程建模和解决方案构想。
- 对第3章进行了修改——第2版的3.1节移到了附录，修改和扩充之后更好地反映了UML

标准的最新改进。

- 对第4章进行了扩充和修改，以利用UML的更新版本，并更好地解释体系结构的特性和框架。
- 第5章得到了修改和扩充，在末尾增加了高级交互建模。
- 第6章得到了扩充和修改，在逻辑体系结构一节（6.2节）加进了对系统体系结构复杂性和模式的探讨，并对协作建模一节（6.5节）进行了修改以反映UML标准的相关变化。
- 扩充了第7章，对Web GUI设计进行了探讨。
- 对第9章进行了重新组织和修改，考虑了很多细节。
- 对第10章进行了修改，以反映UML标准的相关变化。
- 如前面所提到的那样，现在增加了附录“对象技术基础”。

补充材料

在相应的网页上提供了内容广泛的补充材料，其中大多数的Web文档对读者都是免费的。本书的主页地址如下：

http://www.booksites.net/maciaszek

http://www.comp.mq.edu.au/books/rasd3ed

网页材料包括：

- Acrobat Reader格式的可打印幻灯片。
- 本书实例研究解决方案的模型文件、学习指导和所有其他建模例子的模型文件（文件格式为Rational Rose、Magic Draw、Enterprise Architect、PowerDesigner和Visio Professional）。

进一步信息

欢迎读者对本书的改进提出评价、更正和建议等，请直接以下面的方式进行联系：

Leszek A. Maciaszek
Department of Computing
Macquarie University
Sydney, NSW 2109
Australia
电子邮箱：leszek@ics.mq.edu.au
网站：www.comp.mq.edu.au/~leszek
电话：+61 2 98509519
传真：+61 2 98509551
地址：North Ryde, Herring Road, Bld. E6A, Room 319

目　录

第1章 软件过程

目标

本章目标是从总体上描述软件开发过程中的若干策略问题，介绍支撑现代软件开发的过程和方法。通过阅读本章，你能够：

- 了解软件开发的本质、社会基础，以及业务系统的开发为何不能完全基于严格的工程和科学原则。
- 学习软件过程标准（CMM、ISO 9000、ITIL）及服从框架（COBIT）。
- 获得策略系统规划和方法（SWOT、VCM、BPR、ISA）的知识，以确保业务目标能够确定信息系统项目。
- 认识到信息系统之间具有很大的差异，这种差异取决于信息系统能够满足的管理水平及其所具有的竞争优势。
- 了解软件开发的结构化方法与面向对象方法的差异。
- 学习软件开发生命周期的各个阶段及跨越生命周期的活动。
- 了解现代及新兴的软件开发模型/方法（螺旋模型、IBM Rational统一过程、模型驱动的体系结构、敏捷软件开发及面向方面的软件开发）。
- 了解7个实例研究，这些实例用于作为贯穿全书的例子和练习。

1.1 软件开发的本质

在关于信息系统（information system, IS）管理的文献中，充满了项目失败、逾期和超预算、有缺陷的解决方案，以及不可维护的系统等例子。虽然大量引用Standish Chaos报告（声称有70%的软件项目失败）是有些夸张（Glass 2005），但毋庸置疑的是，许多“成功的”系统（换句话说，就是已经付款并交付给用户的系统）被可靠性、性能、安全性、可维护性及其他问题所困扰。

为了了解这些问题的原因，我们首先需要了解软件开发的本质。在一篇有代表性的论文中，Brooks（1987）阐述了软件工程的本质问题和意外事件。软件工程的本质问题体现在软件本身所固有的困难中，我们只能承认这些困难——没有获得突破性进展或“银弹”的方法。按照Brooks的说法，软件工程的本质问题是由软件固有的复杂性、一致性、可变性和不可见性所导致的。

软件的“本质困难”定义了软件开发的不变事实。不变事实声明软件是一种创造性开发行为的产品——由工匠而不是优秀艺术家所完成的行为意义上的一种工艺品或艺术品。在典型的情况下，软件并不是制造业重复性行为的结果。

一旦理解了软件开发的不变事实，人们就应该能够处理软件工程的意外事件——由于软件生产实践而带来的困难，可以由人为的干涉来解决。可以将各种“意外困难”分为3类：

- 利益相关者。
- 过程。
- 建模。

1.1.1 软件开发的不变事实

一些重要的软件特性不易受到人为因素的影响，这些特性在所有的软件项目中都保持不变，并需要在项目中得到承认。软件开发的任务是确保不变事实不会失去控制，并且不要对项目施加任何过多的负面影响。

软件本身就是复杂的。在现代软件系统中，复杂性不过是软件规模（如以代码行表示）的函数，以及组成软件产品的构件之间相互依存关系的函数。

软件的复杂性随着软件的应用领域的性质不同而不同。通常情况下，计算密集型应用领域的软件系统比数据密集型应用领域的软件系统的复杂性要低。数据密集型应用系统包括电子商务，它是本书的主题。这样的系统处理大量数据和业务规则，而这些数据和业务规则往往是不一致或不明确的。构建能够容纳所有业务数据、规则和特殊情况的软件一贯是困难的。

Brooks认为，另外3个重要特性（一致性、可变性及不可见性）加重了这种困难。应用软件必须与其所基于的特定硬件/软件平台相符合（一致），也必须与现有的信息系统相符合，并集成在一起。因为业务过程和需求是在不断变化的，所以在建立应用软件时必须能够容纳变化。尽管应用软件提供了可见的输出，但是负责输出的代码通常深深地隐藏在“不可见”的程序语句、二进制代码库，以及周边的系统软件中。

软件是开发出来的，而不是成批制造出来的（Pressman 2005）。当然，我们不能否认，虽然软件工程的发展为开发实践引入了更多的确定性，但是并不能保证软件项目的成功。这可以与传统的工程分支相对比，如土木工程或机械工程。在传统的工程中，产品（人工制品）是以数学般的精确来设计，然后利用机械和生产线来制造（通常为成批制造）的。

一旦将软件产品开发出来，就能够以最小的代价复制（成批制造），但是对于企业信息系统这种情况，从来都不需要复制软件。每个系统都是独特的，并且是为特定企业开发的。困难在于开发，而并不在于成批制造。因此，整个软件生产的成本都在于它的开发。

为了降低软件开发的工作量和成本，软件产业以可复用软件构件的形式提供了部分解决方案，在开发过程中可以利用这些构件。我们所面临的挑战是，将该解决方案的一个个小的部分组装成一个连贯的企业系统，以满足复杂业务过程的需要。

软件实践鼓励从可定制的软件框架或软件包——商用成品软件（commercial off-the-shelf, COTS）解决方案或企业资源规划（enterprise resource planning, ERP）系统——来进行系统开发。然而，软件框架只能提供常规的财务、制造或人力资源系统。这些常规的解决方案必须要适应企业所期望和需要执行的特定业务过程。必须要对这些业务过程进行定义，然后开发系统模型。虽然所强调的重点由“从零开始的开发”转变到了“通过定制的软件框架进行开发”，但是在这两种情况下，软件开发的真正本质仍然是相同的。

必须为每个系统的最终解决方案创建概念性构想（模型），以确保这些构想能够满足组织的特定需要。一旦创建了这些概念性构想，就可以对软件框架的功能性进行定制，以符合概念性构想。编程任务可能有所不同，但是需求分析和系统设计活动与那些从头开发的软件类似。毕竟，一个概念性构想（模型）在许多可能的表示（实现）下是相同的。

同样重要的是，一个组织不可能找到一个软件框架来自动实现它的核心业务活动。电话公司的核心业务活动是电话技术，而不是人力资源或财务。因此，支持核心业务活动的软件很少有机会依赖软件构件或框架。此外，支持其他业务活动（如财务）的软件必须包含有针对性的或独特的解决方案，来为组织提供竞争优势。就如Szyperski（1988:5）所评述的：“标准软件包创建了一个公平的比赛场地，竞争只能来自其他领域。”

在各种情形下，开发过程都应该利用构件技术（Allen和Frost 1998；Szyperski 1998）。**构件**是软件的一个可执行单元，具有明确定义的功能（服务）及与其他构件之间的通信协议

（接口）。可以对构件进行配置来满足应用需求。最具影响力和最直接的竞争构件技术标准是Sun的J2EE/EJB和Microsoft的.NET。面向服务的体系结构（Service-Oriented Architecture，SOA）的相关技术提倡由**服务**——也就是运行的软件实例（而不是构件；必须在运行构件之前加载、安装、组合、部署和初始化）——来构建系统。

软件包、构件、服务以及类似的技术并没有改变软件生产的本质问题，尤其是需求分析与系统设计的原则和任务保持不变。能够把标准的和客户定制的构件组装成最终的软件产品，而这个“组装”过程仍然是一门艺术。正如Pressman（2005）所说：我们甚至没有软件“备件”来代替正在运行的系统中破损的构件。

1.1.2 软件开发的“意外事件”

软件开发的不变事实定义了软件生产的本质问题，并引发了软件生产中的最大挑战。极为重要的是，软件开发中的“意外事件”并不会增加软件产品的复杂性，也不会产生软件产品的可支持性的潜在缺乏。**可支持性**（*适应性*）由3个系统特征组成的集合来定义，包括软件的可理解性、可维护性和可伸缩性（可扩展性）。

软件开发的意外事件大部分可以归因于信息系统即社会系统这样的事实。它的成功或失败依赖于：人、他们对系统的接受或支持、用于开发的过程、管理措施、软件模型技术的利用等。

1.1.2.1 利益相关者

利益相关者是在软件项目中存在利害关系的人。任何受到系统影响或对系统开发产生影响的人，都是利益相关者。有两组主要的利益相关者：

- 客户（用户或系统所有者）。
- 开发者（分析员、设计员、程序员等）。

在一般的交流中，术语“用户”通常是指“客户”。我们不能否认这样的事实：术语“客户”能够更好地反映期望的含义。首先，客户是为开发付款并负责决策的人。其次，即使客户并不总是正确的，开发者也不能随意改变或拒绝客户的需求——对于任何冲突的、不可行的或非法的需求，都必须与客户再次协商。

信息系统是*社会系统*，它们是*由人*（开发者）*为人*（客户）开发的。软件项目的成功由社会因素所决定——技术则是次要的。技术低劣的系统在为客户工作并使客户获益，这样的例子有很多，反之则不然。对客户没有好处（被认为的或实际上的）的系统将会被抛弃，无论它具有多么辉煌的技术。

在典型的情况下，软件失败的主要原因可以追溯到利益相关者。在*客户*端，项目失败是因为（例如，见Pfleeger，1998）：

- 客户的需求被误解了，或者没有被完全捕获。
- 客户的需求改变得过于频繁。
- 客户没有准备为项目提供足够的资源。
- 客户不想与开发者合作。
- 客户怀有不切实际的期望。
- 系统不再对客户有利。

项目也会因为*开发者*的不胜任而失败。随着软件复杂性的增加，人们越来越认识到，开发者的技能和知识是至关重要的。良好的开发者能够交付一个可接受的解决方案；卓越的开发者能够更快、更廉价地交付一个更优越的解决方案。如同Fred Brooks（1987:13）的名言：“伟大的设计来源于伟大的设计者。”

开发者的杰出和投入是最能够促进软件质量和生产力的因素。为了确保软件产品能够成功地交付给用户，而且更重要的是使用户从中获得生产效益，软件组织必须对开发者使用正确的管理措施（Brooks 1987；Yourdon 1994），即：

- 雇佣最好的开发者。
- 为现有的开发者提供持续的培训和教育。
- 鼓励开发者之间进行信息交流和互动，使他们互相促进。
- 通过排除阻力以及将他们的精力引导到生产性工作中，来激励开发人员。
- 提供一个令人振奋的工作环境（这往往比偶尔加薪更重要）。
- 将个人目标同组织策略和目标统一起来。
- 强调团队工作。

1.1.2.2 过程

软件过程定义在软件生产和维护中所使用的活动和组织程序。过程的目标是在开发中管理和改进协作并维持团队，使团队能够向客户交付符合质量要求的产品，并在以后对产品提供适当的支持。

一个过程模型：

- 声明了所执行活动的次序。
- 详细说明要交付哪些开发的人工制品，以及什么时候交付。
- 将活动和人工制品分配给开发者。
- 提供用来监控项目进展、评估结果和规划未来项目的标准。

不同于建模和编程语言，软件过程不易被标准化。每个组织必须开发属于它自己的过程模型，或者对通用的过程模板进行定制，例如著名的Rational统一过程®（Rational Unified Process®, RUP®）模板（Kruchten 2003）。软件过程是组织的整体业务过程的一个重要组成部分，它决定了组织在市场中的独特性和竞争能力。

组织所采用的过程必须符合其开发文化、社会动态、开发者的知识和技能、管理方法、客户的期望、项目规模，甚至是应用领域的种类。由于所有这些因素都是可变的，组织可能需要将它的过程模型多样化，并且为每个软件项目创建变体。例如，依据开发者对建模方法和工具的熟悉程度，可能需要在过程中加入专题培训课程。

项目规模有可能对过程产生最大的影响。在小项目中（10个人左右的开发者），也许根本不需要正式的过程。这样的小团队可能会不拘形式地进行沟通，并且对变化做出响应。然而在大项目中，一个非正式的通信网络是不够的，因此为了控制开发而明确定义的过程是必要的。

1.1.2.2.1 迭代和增量过程

现代软件开发过程总是迭代和增量的。系统模型通过分析、设计和实现阶段而逐步完善和改造。在连续的**迭代**中增加细节，必要时还引入了变更和改进，而软件模块的**增量**版本则保持了用户的满意度，并且为尚在开发中的模块提供重要的反馈。

可执行代码的**构造**是以程序块的形式交付给客户，每次的下一个构造都是在之前构造上的增量。作为由系统必须满足的一套功能性的用户需求所决定的增量，并不会扩大项目的范围。迭代是短期的——在数周中而非数月。用户反馈是频繁的，规划是持续的，变更管理是**生命周期**中至关重要的一个方面，度量标准和风险分析的定期收集设定了连续迭代的议程。

这个迭代和增量过程有各种各样的变体，具有特殊意义的变体包括（Maciaszek和Liong 2005）：

- 螺旋模型。
- Rational统一过程（the Rational Unified Process, RUP）。