计算机科学本科核心课程教材

A Concise Introduction to Software Engineering

软件工程导论

Pankaj Jalote 著

罗 飞 邵凌霜 等译

陈世鸿 主审

清华大学出版社

Springer

计算机科学本科核心课程教材

A Concise Introduction to Software Engineering

软件工程导论

Pankaj Jalote 著

罗 飞 邵凌霜 等译

陈世鸿 主审

清华大学出版社
北京

Springer

图书在版编目（CIP）数据

软件工程导论/(美)夏洛特(Jalote,P.)著；罗飞等译.—北京：清华大学出版社，2012.1
书名原文：A Concise Introduction to Software Engineering
ISBN 978-7-302-27251-9

Ⅰ. ①软…　Ⅱ. ①夏…②罗…　Ⅲ. ①软件工程　Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2011)第 226835 号

责任编辑：龙啟铭
责任校对：时翠兰
责任印制：张雪娇

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185×230　　**印　张**：13.25　　**字　数**：269 千字
版　　次：2012 年 1 月第 1 版　　**印　次**：2012 年 1 月第1次印刷
印　　数：1～3000
定　　价：26.00 元

产品编号：035705-01

前言 FOREWORD

软件工程导论应包括大量的软件工程方面的概念、原理、技术和方法，内容繁杂，这在很大程度上决定了这门课程成为迄今为止最难教的课程之一。本人认为这是由于在教学时往往太过偏重于概念和原理等浅层知识的介绍，而忽略了这些知识的实际运用造成的，软件工程的最终目的是要运用这些知识有效地设计和开发出求解问题的好软件。

本书的目标

我认为，软件工程入门课程的目标应该是：传授给学生一些知识和技能，使得他们能够运用这些知识和技能成功地实施一个规模为几个人月的商业软件项目的开发。现在许多企业都有很多这方面的项目——能够由一个小团队经过几个月的努力而完成的项目。我还认为，在软件工程的知识海洋中，如果对各个知识点进行细致、认真的筛选，能够在一个学期的时间里，使学生们具有上述本领，而这正是本书的目标。

本书的目标是对学生介绍有限的符合下列要求的知识和练习：

- 这些有限的知识和技能能够使学生完全胜任一个较小的商业软件项目的开发；
- 提供学生所需的知识背景，使学生通过课程或自学可从事软件工程领域的高级研究。

本书的组织

在本书中，有意地删除了软件工程的高级议题，只是介绍了一些我认为是基础的或者能够满足本书目标的知识点。另外，软件项目的实施主要需要两个方面的技能——工程学和项目管理，因此，本书将重点讨论这两个方面的主要活动以及执行这些活动所需

要的知识和技术。

本书的安排方式很简单，它的每一章对应项目开发中的一个主要活动。工程学方面的主要活动包括需求分析和需求规范、体系结构设计、模块设计、编码和单元测试、以及测试；项目管理方面的主要活动包括项目计划和项目监控两个方面，而这两个活动都安排在项目计划这一章，这是因为项目的监控也需要计划。另外，在这本书中，还有一章主要用来阐明软件工程领域存在的问题，而另一章则讨论了软件过程中把每个活动联系起来的中心观点。

本书的每一章都是以对本章的介绍、目标或者读者在本章中应有的收获开始。每一章在介绍项目的开发活动时，总是首先介绍有关概念和知识，然后介绍该活动的结果或所具备的某些期望质量特性，以及一些实践方法或执行该活动所需要的技术，最后通过一些实例说明这一章的有关知识点，并为读者总结本章学习的主要内容，每一章的结尾都提供有一些自测练习。

预期读者

本书是大学本科生或研究生的软件工程入门课程，所以，它的预期读者是那些了解编程但是没有正式学习过软件工程的本科生和研究生。

另外，本书也可作为具有相似情况的专业人士（了解编程但需要软件工程系统知识）的参考书。

教学支持和辅助资源

尽管这本书是独立的，但是一些相关的教学支持和辅助资源可以通过下列网站得到。

http://www.cse.iitd.ac.in/ConciseIntroToSE

该网站上的资源包括：

- 每一章的 ppt 格式的 powerpoint 演讲稿；
- 项目开发过程中各阶段文档的模板；
- 一个项目开发文档的示例；
- 一些单元测试和审查的实践练习。

致谢

感谢我的编辑 Wayne Wheeler，这本精练的导论是他想出来的，他给我提供了如此好的机会。

我还要感谢我的妻子 Shikha 以及我的两个女儿 Sumedha 和 Sunanda，谢谢她们一直都忍耐我的坏脾气和陪我度过了这段孤独的时光。

Pankaj Jalote

目 录 CONTENTS

第1章 软件问题

给出一个需要编写软件系统来解决的问题，大多数学生认为需要编写约10 000行（比如C或Java）代码来解决，现在向一个有些编程经验的学生提问：如果让你全职工作，需要多长时间建立这个系统？

学生的回答一般为1～3个月。考虑到学生的编程技术能力，最有可能的情况是，在2个月内他们可编写一个软件并演示给教授看。用2个月时间完成软件，那么学生每个月的代码产出率约为5000行/人月。

现在让我们看看另一种情况，当我们作为客户向从事软件开发业务的公司提出同一问题时。虽然行业中没有代码产出率的标准水平，而且针对解决不同问题的代码产出率也存在很大差异，但如果对一个代码产出率为1000行/人月的程序员评价他的生产效率，说他的生产力是相当可观的（尽管在嵌入式系统中，产出率可以低至100行/人月），这是客观公正的。以这样的生产率，那么软件公司的专业团队将需10个人月的投入才能建立这个软件系统。

为什么在这两种情况下代码产出率会存在差异？为什么同一个学生，当他在学校时他的产出率可达几千行/人月，而当他在公司上班时却仅仅只有1000行/人月？

答案当然是两个不同的东西在两种不同的情况下完成。首先，学生建立系统的主要目的是出于演示，而不期望以后会使用。因为不被使用，所以软件本身没有多大意义，软件存在的缺陷和质量问题也不被人们所关注。同样，其他的质量问题，如可用性、可维护性、可移植性等也都可以忽略。

另一方面，工业强度的软件系统则是为了解决客户的特定问题而建立的，它用于客户处理某些业务。若系统出现故障则会产生巨大损害，如金融或商业损失，给用户造成不便，甚至财产和生命损失。因此，软件系统应该

保证性能健全，比如可靠性，可用性，便携性等性能。

这种满足最终用户高品质需要的软件在很大程度上会影响软件的开发方式和成本，布鲁克斯的经验法则给出建议说，工业强度的软件可能要花上10倍学生软件的投入[16]。

软件产业主要关注的是工业强度软件的开发，软件工程领域的重点也是如何建立这样的系统。也就是说，软件工程的问题域针对的是工业强度软件。在本书的余下部分，当我们使用软件这一术语时都是指工业强度的软件。在本章的剩余部分，我们将学习：

- 质量、成本和进度是（工业强度）软件项目的主要驱动力。
- 如何定义项目的成本和生产效率， 如何评定软件的质量。
- 大规模和变化是问题域的重要属性，及相应的解决办法。

1.1 成本、进度和质量

除了高品质的要求是工业强度软件与其他软件的区别，成本和进度也是这类软件的主要驱动力。在工业强度的软件领域，有三个基本的驱动力——成本、进度和质量。软件开发必须在综合考虑费用的合理性、时间的合理性、质量的良好性这三个前提下开发。一个软件项目往往由这三个参数来驱动和定义。

工业强度软件费用非常高，主要是基于这样的事实：软件开发是极度密集型的劳力。为理解成本观念，我们不妨先看看行业当前的现实状况，在行业中交付代码行（LOC）或千行代码（KLOC）是目前最常用的衡量软件规模的方法，因为人力费用是软件生产中主要的成本，故软件开发成本通常以在开发过程中每人每月工作量来计算，而生产率通常以每人每月所产出的代码行（或KLOC）来衡量。

在软件行业中，生产率幅度从每人每月数百行至超过千行范围变化。这个生产率是对整个开发周期而言的，而不仅仅只是编码任务阶段。软件公司通常向提出软件开发要求的客户报价为3000～15 000美元/人月的费用。若生产率为每人每月1000行代码，这意味着交付的每一行代码要花费3～15美元的成本！甚至一个小项目也很容易达到5万行代码规模，按这样的软件生产率，那么5万行代码的软件项目也将耗资15万～75万美元！

在许多项目中进度是另一个重要因素，商业趋势要求产品尽可能即时地投入市场，也就是说从概念到交付产品的周期要缩短。这对于软件而言，就是需要在指定的时间更快地被开发出来。不幸的是，在软件的历史案例中不乏项目延迟的例子。

显然，降低成本和缩短软件开发周期是软件工程中心目标，每人每月（KLOC）生产率可以充分反映对成本和进度的关注。如果生产率较高，显然人月的成本会较低（同样的工作，现在可以用更少的人力来完成）；同样，如果生产率较高，在更短的时间内开发出软件的可能性会提高，相比低生产率团队，一个相同规模的高生产率团队可以用更

少的时间来完成一项工作（当然项目所需的实际时间还取决于劳力的分配数量）。因此，追求更高生产率是软件工程背后的基本驱动力，也是使用不同工具和技术的主要原因。

除了成本和进度外，另外一个驱动软件工程的重要因素是质量。质量是软件的主要问题，现今的商业策略通常是围绕质量问题而制定的。不幸的是，已经出现了大量的软件不稳定性方面的案例，软件通常没有做它应该做的或做了一些不应该做的事情。显然，开发高品质的软件是软件工程的另一个基本目标。然而，成本一般很好理解，但软件质量这个概念有待进一步阐述。

软件产品质量的国际标准[55]表明，软件的质量包括 6 个主要属性，如图 1.1 所示。这些属性可以定义如下：

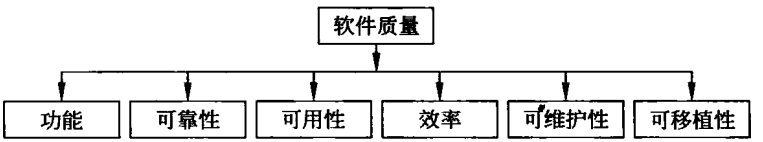

图 1.1　软件质量属性

- 功能（Functionality）　当软件被使用时提供满足需要功能的能力。
- 可靠性（Reliability）　提供无故障服务的能力。
- 可用性（Usability）　能被理解、学习、使用的能力。
- 效率（Efficiency）　对使用的资源总量所提供合适性能的能力。
- 可维护性（Maintainability）　为了更正错误、改善性能或者作出适应性调整而能被修改的能力。
- 可移植性（Portability）　无须实施额外的措施就能适应不同具体环境的能力。

由于质量具有多方面特性，不同的项目可能会强调不同属性，所以全局性的质量指标是不可能存在的。然而，尽管有许多质量属性，但可靠性被普遍认为是主要的质量标准。软件不可靠是由于软件缺陷的存在，故质量好坏的衡量是以每单位规模（一般取为千行代码，或 KLOC）软件存在的缺陷数作为主要的标准。因而提高质量就是尽可能地减少单位规模中的缺陷数量。当前好的软件工程实践已经能够将缺陷密度降低到每单位规模的缺陷数少于 1。

为确定软件产品质量，我们需要确定已交付的软件中存在的缺陷数量。而这个数字显然在交付时是不知道的，并可能永远不得而知。另一种衡量质量的方法是在交付 6 个月（或 1 年）的时间内记录发现的缺陷数，然后根据这些缺陷定义软件的质量，这意味着已交付的软件的质量在 6 个月后才能确定。而缺陷密度也可用过去类似项目的数据来估计，如果使用了类似的方法，那么可以预计目前的项目与过去的项目会有类似的缺陷密度。

应该指出的是，如果使用这种质量定义方法，那么什么是缺陷就必须明确界定。缺陷可能是一些导致软件崩溃的问题或是导致输出结果未能正确对齐的问题或是拼错一些单词等问题。缺陷的确切定义显然依赖于项目或公司开发项目所使用的标准（通常是后者）。

除了可靠性，另一个值得关注的质量属性是可维护性。一旦软件交付和使用后，它就进入维护阶段。软件没有随时间增长而损耗的物理组件，那为什么还需要维护呢？这是因为软件系统本身存在缺陷。普遍认为，现在的工艺水平是有限的，零缺陷密度的软件开发是不可能的。这些缺陷，一经发现就必须要隔离，即所谓的矫正性维护。维护也需要改变交付的软件以满足用户新的需求和适应环境，即适应性维护。在软件系统的整个生命周期中，维护成本可能远远超过原来的开发成本。对于维护成本与开发成本比例，不同的人有不同的看法，主要有三种，即 80∶20、70∶30 或 60∶40。由于维护成本高，所以软件系统的易于维护自然是人们所追求的。

1.2 规模和变更

对于我们的问题域（工业强度软件），尽管成本、进度和质量是主要的驱动力，也有一些问题域的其他特征会影响到所采用的解决方案。请关注软件的另两个特点——规模和变化。

大多数工业强度的软件系统往往是庞大而复杂的，需要成千上万行的代码。一些知名的软件产品的大小列于表 1.1 中。

表 1.1　KLOC 大小的一些知名产品

大小（KLOC）	软　件	语　言	大小（KLOC）	软　件	语　言
980	gcc	Ansic,cpp,yacc	65	Sendmail	Ansic
320	Perl	Perl,ansic,sh	30 000	RedHat Linux	Ansic,cpp
200	Openssl	Ansic,cpp,perl	40 000	Windows XP	Ansic,cpp
100	Apache	Ansic,sh			

正如所预料的那样，与开发小型系统相比，大型系统需要有不同的方法集，因为用于开发小型系统所用的方法通常不能扩展到大型系统中。下面举个例子说明这一点，考虑计算一个房间内的人数与一个国家的人口普查问题。两者本质上是计数问题，但计算一个房间里人数所使用的方法用到人口普查中就不能正常工作。人口普查将不得不使用一组不同的方法，除了计数外，人口普查问题需要相当多的管理，组织和验证。

同样，用来开发几百行程序的方案也不能被期望用到一个成千上万行的代码程序开发中，大型软件开发必须使用一组不同的方法。

任何软件项目都会涉及工程的使用和项目管理。在小型项目中，开发和管理都可以使用非正规方法。但是，对于大型项目，这二者都要严格得多，如图 1.2 所示。换句话说，若要成功地执行该系统的工程项目，必须采用适当的方法，项目的管理也必须严格以确保成本，进度和质量处于控制之中。大规模是问题域的一个关键特征，所以解决方案应采用有能力建设大型软件系统的工具和技术。

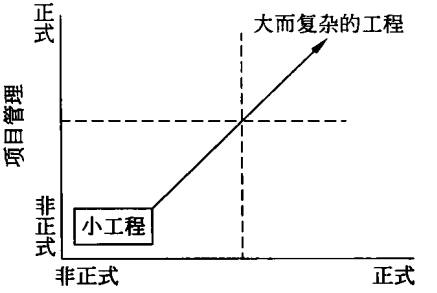

图 1.2　规模问题

变化是问题域的另一个特征，开发方案必须考虑变化。由于系统的完整需求集合一般不知道（往往不能在项目开始时知道）或者没有申明，随着开发进度和时间的推进更多的需求被提出来，这就需要纳入到正在开发的软件系统中。这种对于不断修改的需求要求开发方案包括变化，并能高效地适应变化。对于一个项目，变更请求可能会是破坏性的，如果处理不当，将会耗费多达 30%～40%的开发成本[14]。

如上所述，软件有时不得不做改变，即使它已被安装。虽然在维护过程中出现的变化可以与发生在开发过程中的变化加以区别，但这些区别是模糊的，因为从根本上来说，这两种情况下的变化是相似的，现有源代码的改变是因为需求方面的改变或者由于需要删除某些缺陷。

总体而言，随着世界变化的加快，即使正在开发中的软件也必须快速改变。因此，需求变化是问题域的一个特点。在当今世界，不能接受和适应变化的方案是没有多大用处的，它们只能解决少数拒绝变化的问题。

1.3　小结

- 软件工程的问题域是面向工业强度级软件，这种软件是为了解决某些用户组的问题，期望高质量。
- 在这个问题域中，成本、进度和质量是基本驱动力。因此，用于解决这类问题的方法和工具必须确保高生产率和高品质。
- 生产率用来衡量每单位投入资源的产出量。在软件中，产出可以用交付的代码行数来度量，人力是主要的资源，其投入可以用每人花费的时间作定量标准。因此，生产率可以按每人每月交付的代码行数来衡量。
- 软件质量包括许多属性，如功能性、可靠性、可用性、效率、可维护性和可移植性等。可靠性往往被视为主要的质量属性，软件缺陷可反映软件不可靠，所以每千行代码所含的缺陷数可反映出软件的质量。

- 这一领域中的问题常常是非常大的，且客户需求变化也很快。因此，开发工业强度软件所使用的技术应该是：有能力建立大型软件系统，并有能力处理变化。

自测练习

1. 学生型软件和工业强度软件的主要差别是什么？

2. 如果开发一个普通程序用于解决问题需要的努力为 E，可以估计为解决这一问题的工业强度软件将需要 10E 的努力。那么你认为这些额外的努力在哪些方面被消耗了呢？

3. 你会采取什么样的标准来衡量一个项目的生产率？你又将如何从这些衡量标准中确定生产率？

4. 软件质量有哪些不同的属性？对于一个会计软件，最关心的是它能确保计算不出任何错误，那么它的哪一个质量属性是我们应该最关注的？

5. 大型项目与小型项目相比，哪些是项目管理任务该做的？你将如何执行这些任务的变化？

6. 假设一个软件系统在操作过程中需要做一些改变，为什么对系统作改变所需的成本比对源代码做修改所需的成本要高？