内 容 简 介

本书由工业和信息化部教育与考试中心组织编写，系统地介绍了系统分析师必须掌握的理论知识和应用技术，是系统分析师考试应试者必备教材。

全书分三篇内容，共计22章。第一篇为基础知识，主要内容包括绪论、数学与工程基础、计算机系统、计算机网络与分布式系统、数据库系统、企业信息化、软件工程、项目管理和信息安全等系统分析与设计的基本知识。第二篇为关键技术，主要内容包括系统规划与分析、软件需求工程、软件架构设计、系统设计、软件实现与测试、系统运行与维护等系统分析与设计技术。第三篇为案例实践，主要内容包括Web应用系统、嵌入式系统、移动应用系统、大数据处理系统、微服务系统、信息物理系统等代表性信息系统的分析与设计方法以及系统分析师论文写作要点。

本书可以作为系统分析师的工作手册，也可以作为系统分析与设计技术的培训和辅导教材，还可以作为计算机专业教师的教学参考用书。

图书在版编目 (CIP) 数据

系统分析师教程 / 宋胜利主编 . -- 2 版 . -- 北京 : 清华大学出版社 , 2024. 10.
(全国计算机技术与软件专业技术资格（水平）考试用书). -- ISBN 978-7-302-66916-6

Ⅰ. TP311.5

中国国家版本馆 CIP 数据核字第 2024AG1075 号

责任编辑：杨如林　邓甄臻
封面设计：杨玉兰
版式设计：方加青
责任校对：徐俊伟
责任印制：刘　菲

出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-83470000　　**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：三河市君旺印务有限公司
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　张：**48.5　　**插　页：**1　　**字　数：**1178 千字
版　　次：2010 年 2 月第 1 版　2024 年 10 月第 2 版　　**印　次：**2024 年 10 月第 1 次印刷
定　　价：179.00 元

产品编号：091905-01

前　言

信息化已经全面融入社会生产和人类生活的各个环节，信息化的核心是信息资源开发、信息技术应用和信息系统建设。系统分析师是负责分析、设计和指导开发企业或组织计算机信息系统的高级复合型人才，其知识水平和业务工作能力决定了信息系统的建设质量和应用效果。一名合格的系统分析师不但应具备扎实的信息技术知识，掌握计算机技术的发展方向，还必须具备管理学的知识；不但要具备较强的系统观点和逻辑分析能力，能够从复杂的事物中抽象出系统模型，还要具备较好的口头和书面表达能力，较强的组织能力和人际交往能力；不但要具备扎实的理论基础，还要具备丰富的项目实践经验。

全国计算机技术与软件专业技术资格（水平）考试是由国家人力资源和社会保障部、工业和信息化部领导的国家级考试，其目的是科学、公正地对全国计算机与软件专业技术人员进行职业资格、专业技术资格认定和专业技术水平测试。考试实施多年来在社会上产生了重大的影响，广泛调动了专业技术人员工作和学习的积极性，为选拔高素质的专业技术人员起到了积极的促进和推动作用。为了培养更多的系统分析与设计专业人才，帮助广大考生顺利通过系统分析师考试，工业和信息化部教育与考试中心组织专家修订了系统分析师考试大纲和系统分析师教程，作为系统分析师考试的必备教材。

本书围绕系统分析师的工作职责和任务展开，系统地介绍了系统分析师必须掌握的理论知识和应用技术。本书内容既是对系统分析师考试的总体纲领性的要求，也是系统分析师职业生涯所必需的知识与技能。准备参加考试的人员可通过阅读本书明确考试大纲规定的知识、技术和能力要求，把握考试的重点和难点。

本书由宋胜利主编，鲍亮、王高亮、刘伟副主编。全书共计 22 章，第 1、2、10、16、22 章由宋胜利编写，第 3 章由王亚平编写，第 4 章由张淑平编写，第 5、8、11 章由刘伟编写，第 6、12、18 章由蔺一帅编写，第 7、9、10 章由张琛编写，第 13、19、20 章由鲍亮编写，第 14 章由王黎明编写，第 15、21 章由王高亮编写，第 17 章由叶宏编写，全书由宋胜利统稿。

本书在编写过程中，参考并引用了许多相关的书籍、资料和互联网发布的信息，编者在此对这些文献的作者表示感谢。同时感谢清华大学出版社在本书出版过程中所给予的支持和帮助。

因编者水平有限，且本书涉及的知识点较多，书中难免存在错漏和不妥之处，诚恳期望各位专家和读者指正，在此表示由衷感谢。

编者

2024 年 2 月

目　录

第一篇　基础知识

第二篇 关键技术

第三篇 案例实践

第 7 章　软件工程

软件工程是指应用计算机科学、数学及管理科学等原理，以工程化的原则和方法来解决软件问题的工程，其目的是提高软件生产率、提高软件质量、降低软件成本。IEEE 对软件工程的定义是：将系统的、规范的、可度量的工程化方法应用于软件开发、运行和维护的全过程及上述方法的研究。

软件工程由方法、工具和过程三个部分组成。软件工程方法是完成软件工程项目的技术手段，它支持整个软件生命周期；软件工程使用的工具是人们在开发软件的活动中智力和体力的扩展与延伸，它自动或半自动地支持软件的开发和管理，支持各种软件文档的生成；软件工程中的过程贯穿软件开发的各个环节，管理人员在软件工程过程中，要对软件开发的质量、进度、成本进行评估、管理和控制，包括人员组织、计划跟踪与控制、成本估算、质量保证和配置管理等。

7.1　软件生命周期

软件产品从形成概念开始，经过开发、使用和维护，直到最后退役的全过程称为软件生命周期或生存周期。一个完整的软件生命周期是以需求为出发点，从提出软件开发计划的那一刻开始，直到软件在实际应用中完全报废为止。软件生命周期的提出是为了更好地管理、维护和升级软件，其中更大的意义在于管理软件开发的步骤和方法。

目前，划分软件生命周期阶段的方法有许多种，软件规模、种类、开发方式和开发环境，以及开发时使用的方法论都影响软件生命周期阶段的划分。对软件生命周期各阶段进行划分，必须遵循一条基本的原则，那就是各阶段的任务应尽可能地相对独立，同一阶段各项任务的性质应尽可能地相同，从而降低每个阶段任务的复杂度，减少不同阶段任务之间的联系，有利于软件工程的组织和管理。

1. 软件生命周期过程

在国家标准《系统与软件工程 软件生存周期过程（GB/T 8566—2022）》中，将软件生存周期中可能执行的活动分为 5 个基本过程、9 个支持过程和 7 个组织过程。每个生存周期过程划分为一组活动，每一项活动进一步划分为一组任务。

（1）基本过程。基本过程供各主要参与方在软件生存周期期间使用，主要参与方是发起或完成软件产品开发、运行或维护的组织。基本过程分为获取过程、供应过程、开发过程、运作过程和维护过程。获取过程是指为获取系统、软件产品或软件服务的组织（即需方）而定义的活动；供应过程是指为向需方提供系统、软件产品或软件服务的组织（即供方）而定义的活动；开发过程是指为定义并开发软件产品的组织（即开发方）而定义的活动，包括需求分析、设计编码、集成、测试以及与软件产品有关的安装和验收等活动；运作过程是指为在规定的环境中为其用户提供运行计算机系统服务的组织（即操作方）而定义的活动；维护过程是指为提供维

护软件产品服务的组织（即维护方）而定义的活动，也就是对软件的修改进行管理，使它保持合适的运行状态，包括软件产品的迁移和退役。

（2）支持过程。支持过程作为一个有机组成部分支持其他过程，以便取得软件项目的成功，并提高软件项目的质量。支持过程包括文档编制过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程、审核过程、问题解决过程和易用性过程。根据需要，支持过程可被其他过程应用和执行。文档编制过程是指为记录生存周期过程所产生的信息而定义的活动；配置管理过程是指定义配置管理活动；质量保证过程是指为客观地保证软件产品和过程符合规定的需求以及已建立的计划而定义的活动，联合评审、审核、验证和确认可以作为质量保证技术使用；验证过程是指根据软件项目需求，按不同深度为需方、供方或某独立方验证软件产品而定义的活动；确认过程是指为需方、供方或某独立方确认软件项目的软件产品而定义的活动；联合评审过程是指为评价一项活动的状态和产品而定义的活动，该过程可由任何两方应用，其中一方（评审方）以联合讨论会的形式评审另一方（被评审方）；审核过程是指为判定符合需求、计划和合同而定义的活动，该过程可由任何两方应用，其中一方（评审方）审核另一方（被评审方）的软件产品或活动；问题解决过程是指为分析和解决问题（包括不合格）而定义的活动，不论问题的性质或来源如何，它们都是在实施开发、运作、维护或其他过程期间暴露出来的；易用性过程是指为易用性专业人员而定义的活动。

（3）组织过程。组织过程可被某个组织用来建立和实现由相关的生存周期过程和人员组成的基础结构，并不断改进这种结构和过程。应用这些过程通常超出特定的项目和合同的范围，但是，这些特定项目和合同的经验教训有助于改善组织状况。组织过程包括管理过程、基础设施过程、改进过程、人力资源过程、资产管理过程、重用大纲管理过程和领域工程过程。管理过程是指为生存周期中的管理（包括项目管理）而定义的基本活动；基础设施过程是指为建立生存周期过程基础结构而定义的基本活动；改进过程是指为某一组织建立、测量、控制和改进其生存周期过程而定义的需要执行的基本活动；人力资源过程是指为给组织或项目提供拥有技能和知识的员工而定义的活动；资产管理过程是指为组织的资产管理人员而定义的活动；重用大纲管理过程是指为组织的复用大纲主管而定义的活动；领域工程过程是指为领域模型、领域架构的确定及该领域资产的开发和维护而定义的活动。

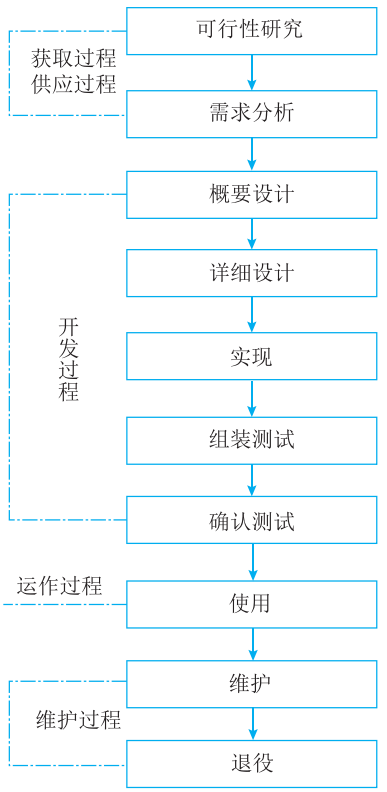

图 7-1　软件生命周期与 5 个基本过程的对应关系

2. 软件生命周期各阶段的任务

根据国家标准 GB/T 8566—2022，软件生命周期可以划分为可行性研究、需求分析、概要设计、详细设计、实现、组装测试、确认测试、使用、维护、退役 10 个阶段，各自分别对应于软件生存周期的基本过程，如图 7-1 所示。

（1）可行性研究和项目开发计划。通过分析用户提出的软件开发要求，确定软件项目的性质、目标和规模，得出可行性研究报告。如果可行性研究的结果是可行的，就要制定详细的项目开发计划。这两个活动通常被整合在一起进行，在实际工作中通常把它们归类到同一个阶段中。

（2）需求分析。需求分析工作是软件生命周期中重要的一步，也是决定性的一步。只有通过需求分析，才能把软件功能和性能的总体概念描述为具体的软件需求规格说明，从而奠定软件开发的基础。

（3）概要设计。根据软件需求规格说明建立软件系统的总体结构和模块间的关系，定义各功能模块接口，设计全局数据库或数据结构，规定设计约束，制定组装测试计划。

（4）详细设计。将各模块要实现的功能用相应的设计工具详细描述出来。

（5）实现。写出正确的、易理解的和易维护的程序模块。程序员根据详细设计文档将详细设计转化为程序，完成单元测试。

（6）组装测试（集成测试）。将经过单元测试的模块逐步进行组装和测试。

（7）确认测试。测试系统是否达到了系统需求，按照规格说明书的规定，由用户（或在用户积极参与下）对系统进行验收。必要时，还可以再通过现场测试或并行运行等方法对系统进行进一步的测试。

（8）使用。将软件安装在用户确定的运行环境中，测试通过后移交用户使用。在软件的使用过程中，客户和维护人员必须认真收集发现的软件错误，定期或阶段性地撰写软件问题报告和软件修改报告。

（9）维护。通过各种必要的维护活动使系统持久地满足用户的需要。

（10）退役。终止对软件产品的支持，软件停止使用。

7.2 软件开发方法与模型

软件开发方法是指软件开发过程所遵循的办法和步骤，从不同的角度可以对软件开发方法进行不同的分类。从开发风范上看，可分为自顶向下的开发方法与自底向上的开发方法。在实际软件开发中，经常是两种方法结合使用，只不过是应用于开发的不同阶段和以何者为主而已。

软件开发模型则给出了软件开发活动各阶段之间的关系，它是软件开发过程的概括，是软件工程的重要内容。软件开发模型为软件工程管理提供了里程碑和进度表，为软件开发过程提供了原则和方法。

7.2.1 传统软件开发模型

软件开发模型大体上可分为三种类型：第一种是以软件需求完全确定为前提的瀑布模型；第二种是在软件开发初始阶段只能提供基本需求时采用的迭代式或渐进式开发模型，例如，喷泉模型、螺旋模型、统一开发过程和敏捷方法等；第三种是以形式化开发方法为基础的变换模型。

1. 瀑布模型

软件开发生命周期（Software Development Life Cycle，SDLC）模型的发展体现了软件工程理论的发展。在最早的时候，软件的开发过程处于无序和混乱的状况，人们为了能够控制软件的开发过程，就将软件开发严格区分为多个不同的阶段，并在阶段之间加上严格的审查，这就是瀑布模型产生的起因。

瀑布模型是一种严格定义方法，它将软件开发的过程分为软件计划、需求分析、软件设计、程序编码、软件测试和运行维护 6 个阶段，形如瀑布流水，最终得到软件产品，如图 7-2 所示。

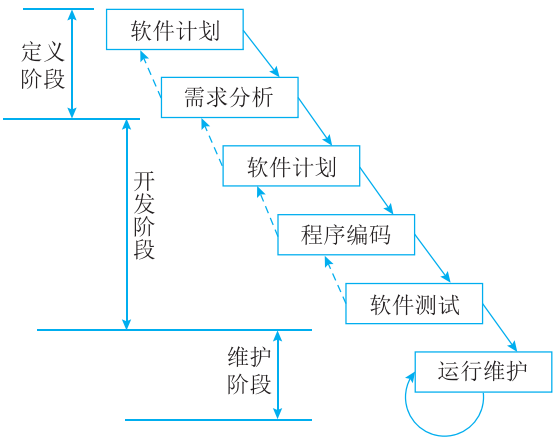

图 7-2　瀑布模型

瀑布模型是一个线性顺序模型，支持线性开发。它假设当线性序列完成之后就能交付一个完善的系统，并没有考虑软件的演化特征。其优点是强调开发的阶段性、早期计划和需求调查以及产品测试，以这样严格的方式构造软件，开发人员很清楚每一步应该做什么，有利于项目管理。

然而，在瀑布模型中，依赖于早期进行的需求调查，不能适应需求的变化；由于是单一流程，开发中的经验教训不能反馈应用于本产品的过程；风险往往推迟至后期的开发阶段才显露出来，从而失去了及早纠正的机会。在瀑布模型中，需求或设计中的错误往往只有到了项目后期才能够被发现，对于项目风险的控制能力较弱，从而导致项目常常延期完成，开发费用超出预算。

2. 演化模型

演化模型主要针对事先不能完整定义需求的软件开发，是在快速开发一个原型的基础上，根据用户在调用原型的过程中提出的反馈意见和建议，对原型进行改进，获得原型的新版本，重复这一过程，直到演化成最终的软件产品。

演化模型的主要优点是，任何功能一经开发就能进入测试，以便验证是否符合产品需求，可以帮助引导出高质量的产品要求。其主要缺点是，如果不加控制地让用户接触开发中尚未稳定的功能，可能对开发人员及用户都会产生负面的影响。

3. 螺旋模型

螺旋模型是瀑布模型与演化模型相结合，并加入两者所忽略的风险分析所建立的一种软件开发模型。螺旋模型是一种演化软件过程模型，它将原型实现的迭代特征与线性顺序模型中的控制和系统化的方面结合起来，使软件的增量版本的快速开发成为可能。在螺旋模型中，软件开发是一系列的增量发布。

螺旋模型沿着螺旋线进行若干次迭代，每次迭代都包括制订计划、风险分析、实施工程和客户评估 4 个方面的工作。螺旋模型强调风险分析，使得开发人员和用户对每个演化层出现的风险有所了解，继而做出应有的反应。因此，它特别适用于庞大、复杂并具有高风险的系统。

与瀑布模型相比，螺旋模型支持用户需求的动态变化，为用户参与软件开发的所有关键决策提供了方便，有助于提高软件的适应能力，并且为项目管理人员及时调整管理决策提供了便利，从而降低了软件开发的风险。在使用螺旋模型进行软件开发时，需要开发人员具有相当丰富的风险评估经验和专门知识。另外，过多的迭代次数会增加开发成本，延迟提交时间。

4. 喷泉模型

喷泉模型是一种以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程。该模型认为软件开发过程自下而上的各阶段是相互重叠和多次反复的，就像水喷上去又可以落下来，类似一个喷泉。各个开发阶段没有特定的次序要求，并且可以交互进行，可以在某个开发阶段随时补充其他任何开发阶段中的遗漏。

在喷泉模型中，各活动之间无明显边界，例如，分析和设计之间没有明显的边界。这种特性称为无间隙性。由于对象概念的引入，只用类和关系来表达分析、设计和实现等活动，从而可以较容易地实现活动的迭代和无间隙，提高软件项目开发效率，节省开发时间。

5. 变换模型

变换模型是基于形式化规格说明语言和程序变换的软件开发模型，它对形式化的软件规格说明进行一系列自动或半自动的程序变换，最后映射为计算机能够接受的软件系统。为了确认形式化规格说明与软件需求的一致性，往往以形式化规格说明为基础开发一个软件原型，用户可以从人机界面、系统主要功能和性能等方面对原型进行评审。必要时，可以修改软件需求、形式化规格说明和原型，直至原型被确认为止。这时，开发人员即可对形式化的规格说明进行一系列的程序变换，直至生成计算机可以接受的目标代码。

变换模型的优点是解决了代码结构经多次修改而变坏的问题，减少了许多中间步骤（例如，设计、编码和测试等）。但是，变换模型仍有较大的局限性，以形式化开发方法为基础的变换模型需要严格的数学理论和一整套开发环境的支持。

6. 智能模型

智能模型也称为基于知识的软件开发模型，它综合了上述若干模型，并把专家系统结合在一起。该模型应用基于规则的系统，采用规约和推理机制，帮助开发人员完成开发工作，并使维护在系统规格说明一级进行。为此，需要建立知识库，将模型本身、软件工程知识与特定领域的知识分别存入知识库。

7. V 模型

V 模型是在快速应用开发模型的基础上演变而来的，由于将整个开发过程构造成一个 V 字形而得名。V 模型应用在软件测试方面，和瀑布模型有一些共同的特征。V 模型的过程从左到右，描述了基本的开发过程和测试行为，其价值在于它非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发过程各阶段的对应关系，如图 7-3 所示。

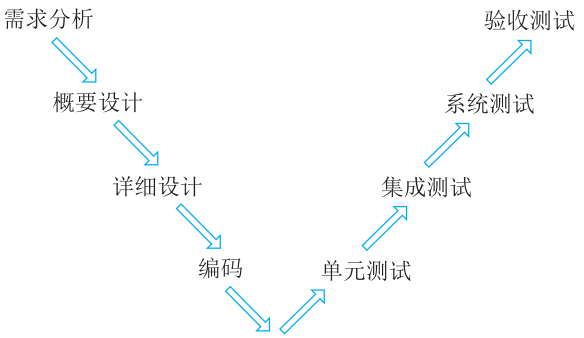

图 7-3　V 模型

在 V 模型中，单元测试是基于代码的测试，最初由开发人员执行，以验证程序代码的各个部分是否已达到预期的功能要求；集成测试验证了两个以上单元之间的集成是否正确，并有针对性地对详细设计中所定义的各单元之间的接口进行检查；在所有单元测试和集成测试完成后，系统测试开始以客户环境模拟系统的运行，验证系统是否达到了在概要设计中所定义的功能和性能；最后，当技术部门完成了所有测试工作后，由业务专家或用户进行验收测试，以确保产品能真正符合用户业务上的需要。

V 模型强调软件开发的协作和速度，将软件实现和验证有机地结合起来，在保证较高的软件质量的情况下缩短开发周期。V 模型适合企业级的软件开发，它更清楚地揭示了软件开发过程的特性及其本质。

7.2.2　快速应用开发

快速应用开发（Rapid Application Development，RAD）是一种比传统生命周期法快得多的开发方法，它强调极短的开发周期。RAD 模型是瀑布模型的一个高速变种，通过使用基于构件的开发方法获得快速开发。如果需求理解得很好，且约束了项目范围，利用这种模型可以很快开发出功能完善的信息系统。

1. RAD 的基本思想

RAD 的基本思想体现在以下 4 个方面：

（1）让用户更主动地参与到系统分析、设计和构造活动中来。

（2）将项目开发组织成一系列重点突出的研讨会，研讨会要让项目投资方、用户、系统分析师、设计人员和开发人员一起参与。

（3）通过一种迭代的构造方法，加速需求分析和设计阶段。

（4）让用户提前看到一个可工作的系统。

2. RAD 的开发阶段

RAD 的流程从业务建模开始，随后是数据建模、过程建模、应用生成、测试与交付。

（1）业务建模。确定驱动业务过程运作的信息、要生成的信息、如何生成、信息流的去向及其处理等，可以使用数据流图来帮助建立业务模型。

（2）数据建模。为支持业务过程的数据流查找数据对象集合、定义数据对象属性，并与其他数据对象的关系构成数据模型，可以使用 E-R 图来帮助建立数据模型。

（3）过程建模。将数据对象变换为要完成一个业务功能所需的信息流，创建过程以描述增加、修改、删除或获取某个数据对象，即细化数据流图中的加工。

（4）应用生成。利用第四代语言（4GL）写出处理程序，复用已有构件或创建新的可复用构件，利用环境提供的工具自动生成并构造出整个应用系统。

（5）测试与交付。因为 RAD 强调复用，许多构件已经是测试过的，这就减少了测试的时间。由于大量复用，所以一般只做总体测试，但新创建的构件还是要测试的。

3. RAD 的特点

RAD 采用基于构件的开发方法，复用已有的程序结构（如果可能的话）或使用构件，或者创建可复用的构件（如果需要的话）。在所有情况下，均可以使用 CASE 工具辅助进行软件构建。如果一个业务能够被模块化，使得其中每一个主要功能均可以在不到三个月的时间内完成，那么，它就是 RAD 的一个候选者。每个主要功能可由一个单独的 RAD 组来实现，最后再集成起来，形成一个整体。

RAD 通过大量使用可复用构件，加快了开发速度。但是，RAD 也具有以下局限性：

（1）并非所有应用都适合 RAD。RAD 对模块化要求比较高，如果有哪一项功能不能被模块化，那么 RAD 所需要的构件就会有问题；如果高性能是一个指标，且该指标必须通过调整接口使其适应系统构件才能获得，则 RAD 也有可能不能奏效。

（2）开发者和客户必须在很短的时间完成一系列的需求分析，任何一方配合不当，都会导致 RAD 项目失败。

（3）RAD 只能用于管理信息系统的开发，不适合技术风险很高的情况。例如，当一个新系统要采用很多新技术，或当新系统要与现有系统有较高的互操作性时，就不适合使用 RAD。

7.2.3 统一过程模型

统一过程（Unified Process，UP）是一个通用过程框架，可以用于种类广泛的软件系统、不同的应用领域、不同的组织类型、不同的性能水平和不同的项目规模。UP 是基于构件的，在为软件系统建模时，UP 使用的是 UML。与其他软件过程相比，UP 具有三个显著的特点，即用例驱动、以架构为中心、迭代和增量。

UP 提供了在开发组织中分派任务和责任的纪律化方法，它的目标是在可预见的日程和预算前提下，确保满足最终用户需求的高质量产品。对所有的关键开发活动，它为每个团队成员提供了使用准则、模板和工具指导。而通过对相同基础知识的一致理解，在进行需求分析、设计、

测试或配置管理等工作时，均能确保全体成员共享相同的知识、过程和开发软件的视图。

1. RUP 概述

RUP（Rational Unified Process）是 Rational 公司开发和维护的过程产品，是由 Objectory 过程演化而来的。RUP 将项目管理、业务建模、分析与设计等统一起来，贯穿整个开发过程。RUP 采用 Internet 技术，可以增强团队的开发效率，并为所有成员提供最佳的软件实现方案，它使团队中每个开发人员的见解和思想得到统一，使开发小组成员的沟通更为容易，而这正是任何项目取得成功的关键因素。RUP 可以增强开发人员对软件的预见性，最终的好处就是提高了软件质量，并有效缩短了软件从开发到投放市场的时间。RUP 为软件开发提供了规范性的指南、模板和范例，可用来开发所有类型的应用。

RUP 中的软件过程在时间上被分解为 4 个顺序的阶段，分别是初始阶段、细化阶段、构建阶段和移交阶段。每个阶段结束时都要安排一次技术评审，以确定这个阶段的目标是否已经满足。如果评审结果令人满意，就允许项目进入下一个阶段。基于 RUP 的软件过程模型如图 7-4 所示。

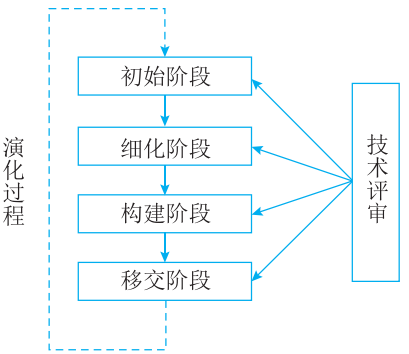

图 7-4　基于 RUP 的软件过程

从图 7-4 中可以看出，基于 RUP 的软件过程是一个迭代过程。初始、细化、构建和移交 4 个阶段就是一个开发周期，每次经过这 4 个阶段就会产生一代软件。除非产品退役，否则，通过重复同样的 4 个阶段，产品将演化为下一代产品，但每一次的侧重点都将放在不同的阶段上。

2. 初始阶段

初始阶段的任务是为系统建立业务模型并确定项目的边界。在初始阶段，必须识别所有与系统交互的外部实体，定义系统与外部实体交互的特性。在这个阶段关注的是整个项目的业务和需求方面的主要风险。对于建立在原有系统基础上的开发项目来说，初始阶段可能很短。初始阶段的实现过程如下：

（1）明确项目规模。建立项目的软件规模和边界条件，包括验收标准；了解环境及重要的需求和约束，识别系统的关键用例。

（2）评估项目风险。软件过程主要关心的是软件开发的已知方面，只能准确描述、计划、分配和评审那些已经知道将要完成的事情。风险管理则主要关心未知方面。在基于 RUP 的迭代式软件过程中，很多决策要受风险决定。要达到这个目的，开发人员需要详细了解项目所面临的风险，并对如何降低或处理风险有明确的策略。

（3）制订项目计划。估计整个项目的总体成本、进度和人员配备。综合考虑备选架构，评估设计和自制 / 外购 / 复用方面的方案，从而估算出成本、进度和资源。在这个过程中，要通过对一些概念的证实来证明可行性，可以采用可模拟需求的模型形式或用于探索高风险区的初始原型。初始阶段的原型设计工作应该限制在确信解决方案可行就可以了，具体实现留到细化阶段和构建阶段。

（4）阶段技术评审。初始阶段结束时要进行一次技术评审，检查初始阶段的目标是否完成，并决定继续进行项目还是取消项目。在评审过程中，需要考虑项目的规模定义、成本和进度估算是否适中、估算根据是否可靠、需求是否正确、开发方和用户方对软件需求的理解是否达成一致、是否已经确定所有风险且有针对每个风险的规避策略等问题。

3. 细化阶段

细化阶段的任务是分析问题领域，建立完善的架构，淘汰项目中最高风险的元素。在细化阶段，必须在理解整个系统的基础上，对架构做出决策，包括其范围、主要功能和诸如性能等非功能需求，同时为项目建立支持环境。细化阶段的实现过程如下：

（1）确定架构。确保架构、需求和计划足够稳定，充分减少风险，从而能够有预见性地确定开发所需的成本和开发进度。通过处理架构中的关键场景，建立一个已确定基线的架构，并验证其将在适当时间、以合理的成本支持系统需求。

（2）制订构建阶段计划。为构建阶段制订详细的过程计划，并为其建立基线。

（3）建立支持环境。包括开发环境、开发流程、支持构建团队所需的工具和自动化 / 半自动化支持。

（4）选择构件。评估现有的构件库和潜在构件，充分了解自制 / 外购 / 复用决策，以便有把握地确定构建阶段的成本和进度。集成所选构件，并按主要场景进行评估。

（5）阶段技术评审。评审时，需要检验详细的系统目标和范围、架构的选择，以及主要风险的解决方案。

在细化阶段，可执行的原型依赖于项目的范围、规模、风险和先进程度。必须至少处理初始阶段中识别的关键用例，因为关键用例通常揭示了项目的主要技术风险。

4. 构建阶段

在构建阶段，要开发所有剩余的构件和应用程序功能，把这些构件集成为产品，并进行详细测试。从某种意义上说，构建阶段是一个制造过程，其重点放在管理资源及控制操作，以优化成本、进度和质量。构建阶段的主要任务是通过优化资源和避免不必要的报废和返工，使开发成本降到最低；完成所有所需功能的分析、开发和测试，快速完成可用的版本；确定软件、场地和用户是否已经为部署软件做好准备。

在构建阶段，开发团队的工作可以实现某种程度的并行。一些项目的规模大得足够产生许多并行的增量构建过程，即使是较小的项目，也通常包括可以相互独立开发的构件，从而使各团队之间实现并行开发。这些并行活动在加速版本发布的有效性的同时，也增加了资源管理和工作流同步的复杂性。

构建阶段结束时也要进行技术评审，评审产品是否可以在 β 测试环境中进行安装和运行。

5. 移交阶段

当基线已经足够完善，可以安装到最终用户实际环境中时，则进入移交阶段。移交阶段的重点是确保软件对最终用户是可用的。移交阶段的主要任务是进行 β 测试，制作产品发布版本；对最终用户支持文档定稿；按用户的需求确认新系统；培训用户和维护人员；获得用户对

当前版本的反馈，基于反馈调整产品，例如，进行调试、性能或可用性的增强等。

移交阶段结束时也要进行技术评审，评审目标是否实现、是否应该开始演化过程、用户对交付的产品是否满意等。

从本节的介绍可以看出，RUP 由于太过庞大和复杂，相对于轻量级的敏捷方法来说，显得死板和难以实施。RUP 不但不能快速适应需求的变化，而且变更一个需求要经历复杂的过程和很多额外的工作。对于较小的组织和项目来说，使用敏捷方法可能比较合适，而使用 RUP 似乎有些费力不讨好。

7.2.4　敏捷方法

敏捷方法是从 20 世纪 90 年代开始引起广泛关注的一些新型软件开发方法，以应对快速变化的需求。虽然它们的具体名称、理念、过程、术语都不尽相同，但相对于“非敏捷”而言，它们更强调开发团队与用户之间的紧密协作、面对面的沟通、频繁交付新的软件版本、紧凑而自我组织型的团队等，也更注重人的作用。

1. 敏捷宣言

2001 年，Kent Beck 等人组织了敏捷联盟，阐述了敏捷开发的原则，试图强调灵活性在快速且有效地开发软件中所发挥的作用，他们共同签署了敏捷软件开发宣言，该宣言认为，个体和交互胜过过程和工具；可工作的软件胜过大量的文档；客户合作胜过合同谈判；响应变化胜过遵循计划。

敏捷方法强调：让客户满意和软件尽早增量发布；小而高度自主的项目团队；非正式的方法；最小化软件工程工作产品以及整体精简开发。产生这种情况的原因是，在绝大多数软件开发过程中，提前预测哪些需求是稳定的和哪些需求会变化非常困难；对于软件项目构建来说，设计和实现是交错的；从制定计划的角度来看，分析、设计、实现和测试并不容易预测；可执行原型和部分实现的可运行系统是了解用户需求和反馈的有效媒介。

目前，主要的敏捷方法有极限编程（eXtreme Programming，XP）、自适应软件开发（Adaptive Software Development，ASD）、水晶方法（Crystal）、特性驱动开发（Feature-Driven Development，FDD）、动态系统开发方法（Dynamic Systems Development Method，DSDM）、测试驱动开发（Test-Driven Development，TDD）、Scrum、敏捷数据库技术（Agile Database Techniques，AD）和精益软件开发（Lean Software Development）等。虽然这些过程模型在实践上有差异，但都遵循敏捷宣言或者敏捷联盟所定义的基本原则。这些原则包括客户参与、增量式移交、简单性、接受变更、强调开发人员的作用和及时反馈等。

2. 敏捷方法的特点

敏捷方法是一种以人为核心、迭代、循序渐进的开发方法。在敏捷方法中，软件项目的构建被切分成多个子项目，各个子项目的成果都经过测试，具备集成和可运行的特征。在敏捷方法中，从开发者的角度来看，主要的关注点有短平快的会议、小版本发布、较少的文档、合作为重、客户直接参与、自动化测试、适应性计划调整和结对编程；从管理者的角度来看，主要的关注点有测试驱动开发、持续集成和重构。

近年来，虽然敏捷方法发展得较快，但在实施的过程中，也暴露出很多问题，一些敏捷方法的基本原则很难实施，主要体现在以下 4 个方面：

（1）客户参与往往依赖于客户参与的意愿和客户自身的代表性。

（2）团队成员的性格可能不适合激烈的投入，可能无法做到与其他成员之间的良好沟通。

（3）对系统的变更做出优先级排序可能是极端困难的。

（4）维护系统的简洁性往往需要额外的工作，但迫于时间表的压力，可能没有时间执行系统简化过程。

与 RUP 相比，敏捷方法的周期可能更短。敏捷方法在几周或几个月的时间内完成相对较小的功能，强调的是能尽早将尽量小的可用的功能交付使用，并在整个项目周期中持续改善和增强，并且更加强调团队中的高度协作。相对而言，敏捷方法主要适用于以下场合：

（1）项目团队的人数不能太多，适合于规模较小的项目。

（2）项目经常发生变更。敏捷方法适用于需求萌动并且快速改变的情况，如果系统有比较高的关键性、可靠性、安全性方面的要求，则可能不完全适合。

（3）高风险项目的实施。

（4）从组织结构的角度看，组织结构的文化、人员、沟通性决定了敏捷方法是否适用。与这些相关联的关键成功因素有组织文化必须支持谈判、人员彼此信任、人少但是精干、开发人员所做的决定得到认可、环境设施满足团队成员之间快速沟通的需要。

3. 极限编程方法

敏捷方法中最著名的就是极限编程（XP），XP 是一种轻量、高效、低风险、柔性、可预测、科学且充满乐趣的软件开发方式，适用于小型或中型软件开发团队，并且客户的需求模糊或需求多变。与其他方法相比，XP 方法最大的不同如下：

（1）在更短的周期内，更早地提供具体、持续的反馈信息。

（2）迭代地进行计划编制，首先在最开始迅速生成一个总体计划，然后在整个项目开发过程中不断地发展它。

（3）依赖于自动测试程序来监控开发进度，并尽早捕获缺陷。

（4）依赖于口头交流、测试和源程序进行沟通。

（5）倡导持续的演化式的设计。

（6）依赖于开发团队内部的紧密协作。

（7）尽可能达到程序员短期利益和项目长期利益的平衡。

XP 由价值观、原则、实践和行为 4 个部分组成，它们彼此相互依赖、关联，并通过行为贯穿于整个生命周期。XP 的核心是其总结的四大价值观，即沟通、简单、反馈和勇气。它们是 XP 的基础，也是 XP 的灵魂。XP 的 5 个原则是快速反馈、简单性假设、逐步修改、提倡更改和优质工作。而在 XP 方法中，贯彻的是“小步快走”的开发原则，因此工作质量决不可打折扣，通常采用测试先行的编码方式来提供支持。

4. 动态系统开发方法

动态系统开发方法（DSDM）倡导以业务为核心，快速而有效地进行系统开发。可以把

DSDM 看成一种控制框架，其重点在于快速交付并补充如何应用这些控制的指导原则。

DSDM 是一整套的方法论，不仅仅包括软件开发内容和实践，也包括了组织结构、项目管理、估算、工具环境、测试、配置管理、风险管理、重用等各个方面的内容。

1）DSDM 的基本观点

DSDM 认为任何事情都不可能一次性圆满完成，应该用 20% 的时间完成 80% 的有用功能，以适合商业目的为准。实施的思路是：在时间进度和可用资源预先固定的情况下，力争最大化地满足业务需求（传统方法一般是需求固定，时间和资源可变），交付所需要的系统。对于交付的系统，必须达到足够的稳定程度，以在实际环境中运行；对于业务方面的某些紧急需求，也必须能够在短时间内得到满足，并在后续迭代阶段对功能进行完善。

2）DSDM 的基本原则

DSDM 的基本原则包括：

- 用户必须参与。
- 必须授权DSDM团队进行决策。
- 注重频繁交付产品。
- 判断产品是否可接受的基本标准是是否符合业务目的。
- 对准确的业务解决方案需要采用循环和增量开发。
- 开发期间的所有更改都是可逆的。
- 在高层次上制定需求的基线（以在低优先级的项目上获得一定的灵活性）。
- 在整个生命周期集成测试。
- 在所有参与者之间采用协作和合作方法。

5. Scrum

Scrum 是一个用于开发和维护复杂产品的框架，是一个增量的、迭代的开发过程。在这个框架中，整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个 Sprint，每个 Sprint 的建议长度是 2 ～ 4 周。

在 Scrum 中，使用产品 Backlog 来管理产品的需求，产品 Backlog 是一个按照商业价值排序的需求列表，列表条目的体现形式通常为用户故事。Scrum 团队总是先开发对客户具有较高价值的需求。在 Sprint 中，Scrum 团队从产品 Backlog 中挑选最高优先级的需求进行开发。

挑选的需求在 Sprint 计划会议上经过讨论、分析和估算得到相应的任务列表，称为 Sprint Backlog。在每个迭代结束时，Scrum 团队将递交潜在可交付的产品增量。Scrum 起源于软件开发项目，但它适用于任何复杂的或创新性的项目。

7.3　软件开发环境与工具

软件开发环境（Software Development Environment，SDE）是指支持软件的工程化开发和维护而使用的一组软件，由软件工具集和环境集成机制构成。软件工具是指 CASE 工具，用以

支持软件开发的相关过程、活动和任务；环境集成机制是指为工具集成和软件开发、维护及管理提供统一的支持。通过环境集成机制，各工具用统一的数据接口规范存储或访问环境信息库，采用统一的界面形式，保证界面的一致性，同时，为各工具或开发活动之间的通信、切换、调度和协同工作提供支持。

7.3.1 软件开发环境

软件开发环境应支持多种集成机制，例如，平台集成、数据集成、界面集成、控制集成和过程集成等。软件开发环境应支持小组工作方式，并为其提供配置管理，环境的服务可用于支持各种软件开发活动，包括分析、设计、编程、调试和文档等。

较完善的软件开发环境通常具有多种功能，例如，软件开发的一致性与完整性维护，配置管理及版本控制，数据的多种表示形式及其在不同形式之间的自动转换，信息的自动检索与更新，项目控制和管理，以及对开发方法学的支持。软件开发环境具有集成性、开放性、可裁减性、数据格式一致性、风格统一的用户界面等特性，因而能大幅度提高软件生产率。

1. 软件开发环境的分类

软件开发环境可按以下几种角度进行分类：

（1）按软件开发模型与开发方法分类，有支持瀑布模型、演化模型、螺旋模型和喷泉模型等不同模型，以及结构化方法、面向对象方法等不同方法的软件开发环境。

（2）按功能与结构特点分类，有单体型、协同型、分散型和并发型等多种类型的软件开发环境。

（3）按应用范围分类，有通用型和专用型软件开发环境。其中，专用型软件开发环境与应用领域有关。

（4）按开发阶段分类，有前端开发环境（支持系统规划、分析、设计等阶段的活动）、后端开发环境（支持编程、测试等阶段的活动）、软件维护环境和逆向工程环境等。

2. 集成机制

集成机制根据功能的不同，可划分为环境信息库、过程控制与消息服务器、环境用户界面三个部分。

（1）环境信息库。环境信息库是软件开发环境的核心，用以存储与系统开发有关的信息，并支持信息的交流与共享。环境信息库中主要存储两类信息：一类是开发过程中产生的有关被开发系统的信息，例如，分析文档、设计文档和测试报告等；另一类是环境提供的支持信息，例如，文档模板、系统配置、过程模型和可复用构件等。

（2）过程控制与消息服务器。过程控制与消息服务器是实现过程集成和控制集成的基础。过程集成是按照具体软件开发过程的要求进行工具的选择与组合，控制集成使各工具之间进行并行通信和协同工作。

（3）环境用户界面。环境用户界面包括环境总界面和由它实行统一控制的各环境部件及工具的界面。统一的、具有一致性的用户界面是软件开发环境的重要特征，是充分发挥环境的优越性、高效地使用工具并减轻用户的学习负担的保证。

3. 集成计算机辅助软件工程

目前，随着软件开发工具的积累与自动化工具的增多，软件开发环境已经进入了第三代，即集成计算机辅助软件工程（Integrated Computer-Aided Software Engineering，ICASE）阶段。集成方式经历了从点到点的数据转换（早期 CASE 采用的集成方式），到公共用户界面（第二代 CASE，在一致的界面下调用众多不同的工具），再到目前的信息库方式。这是 ICASE 的主要集成方式。ICASE 不仅提供数据集成和控制集成，还提供了一组用户界面管理设施和一大批工具，包括垂直工具集（支持软件生命周期各阶段，保证生成信息的完备性和一致性）、水平工具集（用于不同的软件开发方法）和开放工具槽（用于连接新的工具）。

ICASE 的信息库不仅定义了面向对象的数据库管理系统，提供了数据 - 数据集成机制，还建立了可以被环境中所有工具访问的数据模型，提供了数据 - 工具集成机制，实现了配置管理功能。ICASE 的进一步发展则是与软件开发方法的结合，以及智能化的 ICASE。

ICASE 的最终目标是实现应用软件的全自动开发，即开发人员只要写好软件的需求规格说明书，ICASE 就能自动完成软件开发工作，即自动生成供用户直接使用的软件和有关文档。

7.3.2　软件开发工具

软件开发环境中的工具可包括支持特定过程模型和开发方法的工具（例如，支持瀑布模型及数据流方法的工具，支持面向对象方法的工具等）和独立于模型和方法的工具（例如，界面辅助生成工具和文档出版工具等），也可包括管理类工具和针对特定领域的应用类工具。所有这些工具可分为贯穿整个开发过程的工具（例如，项目管理工具）和解决软件生命周期中某一阶段问题的工具（例如，软件估算工具等）。

1. 软件工具的分类

在软件生命周期中，要使用很多软件工具，从其功能上进行划分，可以分为软件开发工具、软件维护工具、软件管理和支持工具三类。

（1）软件开发工具。软件开发工具用来辅助开发人员进行软件开发活动，包括需求分析工具、设计工具、编码与排错工具等。

（2）软件维护工具。软件维护工具用来辅助维护人员对软件代码及其文档进行各种维护活动，包括版本管理工具、文档分析工具、开发信息库工具、逆向工程工具和再工程工具等。

（3）软件管理和支持工具。软件管理和支持工具用来辅助管理人员和软件支持人员的管理活动和支持活动，以确保软件高质量完成，包括项目管理工具、配置管理工具和软件评价工具等。

2. 开发工具的选择

开发工具的选择是软件项目成功的要素之一。对软件开发工具的评价和选择，要参照具体软件项目对开发工具的选择标准和要求，从功能、易用性、稳健性、硬件要求和性能，以及服务和支持等方面来衡量。简单地说，开发工具的选择主要决定于两个因素，分别是所开发系统的最终用户和开发人员。最终用户需求是一切软件的来源和归宿，也是影响开发工具的决定性因素；开发人员的爱好、习惯和经验也影响着开发工具的选择。

需要强调的是，开发工具的比较没有绝对的标准。评价一种开发工具，不仅要看它对设计模式和对象结构，以及管理的支撑情况，更重要的是要针对具体的使用环境、开发方法、软件架构和开发人员，以及最终用户来评价一种工具的适宜程度。

3. 快速开发工具

在 RAD 方法中，所包括的工具主要有数据库编程语言、界面生成器和报告生成器等。RAD 工具主要使用可视化技术。可视化技术是一种通过集成细粒度可复用构件来构造软件的方法，其主要思想是用图形工具和可复用构件来交互地编制程序。常用的 RAD 工具有 Microsoft Visual Studio.NET、Sun Java Studio Creator、WebLogic Workshop 和 Borland C# Builder 等。一般的可视化编程工具还有应用向导提供模板，按照步骤对程序员进行交互式指导，让用户定制自己的应用，然后就可以生成应用程序的框架代码，用户再在适当的地方添加或修改，以适应自己的需求。

7.4 软件过程管理

在无规则和混乱的管理条件下，先进的软件开发技术和工具并不能发挥应有的作用。于是，人们认识到，改进软件过程的管理是解决上述难题的突破口。但是，各个软件组织的过程成熟度有着较大的差别。为了做出客观、公正的比较，就需要建立一种衡量的标准。使用此标准一方面可以评价软件开发方的质量保证能力，在软件项目评标活动中选择开发方；另一方面，该标准也必然成为软件组织加强质量管理和提高软件产品质量的依据。

软件过程是软件生命周期中的一系列相关活动，即用于开发和维护软件及相关产品的一系列活动。软件产品的质量取决于软件过程，具有良好软件过程的组织能够开发出高质量的软件产品。

7.4.1 软件能力成熟度模型

软件能力成熟度模型（Capability Maturity Model，CMM）是一个概念模型，模型框架和表示是刚性的，不能随意改变，但模型的解释和实现有一定弹性。CMM 提供了一个软件能力成熟度的框架，它将软件过程改进的步骤组织成 5 个成熟度等级，为软件过程不断改进奠定了一个循序渐进的基础。

1. CMM 的等级

CMM 的目的是帮助组织对软件过程进行管理和改进，增强开发与改进能力，从而能按时地、不超预算地开发出高质量的软件。CMM 的 5 个成熟度等级分别为初始级、可重复级、已定义级、可管理级和优化级。

（1）初始级。初始级是未加定义的随意过程，软件过程的特点是无秩序的，有时甚至是混乱的。软件过程定义几乎处于无章法、无步骤可循的状态，软件产品所取得的成功往往依赖于极个别人的努力和机遇。

（2）可重复级。可重复级是规则化和纪律化的过程，软件过程已建立了基本的项目管理流

程，可用于对成本、进度和功能特性进行跟踪。对类似的应用项目有章可循，并能重复以往所取得的成功。

（3）已定义级。已定义级是标准化和一致化的过程，用于管理和工程活动的软件过程均已文档化、标准化，并形成了整个软件组织的标准软件过程。全部项目均采用与实际情况相吻合的、适当修改后的标准软件过程来进行操作。

（4）可管理级。已管理级是可预测的过程，软件过程和产品质量有详细的度量标准。软件过程和产品质量得到了定量的认识和控制。

（5）优化级。优化级是持续改进的过程，通过对来自过程、新概念和新技术等方面的各种有用信息的定量分析，能够不断地、持续性地对过程进行改进。

2. 关键过程域

除初始级以外，CMM 的每个级别的实现都定义成可操作的，每一级包含了实现这一级目标的若干关键过程域（Key Process Area，KPA），如表 7-1 所示。

表 7-1　关键过程域的分类

过程分类 / 成熟度等级	管理方面	组织方面	工程方面
优化级		技术变更管理 过程变更管理	缺陷预防
可管理级	量化过程管理		软件质量管理
已定义级	集成软件管理 组间合作	组织过程焦点 组织过程定义 培训计划	软件产品工程 同行评审
可重复级	需求管理 软件项目计划 软件项目跟踪与监控 软件子合同管理 软件质量保证 软件配置管理		

每个 KPA 都是由关键实施活动（Key Practices，KP）组成的，它们的执行表明该 KPA 在一个组织内部得到实现。

3. 能力成熟度模型集成

能力成熟度模型集成（Capability Maturity Model Integration，CMMI）融合了多种模型，形成了组织范围内过程改进的单一集成模型，其主要目的是消除不同模型之间的不一致和重复，降低基于模型进行改进的成本。CMMI 继承了 CMM 的阶段表示法和 EIA/IS 731 的连续式表示法。这两种表示方法各有优缺点，均采用统一的 24 个过程域，它们在逻辑上是等价的，对同一个组织采用两种模型分别进行 CMMI 评估，得到的结论应该是相同的。

（1）阶段式模型。阶段式模型基本沿袭 CMM 模型框架，仍保持 5 个成熟度等级，但关键

过程域做了一些调整和扩充，如表 7-2 所示。

表 7-2　阶段式模型的过程域分组

成熟度等级	过程域
可重复级	需求管理、项目计划、配置管理、项目监督与控制、供应商合同管理、度量和分析、过程和产品质量保证
已定义级	需求开发、技术解决方案、产品集成、验证、确认、组织级过程焦点、组织级过程定义、组织级培训、集成项目管理、风险管理、集成化的团队、决策分析和解决方案、组织级集成环境
可管理级	组织级过程性能、定量项目管理
优化级	组织级改革与实施、因果分析和解决方案

当组织通过了某一等级过程域中的全部过程，即意味着该组织的成熟度达到了这一等级。利用阶段式模型对组织进行成熟度度量，概念清晰、易于理解、便于操作。

（2）连续式模型。与阶段式模型相比，连续式模型没有与组织成熟度相关的几个阶段。连续式模型将 24 个过程域按照功能划分为过程管理、项目管理、工程和支持 4 个过程组。每组包含的过程域如表 7-3 所示。

表 7-3　连续式模型的过程域分组

过程组	过程域
过程管理	组织级过程焦点、组织级过程定义、组织级培训、组织级过程性能、组织级改革与实施
项目管理	项目计划、项目监督与控制、供应商合同管理、集成项目管理、风险管理、集成化的团队、定量项目管理
工程	需求管理、需求开发、技术解决方案、产品集成、验证、确认
支持	配置管理、度量和分析、过程和产品质量保证、决策分析和解决方案、组织级集成环境、因果分析和解决方案

连续式模型的过程域强调实践，每个过程域代表组织某一方面的能力。每个过程域的能力均分为 5 级，所有过程域共同的能力等级决定组织的能力等级。连续式模型允许组织对过程域进行裁剪，也允许对不同的过程域采用不同的能力等级。采用这种模式的评估结果用能力特征图表示，如图 7-5 所示。连续式模型允许一个过程域出现在多个特征图中，这些特征图分别代表某种能力的过程域的子集。

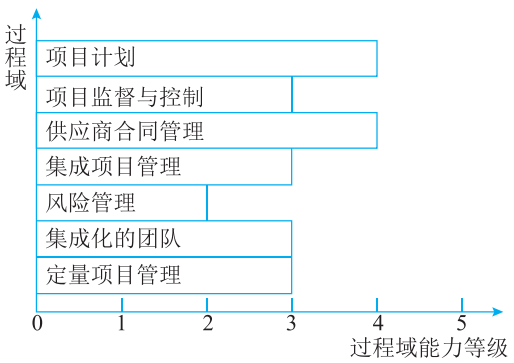

图 7-5　能力特征图示例

CMM 和 CMMI 是提高软件组织的成熟度和软件过程能力的有效模型和工具，组织无论采用 CMM 模型还是采用 CMMI 模型，无论是使用阶段式模型还是使用连续式模型，都能提高软件过程的成熟度，都能提高项目的软件过程能力，用两种模型或两种方法评价的结论应该是基本一致的。

7.4.2　软件过程评估

软件过程评估是根据过程模型或其他模型对组织的软件过程所进行的规范的评估。软件过程评估是由接受过培训的专业软件人员所组成的小组对组织的当前软件过程进行评估，以确定其状态，确定组织所面临的与软件过程相关事务的优先级，并从组织中获得对软件过程改进的支持。

1. CMM 模型

CMM 的一个重要思想是帮助软件组织通过基于模型的过程改进，达到使其软件过程向更高的成熟度等级迈进的目标。在这个过程中，一个组织必须建立起自己的软件过程，并根据 CMM 模型的要求对过程进行评估。根据评估的结果来进一步改进自己的软件过程，然后再一次评估，以期达到更高的成熟度等级，或者防止软件过程成熟度能力退化。如此反复循环，最终使一个组织的软件过程能力趋于更加成熟。基于 CMM 的评估方法分为以下 6 个步骤：

（1）成立评估小组，小组由软件工程和管理工作经验丰富的专家组成，小组成员应接受过 CMM 基本概念和评估方法的专门培训。

（2）参评单位的代表认真填写成熟度问卷表，并回答有关问题。

（3）评估小组分析调查问卷。

（4）评估小组现场访问、召开座谈会、审核过程文档，判断 KPA 的实践活动是否达到预定目标，并将结论记入文档。

（5）整理调查结果，撰写调查报告，指明软件过程的强项和弱项。

（6）绘制 KPA 剖面图，显示是否达到 KPA 的目标，并向有关部门提交评估的结论性意见。

2. Trillum 模型

Trillum 模型是一个主要用于嵌入式软件开发和支持的能力评估模型，它以 CMM 模型为基础，同时有新的发展。Trillum 模型的主要特点如下：

（1）模型的架构建立在路线图的基础上。

（2）模型不仅适用于软件，同时适用于硬件。

（3）模型强调以用户为关注的焦点。

（4）模型包括技术成熟度，主要面向通信产品。

3. Bootstrap 方法

Bootstrap 参考 ISO 9000-3 和 ESA PSS-05 等软件标准，设计了非常详细的过程质量属性结构，包括组织资源管理、测试方法和生命周期技术等 17 个属性，改进了 CMM 的问卷表和成熟度计算方法，使其可用于过程的每一个质量属性，从而得到一个过程质量剖面。Bootstrap 可适用于各类软件组织，在欧洲有很大的影响力。

4. ISO/IEC 15504 标准

ISO/IEC 15504 分为 9 个部分，分别是概念与介绍指南、过程与过程能力的参考模型、实施评估、评估实施指南、一个评估模型和指示指南、评估员资格认证指南、过程改进应用指南、

判断供应商过程能力指南和词汇表，其中第一部分是资料，第二部分和第三部分是标准，其他部分都是参考性的。在 ISO/IEC 15504 的第二部分（过程与过程能力的参考模型）中，在比较高的层次上详细定义了一个用于过程评估的二维参考模型，即过程维和能力维。通过将过程中的特点与不同的能力等级相比较，可以用此参考模型定义的一系列过程和框架对过程能力加以评估。

在 ISO/IEC 15504 中，能力等级是针对每个过程的，它定义了 6 级过程性能，每一级都用主要的过程特征术语和用于性能测量的属性来描述。具体包括：不完善的过程、已实施的过程、已计划与已跟踪的过程、已建立的过程、可预测的过程、优化的过程。

5. SJ/T 11234—2001 标准

我国行业标准《软件过程能力评估模型（SJ/T 11234—2001）》针对软件组织对自身软件过程能力进行内部改进的需要，与 CMMI 连续式模型基本相同。该模型有 22 个过程域，分为四大类，分别是过程管理类、项目管理类、工程类和支持类，如表 7-4 所示。

表 7-4　SJ/T 11234—2001 标准的过程域分组

过程域分组	过程域
过程管理	组织级过程焦点、组织级过程定义、组织级培训、组织级过程性能、组织级改革与实施
项目管理	项目计划、项目监督与控制、供方协议管理、集成项目管理、风险管理、定量项目管理
工程	需求管理、需求开发、技术解决方案、产品集成、验证、确认
支持	配置管理、度量和分析、过程和产品质量保证、决策分析和解决方案、因果分析和解决方案

SJ/T 11234—2001 的每个过程能力划分为 6 个评估等级：不完整级、已执行级、受管理级、已定义级、定量管理级和持续优化级。每个等级包含了通用目标、通用惯例、特定目标和特定惯例，它们组成一套衡量准则。不完整级是反映那些没有得到完整执行过程的状态，可能实现了部分特定目标，也可能什么目标都没有实现；处于已执行级的过程实现了全部特定目标；受管理级、已定义级、定量管理级和持续优化级不仅实现了全部特定目标，而且依次实现了对应更高的通用目标。

7.5　软件重用和再工程

软件重用是在两次或多次不同的软件开发过程中重复使用相同或相似软件元素的过程。软件元素包括程序代码、测试用例、设计文档、设计过程、需求分析文档甚至领域知识。通常，可重用的元素也称作软构件，可重用的软构件越大，重用的粒度越大。软件再工程指通过对目标系统的检查和改造，使用逆向工程、重构和正向工程方法，将现存系统重新构造为新的形式，开发出质量更高、维护性更好的软件。