HZ BOOKS
华章教育

大学计算机优秀教材系列

软件需求工程

第2版

毋国庆　梁正平
袁梦霆　李勇华　编著

Software Requirements Engineering

Second Edition

机械工业出版社
China Machine Press

大学计算机优秀教材系列

软件需求工程

第2版

毋国庆　梁正平
袁梦霆　李勇华　编著

Software Requirements Engineering

Second Edition

机械工业出版社
China Machine Press

图书在版编目(CIP)数据

软件需求工程/毋国庆等编著. —2版. —北京：机械工业出版社，2013.3
(大学计算机优秀教材系列)

ISBN 978-7-111-41735-4

Ⅰ. 软… Ⅱ. 毋… Ⅲ. 软件需求—教材 Ⅳ. TP311.52

中国版本图书馆CIP数据核字(2013)第045104号

本书全面、系统地介绍了软件需求工程的基本概念和原理，以及开发和管理软件需求的方法与技术，按照需求工程中开发和管理过程的顺序，结合许多经典实例，较详尽地介绍了需求开发各个阶段的任务、步骤。此外，本书还介绍了需求工程领域的一些新理论、新技术和新方法。

本书可作为高年级本科生和研究生的教材，也可供从事软件开发和研究工作的专业人员参考与自学。

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：刘立卿
北京市荣盛彩色印刷有限公司印刷
2013年5月第2版第1次印刷
185mm×260mm·16.75印张
标准书号：ISBN 978-7-111-41735-4
定 价：35.00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

目　录

CHAPTER 1

第 1 章

需求工程概述

1.1 需求工程的重要性

随着计算机应用的不断发展和深入，软件系统的日益大型化、复杂化，软件的开发成本越来越高，软件开发的风险也越来越大。Standish 集团公司的研究报告称：在美国，每年用于软件开发的费用在一千亿美元以上，其中，大型公司开发一个软件项目的平均成本为 232.2 万美元，中等大小的公司为 133.1 万美元，小型公司则为 43.4 万美元。调查显示，31%的项目在完成之前被取消，进一步研究的结果还表明：52.7%的项目实际所花费的成本为预算成本的 189%[1]。根据该公司的另一项分析，项目失败或严重超支的 8 个最重要原因中有 5 个都与需求相关：需求不完整、缺乏用户的参与、客户期望不实际、需求和需求规格说明的变更、提供许多不必要的功能[2]。

一些具体的案例令人触目惊心：伦敦股票交易项目 TAURUS，在花费了数百万英镑之后于 1993 年被取消(项目失败的总损失估计达到几亿英镑)。调查结果显示，许多问题源于未能协调那些不一致的需求[3]。Swanick 空中交通控制系统原计划在 1998 年完工，但直到 2001 年尚未交付使用，额外开支高达 1 亿英镑以上。经官方调查，发现其中的一个主要原因在于“缺乏健壮的需求规格说明导致无法继续进行系统实现”[4]。

与此同时，另外的一些调查和研究显示：一个与需求相关的错误发现和解决越迟，其修复的代价越昂贵。A. Davis 研究发现，在需求阶段检查和修复一个错误所需的费用只有编码阶段的 1/5 到 1/10，而在维护阶段做同样的工作所需付出的代价却是编码阶段的 20 倍[5]。这意味着在维护阶段修复一个错误的代价与需求阶段修复一个同样的错误的代价的比值可高达 200∶1。

诸如此类的调查研究目前已有很多。虽然项目失败涉及的原因多种多样，但正如 R. Glass 所说，“项目需求无疑是在软件项目前期造成麻烦的一个最大原因。一个又一个的研究已经发现，当项目失败时，需求问题通常正是核心问题。”[6]因此，在软件开发过程中，必须及早、有效地发现和解决与需求相关的问题。

在很长一段时间里，人们并没有充分认识到软件需求的作用，软件工程界也一直没有将需求工程作为一个独立的部分进行深入的分析和研究。直到上世纪 90 年代中期，随着

软件系统开发中出现诸多问题，人们才逐渐认识到软件需求在整个软件开发中的重要性。通过一系列关于软件需求的重要学术会议进行广泛而深入的研究和讨论，由 IEEE 创办的专门研究软件需求的国际期刊《Requirement Engineering》的出版发行标志着需求工程作为一门独立的子学科正式形成。

1.2 什么是软件需求

“需求”这个词在日常生活中经常使用。通常的需求是指人对于客观事物需要的表现，体现为愿望、意向和兴趣，因而成为行动的一种直接原因。例如，当某个顾客向裁缝师傅订做一套服装时，这位裁缝师傅首先要获得这位顾客的一些数据，如身高、胸围、腰围、臂长和样式等，然后根据这些数据制作服装。这些数据就是该顾客订做服装的具体需求。试想，如果裁缝师傅将顾客的这些具体需求弄错或者根本不知道的情况下，无论其如何精心制作，使用多好的面料，其所做的工作都将是枉然的！因为客户可能根本不能穿，或者穿着不舒适。这个例子说明，需求对最终产品能否适用是至关重要的。同理，对于软件开发来说，软件需求就是软件用户认为其所使用的软件应该具备的功能和性能。

对于软件需求的定义，不同的研究人员有不同的看法。A. Davis 认为，软件需求是从软件外部可见的、软件所具有的、满足于用户的特点、功能及属性等的集合[5]。I. Sommerville 认为，需求是问题信息和系统行为、特性、设计和实现约束的描述的集合[7]。而 M. Jackson 等人则认为，需求是客户希望在问题域内产生的效果[8,9]。在比较正式的文档中，IEEE 软件工程标准词汇表将需求定义为[10]：①用户解决问题或达到目标所需的条件或能力；②系统或系统部件要满足合同、标准、规范或其他正式规定文档所需具有的条件或能力。其中①是从用户的角度定义的，②是从软件系统的角度定义的。

关于软件需求还有其他不同的定义。产生这些不同形式的定义的原因，一是需求工程的发展过程还不太长，人们的认识还在不断深入；二是真正的“需求”实际上是在人们的脑海中形成的，很难给予准确的定义。这也是导致需求工程难度很大的原因之一。不过，根据这些定义，我们可以认为软件需求是指软件系统必须满足的所有功能、性质和限制。

对于一个软件系统，不同的人对它应具有的功能和性能会有不同的需求。例如，对于文字处理系统这一软件，A 先生打算将其用于编辑英文论文。于是他希望文字处理系统具有能简单地描述数学公式、检查英语单词和文法等功能。B 先生则打算利用文字处理系统制作贺年卡片。他希望系统具有能处理图片和进行彩色打印的功能等。C 先生则希望文字处理系统具有简单地制作中文文档和快速打印的功能等。于是，即使对于同一个文字处理系统，由于使用者的立场不同，其应具有的功能和性能也变得有所不同。因此，对于软件开发者来说，在开发一个软件系统之前，应考虑该系统的使用者有什么样的需求，该软件能解决什么问题等。否则，开发出的软件要么使用者不满意，要么根本不能使用，从而导

致软件开发费用和时间的浪费。

1.3 软件需求的分类

虽然对软件需求的定义有多种形式，但从软件用户多年来对软件的实际需求来看，软件的需求(或用户需求)通常可以大致分类如下：

- 目标需求：反映组织机构或客户对系统和产品提出的高层次的目标要求，其限定了项目的范围和项目应达到的目标。
- 业务需求：主要描述软件系统必须完成的任务、实际业务或工作流程等。软件开发人员通常可从业务需求进一步细化出具体的功能需求和非功能需求。
- 功能需求：指开发人员必须实现的软件功能或软件系统应具有的外部行为。
- 性能需求：指实现的软件系统功能应达到的技术指标，如计算效率和精度、可靠性、可维护性和可扩展性等。
- 约束与限制：指软件开发人员在设计和实现软件系统时的限制，如开发语言、使用的数据库等。

在这些需求中，功能需求描述系统做什么，由性能需求和约束与限制构成的非功能需求则为实现这些功能需求设定约束和限制。软件需求间的关系可分层次地表示，如图 1-1 所示。

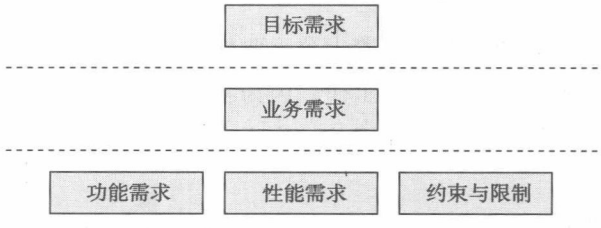

图 1-1 软件需求间的层次关系

由以上的这些需求就可构成软件需求规格说明。下面我们通过与文字处理系统相关的部分需求来说明需求的分类。

- 目标需求：用户使用系统能有效地纠正文档中的拼写错误，系统能满足用户的业务要求以及提高用户的工作效率。
- 业务需求：当找到文档中的拼写错误时，通过可供选择的单词表，选择单词表中的一个单词后，再替换掉原来的单词。
- 功能需求：查找文档中的单词，并高亮度地显示出错的单词。用对话框显示可供选择的单词表，实现整个文档范围内的替换。

- 性能需求：检查单词的速度快，准确率要求达到99%，系统的有效性和可靠性要高等。
- 约束与限制：文件内部格式要与Word系统一致。开发平台为Linux系统，使用C语言等。

1.4 需求规格说明

软件需求规格说明亦称软件需求规约或功能规格说明，可以说是需求工程最终产生的结果。所谓需求规格说明是软件所应满足的全部需求，并可用文档的方式完整和精确地陈述这些需求。需求规格说明是项目相关人员对将要开发的软件系统所达成的共识，是进行系统设计、实现、测试和验收的基本依据，也是整个软件开发过程中最重要的文档[11]。需求规格说明同时代表了权限的移交点：客户对需求规格说明的说明内容拥有最终发言权，而开发人员则需根据软件需求规格说明实施软件系统的开发。因此，最终开发出的软件系统是否能真实、全面地满足客户的要求，取决于需求规格说明是否真实、完整和一致地反映客户的真正意图。

需求规格说明应精确地描述一个软件系统必须提供的功能和性能，以及所要考虑的约束条件与限制。因此，需求规格说明也可以说是集成了1.3节中所定义的所有软件需求，并使用某种描述语言如自然语言按照规定的书写格式编写的文档。关于需求规格说明的模板和具体内容，将在后面给予详细说明。

需求规格说明在软件系统开发中起着十分重要的作用，但要把用自然语言表达的需求完整无误地表达出来并不是一件容易的工作。因为各人的理解不同，即便是同一个人，他在不同的时间也可能产生不同的理解。因此，作为一个质量较高的需求规格说明，通常应满足如下的特征[12]。

完整性。每一项需求必须将所要实现的功能描述清楚，以便开发人员获得设计和实现这些功能所需的必要信息。

正确性。每一项需求都必须准确地陈述其要开发的功能。做出正确判断的参考是需求的来源，如用户或高层的系统需求规格说明。如果软件需求与对应的系统需求相抵触，则是不正确的。只有用户才能确定需求的正确性。这就是一定要有用户的积极参与的原因。

可行性。每一项需求都必须在已知系统和环境的权能和限制范围内是可以实施的。为避免不可能实现的需求，最好在获取需求(或收集需求)的过程中，始终有一位软件开发小组的成员与需求分析人员或考虑市场的人员在一起，由他负责检查技术的可行性。

必要性。每一项需求都应把客户真正需要的和最终系统所遵从的标准记录下来。“必要性”也可以理解为每项需求都是用来授权你编写文档的“根源”。要使每项需求都能回溯至某项客户的输入，如使用实例或别的来源。

划分优先级。给每项需求、特性或使用实例分配一个实施优先级，以指明它在特定产品中所占的分量。如果把所有的需求都看作同样重要，那么项目管理者在开发、节省预算或调度中就丧失了控制自由度。

无二义性。对所有需求说明都只能有一个明确统一的解释，由于自然语言极易导致二义性，所以尽量把每项需求用简洁明了的语言表达出来。避免二义性的有效方法包括对需求文档的正规审查，编写测试用例，开发原型以及设计特定的方案脚本。

可验证性。检查每项需求是否能通过设计测试用例或其他的验证方法。如用演示、检测等来确定产品是否确实按需求实现了。如果需求不可验证，则确定其实施是否正确就成为主观臆断，而非客观分析了。一份前后矛盾、不可行或有二义性的需求也是不可验证的。

1.5 需求工程定义

需求工程是指应用工程化的方法、技术和规格来开发和管理软件的需求。需求工程的目标就是要获取高质量的软件需求。与软件工程中传统的需求分析概念相比，需求工程突出了工程化的原则，强调以系统化、条理化、可重复化的方法和技术进行与软件需求相关的活动，从而有利于提高所有与软件需求相关的活动及其过程的可管理性，降低需求开发和管理的难度和成本。

由于需求工程诞生的时间相对短暂，或者说它还是一个新兴的子学科，因此，对于需求工程来说，并不存在一个得到普遍承认的精确定义。许多不同的研究人员和组织依据各自的研究目的，分别从不同的侧面提出了各不相同但本质上近似的定义。例如 Davis A. M.[13]把需求工程定义为“直到(但不包括)把软件分解为实际架构组建之前的所有活动”，即软件设计之前的一切活动。该定义虽然没有详细说明需求工程是什么，但给出了需求工程的范围。英国的 Bray I. K.[9]则认为，需求工程是指：对问题域及需求做调查研究和描述，设计满足那些需求的解系统的特性，并用文档给予说明。这个定义明确指出了需求工程的任务就是获取、分析和表达软件的需求。

从各种不同形式的需求工程定义可以看出，需求工程应该是由一系列与软件需求相关的活动组成的。如果从这些活动考虑需求工程定义的话，需求工程可认为是由需求的开发活动和需求的管理活动组成的。因此，需求工程的任务可概要表示如下：

1）确定待开发的软件系统的用户类，并获取他们的需求信息。

2）分析用户的需求信息，并按软件需求的类型对这些需求信息进行分类，同时，过滤掉不是需求的信息。

3）根据软件需求信息建立软件系统的逻辑模型或需求模型，并确定非功能需求和约束条件及限制。

4）根据收集的需求信息和逻辑模型编写需求规格说明及其文档。

5）评审需求规格说明。

6）当需求发生变更时，对需求规格说明及需求变更实施进行管理。

1.6 其他一些基本概念

为了便于本书以后的阐述，读者有必要理解以下几个基本概念：

用户(user)

- 利用计算机系统所提供的服务的人。
- 直接操作计算机系统的人，简单地说，就是直接使用软件系统的人。

客户(customer)

- 掌握经费的人，通常有权决定软件需求。客户可以是用户，也可以不是用户。
- 正式接收新开发或修改后的硬件和软件系统的某个人或组织。

简单地说，客户就是为开发软件而提供经费的人。当客户和用户由不同的人组成时，由于身份不同，对软件系统的看法和要求也会不同。例如，用户希望软件系统易于使用，而客户往往希望软件的开发成本较小，并可获得较高的利润。显然这会导致用户和客户对软件产生不同的需求。

软件开发人员(supplier)

为客户开发软件系统的人。当软件系统是由客户委托开发时，客户与软件开发人员属于不同的组织。如果是组织内自行开发软件系统，客户与软件开发人员应属同一组织。

项目相关人员(stakeholder)

与提出和定义软件需求相关的人，包括所有的用户、客户和软件开发人员。这些人都是软件需求的来源，只是他们站在不同的立场看待将要开发的软件系统。

为便于说明，本节以后在不特殊指明的情况下，将把用户和客户统称为用户，意指直接或间接从软件系统获得利益的个人或组织。软件开发人员在需求工程中则主要是指系统分析人员。

CHAPTER 2

第 2 章 软件工程与需求工程

2.1 软件工程

软件工程是指用工程方法开发和维护软件的过程和有关技术。软件工程起因于上世纪60年代后期出现的“软件危机”。所谓“软件危机”实质上是指人们难以控制软件的开发和维护，其具体表现为：大型软件系统十分复杂，很难理解和维护；软件开发周期过长；大型软件系统的可靠性差；软件费用往往超出预算。面对“软件危机”，人们通过调查软件系统开发的实际情况，逐步认识到软件的开发和维护有必要采用工程化的方法，于是软件工程在1968年应运而生。

软件工程的适用对象主要是大型软件。软件工程研究的基本内容包括软件开发过程、软件开发和维护的方法与技术、软件开发和维护工具系统、质量评价和质量保证、软件管理和软件开发环境等。对于软件工程来说，从方法论的角度研究软件的开发过程是十分重要的工作。为说明需求工程与软件工程的关系，本章先介绍几个有代表性的软件开发过程模型，然后结合软件的开发过程来说明需求工程在软件工程中的地位及重要性。

2.2 软件开发过程模型

软件开发过程模型是为获得高质量的软件系统所需完成的一系列任务的框架。它规定了完成各项任务的工作步骤。在软件工程的初期，软件生命期这一概念被提出。这是用标准的形式表示和定义了软件生存过程。所谓软件生命期是指软件从软件计划开始，经历需求分析和定义、设计、编码、测试、运行、维护直到废止为止的期间。由于软件生命期包括了软件的整个生存过程，与软件开发相关的企业和开发组织等都把软件生命期视为软件开发过程模型的依据，工程管理也以该模型为实施依据。当然，这也是模仿其他行业如机器制造业和建筑业等而得到的过程模型。

2.2.1 瀑布式模型

瀑布式开发模型是最早的、依据软件生命期而提出的软件开发模型，亦称软件生命期

模型。瀑布式开发模型如图 2-1 所示，其中软件的开发过程分为六个阶段，每个阶段都有明确的分工和任务，并产生一定的书面结果。各阶段之间是紧密相关的，后一阶段的工作依据前一阶段的工作结果而开展。

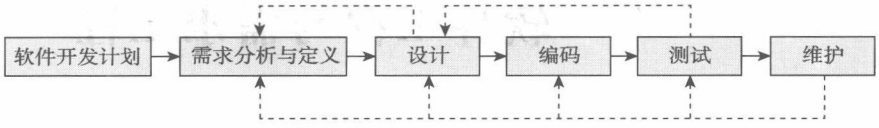

图 2-1 瀑布式开发模型

各个阶段的基本任务如下：

1)软件开发计划：确定软件开发项目必须完成的总目标，论证项目的可行性，给出软件的工作范围，估算所需资源、工作量、费用，安排开发进度，并产生计划任务书及可行性报告等。

2)需求分析与定义：软件开发人员与用户一起理解和表达用户需求，并产生需求规格说明。

3)设计：设计阶段又分为总体设计和详细设计两个子阶段。总体设计阶段就是根据软件需求规格说明建立软件系统的结构，描述软件系统的具体功能和接口；详细设计阶段产生编码阶段所需的一系列模块设计规格说明。

4)编码：根据设计要求，使用某种程序设计语言编写程序。

5)测试：对软件系统进行检查和测试，及时地发现和纠正软件系统中的故障和逻辑错误，并产生测试报告等。测试也可分为单元测试和综合测试。

6)维护：通过各种必要的维护活动保证软件系统正常运行，并能持久地满足用户的需求。

瀑布式模型的特点有很多，如：阶段间具有顺序性和依赖性，各阶段必须完成规定的文档，从而在审查文档的基础上保证软件的质量等。值得注意的是，瀑布式模型只是提供了一个完成软件开发和维护任务的指导性框架，缺乏具体的实施方法和技术，它也并非以线性方式进行，因为在实际的软件开发工作中还存在着反复。例如，如果在设计中发现需求比较含糊，则需回到需求分析与定义阶段重新进行处理。又如，如果在测试中发现设计存在问题，则需返回设计阶段进行修正，然后再进行编码和测试等(见图 2-1)。

瀑布式模型在 20 世纪 80 年代之前一直是唯一广泛采用的生命期模型，现在仍然是软件工程中应用得最广泛的模型。然而，传统的瀑布式模型也存在诸多问题：

1)在实际开发工作中，用户不可能一开始就使自己的需求很清晰，通常是在开发过程中逐渐完善的。另外，当某些需求比较含糊时，如系统的友好用户界面等，将导致软件开发人员不一定能开发出使用户满意的软件系统，再加上在开发过程中用户的需求可能发生变化，也导致软件开发工作按瀑布式模型的步骤又得从头开始，这显然是不合

理的。

2)由于模型各阶段的界线划分清晰，也比较独立，而且参加人员和开发人员也都相对独立，因此在阶段间移交信息(文档)的过程中，由于个人的理解不同，或者当事人不在时，容易产生误解。这容易导致开发出的软件系统与用户需求产生偏差。

3)用户的参与程度不够。因为软件的运行版本要等到测试后才能出现，用户也只能在需求分析与定义阶段和测试阶段的后期参与到开发工作中，因此，用户在相当长的一段时间内没有参与到软件开发中。

针对瀑布式模型的特点和存在的问题，人们又在软件生命期模型的基础上，推出了一些其他开发模型。

2.2.2 快速原型模型

快速原型模型是针对瀑布式模型存在的不足而提出的改进模型。所谓“原型”通常是指一种与原物的结构、大小和一般功能接近的形式或设计，如航天飞船模型和建筑物模型等。软件原型是指待开发的软件系统的部分实现。而快速原型是在完成最终可运行软件系统之前快速建立实验性的、可在计算机上运行的程序(原型)，然后给予评价的过程。快速原型模型的基本思想是快速建立一个实现了若干功能(不要求完全)的可运行模型来启发、揭示和不断完善用户需求，直到满足用户的全部需求为止。图 2-2 表示了快速原型模型的基本过程。

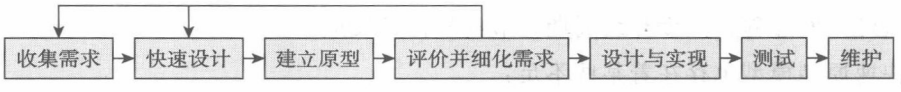

图 2-2 快速原型开发模型

从图 2-2 可以看出，快速原型模型首先是快速建立一个能反映用户主要需求的原型系统，然后提供给用户在计算机上使用。用户在试用原型后会提出许多修改意见，开发人员根据用户提出的意见快速修改原型系统，然后再交给用户试用。重复上述过程，直到用户认为这个原型系统能达到他们的要求为止。开发人员便根据原型系统编写需求规格说明，因此，根据这份需求规格说明开发出的软件应能满足用户的真正需求。

对于开发出的原型来说，由于原型的用途是获知用户的真正需求，一旦需求确定了，原型将被抛弃，因此，原型系统的内部结构并不重要，重要的是，必须迅速地构建原型，然后根据用户意见迅速地修改原型。Unix Shell 和超文本都是广泛使用的快速原型语言。最近的趋势是使用第四代语言(4GL)来构建快速原型。

当快速原型的某个部分是利用软件工具由计算机自动生成的时候，也可以将这部分用到最终的软件产品中。例如，用户界面通常是快速原型的一个关键部分，当使用屏幕生成程序和报表生成程序自动生成用户界面时，实际上可以把得到的用户界面用于最终的软件

产品中。

使用快速原型模型的目的是：

- 明确并完善需求。作为一种需求工具，原型初步实现系统的一部分。用户对原型的评价可以指出需求中的许多问题，这样在真正开发系统之前可以用最低的费用来解决这些问题。
- 探索设计选择方案。作为一种设计工具，原型可以探索不同的界面技术，使系统达到最佳的状态，用于评价以后的技术方案。
- 可以发展为最终的产品。作为一种构造工具，原型是产品最初若干基本功能的实现。通过一系列小规模的开发和完善，逐步完成整个产品的开发。

快速原型模型的特点是：

- 开发过程虽然仍与瀑布式模型相同，但在具体实施细节方面有所不同，从而弥补了瀑布式模型的一些不足，如用户参与程度不够等。
- 通过原型系统能使用户的需求明确化，也可减少用户需求的遗漏或用户频繁修改需求的可能性，特别是在软件开发的后期阶段。
- 用户可以充分地参与到软件开发中。因为通过试用原型系统，用户能及时提出一些反馈意见和建议，从而使开发人员在开发工作中，如设计与实现阶段，能尽量减少错误。
- 快速原型模型的本质是“快速”。开发人员应尽可能快地构造原型系统，从而能加速软件开发过程，减少开发周期，节约软件开发成本。

但快速原型模型也存在着某些不足：

- 用户易于视原型为正式产品。用户看到的是软件系统的可执行版本，他们并不知道某些原型是临时拼凑出来的，并且也不知道为尽快让系统运行而没有保证软件系统的总体质量和可维护性。故当他们知道软件系统要重建时，往往认为只要稍作修改就可获得最终的软件系统。
- 快速原型系统对于软件系统的开发环境要求较多，这也在一定程度上影响了其使用的范围和实用价值。

2.2.3 渐增式模型

渐增式模型亦称增量模型。渐增式模型的基本思想是从核心功能开始，通过不断地改进和扩充，使得软件系统能适应用户需求的变动和扩充，从而获得柔性较高的软件系统。

图 2-3 表示了渐增式模型的开发过程。渐增式模型表明，必须在实现各个构件之前就全部完成需求分析和概要设计工作。

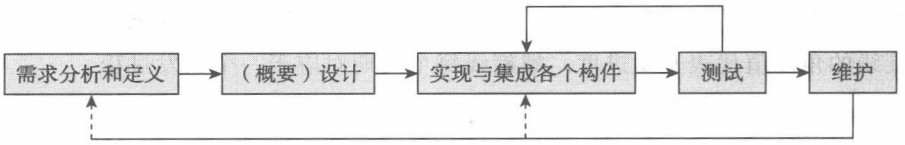

图 2-3 渐增式模型的开发过程

在尽早向用户提供可运行的软件系统方面，渐增式模型类似快速原型模型。但快速原型模型在考虑优先开发什么功能或部分系统方面与渐增式模型有所不同。快速原型模型主要根据用户需求较为模糊的部分优先开发原型；而渐增式模型则从功能明确、设计技术上不确定因素很少的核心功能优先开发，并且分批地逐步向用户提交产品。

渐增式模型的特点是：

- 能在短时间向用户提交可完成部分功能的产品。
- 能逐步增强产品功能，以使用户有较充裕的时间学习和适应新的软件系统。

但是渐增式模型存在如下一些不足：

- 在把每个新增的构件或功能集成到现有的软件系统中时，必须不破坏该软件系统。
- 在设计软件系统的体系结构时，要充分考虑其开放性，加入新构件的过程必须简单和方便。例如，在使用渐增式开发模型开发字处理系统时，首先可实现基本的文件处理、编辑和文档生成功能，然后再实现拼写与语法检查功能，最后完成高级的页面排版功能等。

2.2.4 螺旋式模型

在制定软件开发计划时，软件开发人员要准确回答出诸如软件的需求是什么、需要投入多少资源(人力，经费)以及如何安排开发进度等一系列问题是十分不容易的，但他们又回避不了这些问题，于是只能凭经验和以往的数据进行估算，这便带来一定的风险。实践表明，项目的规模越大，问题越复杂，资源、进度、成本等因素的不确定性越大，承担项目所冒的风险也就越大。软件风险普遍存在于任何软件开发项目，对不同的项目其差别只是风险有大有小而已。因此，在软件的开发过程中应该考虑风险问题。螺旋式模型的基本思想是：将瀑布式模型与快速原型模型结合到一起，加上风险分析。理解这种模型的一个简便方法是把它看作在每个阶段之前都增加风险分析。

螺旋式模型的开发过程为依螺旋方式旋转，其 4 个方面的活动分布在简称为坐标系的 4 个象限上，这些活动如图 2-4 所示。

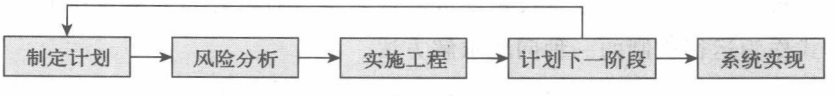

图 2-4 螺旋式模型的开发过程