第 1 章　软件工程过程

软件工程过程由一组相互关联的活动组成。这些活动将一个或多个输入转化为输出，同时消耗资源来完成转换。传统工程领域（如电子、机械、化学）的许多过程涉及将能源和物理实体从一种形式转变为另一种形式，例如，水力发电的大坝将势能转化为电能，石油精炼厂利用化学过程将原油转化为汽油。

软件工程过程关心的是软件工程师对软件的开发、维护和操作如何完成工作活动，如需求、设计、结构、测试、配置管理和其他软件工程过程。为了便于阅读，本书将“软件工程过程”称为“软件过程”。需要注意的是，“软件过程”表示工作活动，而不是实现软件的执行过程。

软件过程的具体目的是：促进人们之间的理解、沟通和协调；协助管理软件项目；以有效的方式衡量和提高软件产品的质量；支持改进过程；为过程执行的自动化提供基础。

本章的章节结构如图 1-1 所示，将从软件过程定义、软件生命周期、软件过程评估与改进和软件过程工具 4 个方面介绍软件过程。

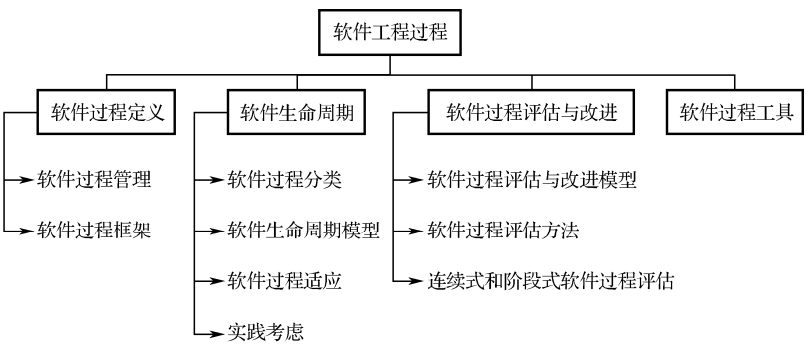

图 1-1　章节结构图

1.1　软件过程定义

如前所述，软件过程是一组相互关联的活动，将输入工作产品转换为输出工作产品。至少，软件过程的描述包括所需的输入、转换工作活动和生成的输出。如图 1-2 所示，软件过程可能还包括其输入和输出标准，并将工作活动分解为任务，这是软件过程管理的最小单位。软件过程输入可能是触发事件或另一个软件过程的输出。输入标准应该在一个软件过程开始之前得到满足。在成功地完成一个软件过程之前，必须满足所有指定的条件，包括输出工作产品或工作产品的验收标准。

软件过程可能包括子过程。例如，软件需求验证是一个过程，用来确定需求是否为软件开发提供了足够的基础，它是软件需求过程的一个子过程。软件需求验证的输入通常是软件需求规范和执行验证所需的资源（人员、验证工具、足够的时间）。软件需求验证的任务可能包括需求审查、原型和模型验证。这些任务包括个人和团队的工作任务。软件需求

验证的输出通常是一个经过验证的软件需求规范，它为软件设计和软件测试过程提供了输入。软件需求验证与软件需求过程的其他子过程通常会以不同的方式进行交叉和迭代。软件需求过程及其子过程可以在软件开发或修改过程中多次输入或输出。

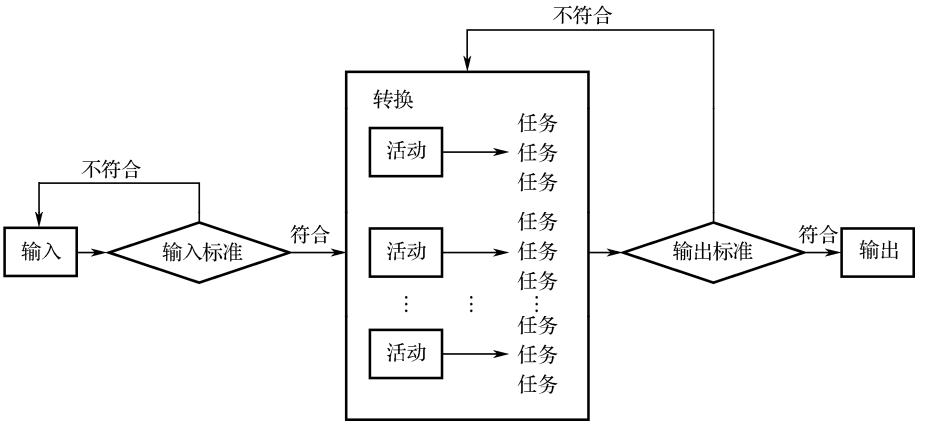

图 1-2　软件过程流程图

软件过程的完整定义可能还包括角色和能力、IT 支持、软件工程技术和工具、执行过程所需的工作环境、用于确定执行过程的效率和有效性的方法与度量。

此外，软件过程可能包括交叉技术、协作和管理活动。

定义软件过程的符号包括组织活动的文本列表和自然语言描述的任务、数据流图、状态图、业务流程建模标注、集成定义、Petri 网和统一建模语言活动图。流程中的转换任务可以定义为软件过程：一个软件过程可以被指定为一个有序的步骤，或者作为待完成任务的工作检查表。

必须强调的是，没有最好的软件过程或软件过程集，必须根据每个项目和每个组织的内容选择、调整和应用软件过程。

1.1.1　软件过程管理

软件过程管理的两个目标是，实现软件过程的效率与有效性和生产工作产品的系统方法的效率与有效性，无论是在个人、项目或组织层面，还是在引入新的或改进的过程方面。

过程随期望而改变，一个新的或修改后的过程将提高过程的效率与有效性，同时提高所生产工作产品的质量。引入一个新的过程、改进现有的过程或者改变现有的组织和框架（技术插入或工具的改变）是密切相关的，因为所有这些的目的通常都是为了改进软件产品的成本、开发进度或质量。过程的改变不仅对软件产品有影响，还经常会导致结构的改变。改变一个过程或引入一个新过程会在整个组织结构中产生连锁反应。例如，对 IT 基础设施、工具和技术的更改通常需要过程改变。

在第一次部署新过程时，可能会修改现有的过程（例如，在软件开发项目中引入检查活动可能会影响软件测试过程——参见第 6 章和第 10 章）。这些情况也可以称为“过程演化”。如果修改是广泛的，那么可能需要在组织培养和业务模型中进行更改以适应过程改变。

1.1.2　软件过程框架

建立、实施、管理软件过程和软件生命周期模型通常发生在单个软件项目的层次上。

然而，在组织中系统地应用软件过程和软件生命周期模型将有益于组织中的所有软件工作，尽管它需要在组织层面上的协议。软件过程框架可以提供过程定义、解释和应用过程的策略，以及用于实现过程的描述。此外，软件过程框架还可以提供资金、工具、培训和工作人员，这些人员被分配负责建立和维护软件过程框架。

软件过程框架因组织的规模和复杂性及组织内的项目而异。小型、简单的组织和项目有小而简单的框架需求，大型、复杂的组织和项目必须有更大、更复杂的软件过程框架。在后一种情况下，可以建立各种组织单元（如软件工程过程组或指导委员会）来监督软件过程的实现和改进。

一个常见的误解是，建立一个软件过程框架和实现可重复的软件过程将增加软件开发和维护的时间及成本。当然，引入或改进软件过程需要付出一定的代价。然而，经验表明，通过提高效率、避免返工，以及使用更可靠和可负担得起的软件，以系统地改进软件过程，往往会降低成本。因此，软件过程性能影响软件质量。

软件过程框架定义了若干框架活动，为实现完整的软件工程过程建立了基础。这些活动可广泛应用于所有软件开发项目，无论项目的规模和复杂性如何。此外，软件过程框架还包含一些适用于整个软件过程的普适性活动。一个通用的软件过程框架通常包含以下 5 个活动。

（1）沟通

在技术工作开始之前，和客户（及其他干系人）的沟通与协作是极其重要的。其目的是理解干系人的项目目标，并收集需求以定义软件特性和功能。

（2）策划

如果有地图，那么任何复杂的旅程都可以变得简单。软件项目好比是一个复杂的旅程，策划活动就是创建一张“地图”，以指导团队的项目旅程。这张地图称为软件项目计划。它定义和描述了软件工程工作，包括需要执行的技术任务、可能的风险、资源的需求、工作产品和工作进度计划。

（3）建模

无论你是工程师、建筑师还是木匠，每天的工作都离不开模型。你会画一张草图来辅助理解整个项目大的构想——体系结构、不同的构件如何结合，以及其他一些特性。如果需要，可以把草图不断细化，以便更好地理解问题并找到解决方案。软件工程师也是如此，需要利用模型来更好地理解软件需求，并完成符合这些需求的软件设计。

（4）构建

必须要对所做的设计进行构建，包括编码（手写的或者自动生成的）和测试。后者用于发现编码中的错误。

（5）部署

部署是指软件（全部或者部分增量）交付给用户，由用户对其进行评测并给出反馈意见。

上述 5 个通用软件框架活动既适用于简单小程序的开发，也可用于 Web APP 的建造，以及用于基于计算机的大型复杂系统工程的实现。在不同的应用案例中，软件过程的细节可能差别很大，但是软件框架活动类型都是相同的。

对许多软件项目来说，随着项目的开展，软件框架活动可以迭代应用。也就是说，在项目的多次迭代过程中，沟通、策划、建模、构建、部署等活动不断重复。每次项目迭代

都会产生一个软件增量，每个软件增量都实现了软件的部分特性和功能。随着每次增量的产生，软件将逐渐完善。

同时，软件过程框架活动由很多普适性活动来补充实现。通常，这些普适性活动贯穿软件项目始终，以帮助软件团队管理与控制项目的进度、质量、变更和风险。典型的普适性活动包括如下活动。

（1）软件项目跟踪和控制：项目组根据计划来评估项目进度，并且采取必要的措施来保证项目按进度计划进行。

（2）风险管理：对可能影响项目成果或者产品质量的风险进行评估。

（3）软件质量保证：确定和执行保证软件质量的活动。

（4）技术评审：评估软件工程产品，尽量在错误传播到下一个活动之前发现并清除它。

（5）测量、定义和收集软件过程、项目及产品的度量：帮助团队在发布软件时满足干系人的要求。同时，测量还可与其他软件框架活动和普适性活动配合使用。

（6）软件配置管理：在整个软件过程中管理变更所带来的影响。

（7）可复用管理：定义工作产品复用的标准（包括软件构件），并且建立构件复用机制。

（8）工作产品的准备和生产：包括生产工作产品（如建模、文档、日志、表格和列表等）所必需的活动。

软件过程框架如图 1-3 所示。可以看出，每个框架活动都由一系列软件工程动作构成：每个软件工程动作都要由一个任务集来定义，这个任务集明确了将要完成的工作任务、将要生产的工作产品、所需要的质量保证点，以及用于表明过程状态的项目里程碑。

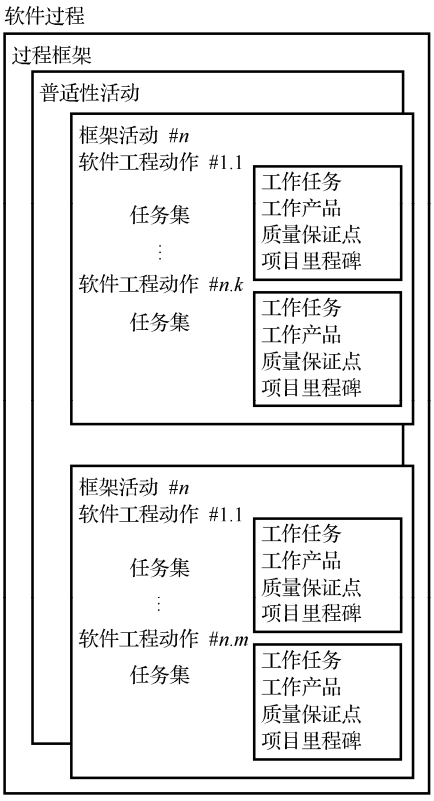

图 1-3　软件过程框架

在软件过程中，使用过程流来描述对软件过程框架中的活动、动作和任务如何在执行顺序和执行时间上进行组织。常用的过程流包括线性过程流、迭代过程流、演化过程流和并行过程流。线性过程流从沟通到部署顺序执行 5 个活动，如图 1-4 所示。迭代过程流在执行下一个活动前重复执行之前的一个或多个活动，如图 1-5 所示。演化过程流采用循环的方式执行各个活动，每次循环都能产生更完善的软件版本，如图 1-6 所示。并行过程流将一个或多个活动与其他活动并行执行，如图 1-7 所示。

图 1-4　线性过程流

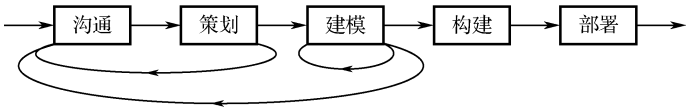

图 1-5　迭代过程流

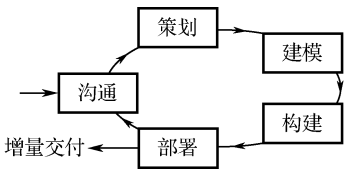

图 1-6　演化过程流

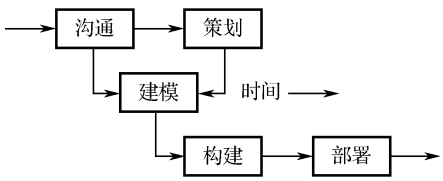

图 1-7　并行过程流

1.2　软件生命周期

软件开发生命周期（Software Development Life Cycle，SDLC）是软件的产生直到报废的生命周期。软件产品生命周期（Software Product Life Cycle，SPLC）包括软件开发生命周期，再加上为软件产品的部署、维护、支持、演化、退役和所有其他从开始到退役过程提供服务的软件过程，以及应用于整个软件产品生命周期的软件配置管理、软件质量保证过程等。一个软件产品生命周期可以包括用于演化和增强软件的多个软件开发生命周期。

单个软件过程没有时间顺序。软件过程之间的时间关系是由软件生命周期模型提供的：要么是软件开发生命周期，要么是软件产品生命周期。软件生命周期模型通常强调模型内的关键软件过程，以及它们在时间和逻辑上的相互依赖关系。在软件生命周期模型中，软件过程的详细定义可以直接提供，也可以参考其他文档。

除在软件过程中传递时间和逻辑关系之外，软件开发生命周期模型（或组织中使用的模型）还包括应用输入和输出标准的控制机制（如项目评审、客户评估、软件测试、质量阈值、项目演示和团队共识）。一个软件过程的输出常常为其他软件过程提供输入（例如，软件需求为软件架构设计、软件构建和软件测试等过程提供了输入）。并发执行多个软件过程活动可能产生一个共享的输出（例如，不同团队开发的多个软件组件之间的接口规范）。一些软件过程可能被认为不那么有效，除非同时执行其他的软件过程（例如，软件需求分析过程中的软件测试计划可以提高软件需求质量）。

1.2.1 软件过程分类

在软件开发和软件维护生命周期的各个部分中，已经定义了许多不同的软件过程。ISO 和 IECC 联合推出了“ISO/IEC 12207 软件生命周期过程”标准，为开发和管理软件提供了标准公共框架，如图 1-8 所示。

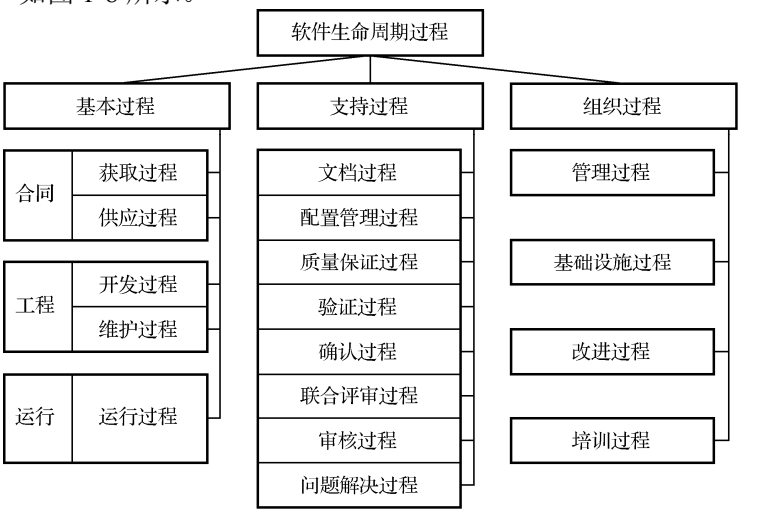

图 1-8 “ISO/IEC 12207 软件生命周期过程”标准公共框架

1．基本过程

基本过程定义了与软件生产直接相关的过程，即软件从无（或原有）到（新）有到运营的过程，包括软件的获取、供应、开发、运行和维护。

（1）获取过程，是指获取方为得到一个软件产品所进行的一系列活动，包括确定获取产品的需求定义、投标准备、合同准备和修改、对供应方的监督及验收完成和结束。

（2）供应过程，是指为获取方提供软件产品所进行的一系列活动，包括理解产品需求、应标准备、合同签订、计划制订、实施和控制、评价及交付完成。

（3）开发过程，是指组织开发软件所从事的一系列活动，包括需求分析、系统设计、编码、测试、安装及验收。在开发过程中还贯穿了其他软件过程的实施。

（4）运行过程，是指操作人员日常使用软件的过程。该过程是指用户和操作人员在用户业务运行环境中，使用软件投入运行所进行的一系列活动。其目的是在软件开发过程完成后，将软件从开发环境转移到用户的业务环境中运行时，对用户的要求提供咨询和帮助，并对运行效果做出评价。这个过程为开发过程和维护过程提供反馈信息。运行管理方可根据对软件项目的总体要求，按照软件管理过程的内容对运行过程进行管理。

（5）维护过程，是指维护人员所从事的一系列活动。其目的是在保持软件整体性能的同时进行修改，使其达到某一需求，直至其被废止。维护包括改正性、适应性和完善性维护。维护过程包括过程实现、问题分析与修改、修改实施、维护评审和验收、移植和软件退役等。在维护过程中还贯穿了其他软件过程的实施。

2．支持过程

支持过程是指为了保证基本过程的正常运行、目标的实现和质量的提高所从事的一系列过程。它们可被基本过程的各个过程部分或全部采用，并由它们自己的组织或一个独立

组织负责实施，也可由用户负责实施。

（1）文档过程，用于记录任何其他过程所产生的特定信息的一组活动。

（2）配置管理过程，用于捕获和维护开发过程中所产生的信息和产品的一组活动，以便于后续开发与维护。

（3）质量保证过程，用于客观地保证产品和相关过程与需求文档和计划保持一致的一组活动。

（4）验证过程，用于检验产品的活动，是依据实现的需求定义和产品规范，确定某项活动的产品是否满足所给定或所施加的要求和条件的过程。验证过程一般根据软件项目需求，按不同深度确定验证产品所需要的活动，包括分析、评审和测试，其执行具有不同程度的独立性。为了节约费用和有效进行，验证活动应尽早与采用它的过程（如获取、开发、运行和维护）相结合。该过程的成功实施期望带来如下结果：

- 根据需要验证的产品所制订的规范（如产品规格说明）实施必要的检验活动；
- 有效地发现各类阶段性产品所存在的缺陷，并跟踪和消除缺陷。

（5）确认过程，用于确认产品的活动。这是一个确定需求和最终的、已建立的系统或软件（产品）是否满足特定的预期用途的过程，集中判断产品中所实现的功能、特性是否满足客户的实际需要。确认过程和验证过程构成了软件测试缺一不可的组成部分，也可以将之看作质量保证活动的重要支持手段。确认应该尽量在早期阶段进行，如阶段性产品的确认活动。确认和验证相似，也具有不同程度的独立性。该过程的成功实施期望带来如下结果：

- 根据客户实际需要，确认所有产品相应的质量准则，并实施必要的确认活动；
- 提供有关证据，以证明开发出的产品满足或适应指定的需求。

（6）联合评审过程，由双方使用的、评估其他活动的状态和产品的活动。联合评审过程评价一项活动的状态和产品所需遵循的规范及要求，一般要求供、需双方共同参加。其评审活动在整个合同有效期内进行，包括管理评审和技术评审。管理评审主要依据合同的目标，与客户就开发进度、内容、范围和质量标准进行评估、审查，使双方通过充分交流达成共识，以保证开发出客户满意的产品。该过程的成功实施期望带来如下结果：

- 与客户、供应商及其他利益相关方（或独立第三方）对开发的活动和产品进行评估；
- 为联合评审的实施制订相应的计划与进度，跟踪评审活动，直至结束。

（7）审核过程，用于确定项目与需求、计划与合同的符合程度。审核过程判断各种软件活动是否符合用户的需求、质量计划和合同所需要的其他各种要求。审核过程发生在软件组织内部，也称内部评审。审核一般采用独立的形式对产品及所采用的过程加以判断、评估，并按项目计划中的规定，在预先确定的里程碑（代码完成日、代码冻结日和软件发布日等）之前进行。对于审核中出现的问题，应加以记录，并按要求输入问题解决过程。该过程的成功实施期望带来如下结果：

- 判断是否与指定的需求、计划及合同相一致；
- 由合适的、独立的一方来安排对产品或过程的审核工作；
- 确定其是否符合特定需求。

（8）问题解决过程，一组在分析和根除存在问题时所要执行的活动。不论问题的性质或来源如何，这些问题都是在实施开发、运行、维护或其他过程期间暴露出来的，需要得到及时纠正。问题解决过程的目的是及时提出相应对策、形成文档，以保证所有暴露的问题得到分析和解决，并能预见到这一问题领域的发展趋势。该过程的成功实施期望带来如下结果：

- 采用及时的、有明确职责的、文档化的方式，以确保所有发现的问题都经过了相应的分析并得到解决；
- 提供一种相应的机制，以识别所发现的问题并根据相应的趋势采取行动。

3．组织过程

组织过程为软件工程提供支持，包括管理、基础设施、改进和培训过程。

（1）管理过程，是指软件生命周期过程中管理者所从事的一系列活动和任务，如对获取、供应、开发、运行、维护或支持过程的活动进行管理，目的是在一定的周期和预算范围内有效地利用人力、资源、技术和工具完成预定的软件产品，实现预期的功能和其他质量目标。管理过程是在整个软件生命周期中为工程过程、支持过程和获取/供应过程的实践活动提供指导、跟踪和监控的过程，从而保证软件过程按计划实施并能到达预定目标。管理过程是软件生命周期中的基本管理活动，为软件过程和执行制订计划，帮助软件过程建立质量方针、配置资源，对软件过程的特性和表现进行度量，收集数据，负责产品管理、项目管理、质量管理和风险管理等。一个有效的、可行的软件过程能够将人力资源、流程和实施方法结合成一个有机的整体，并能全面地展现软件过程的实际状态和性能，从而可以监督和控制软件过程的实现。对软件过程的监督、控制实际孕育着一个管理的过程。管理过程包括：

- 项目管理，是指计划、跟踪和协调项目执行及生产所需资源。项目管理过程的活动，包括软件基本过程的范围确定、策划、执行和控制、评审和评价等。
- 质量管理，是指对项目产品和服务的质量加以管理，从而获得最高的客户满意度。此过程包括在项目及组织层次上建立对产品和过程质量管理的关注。
- 风险管理，是指在整个软件生命周期中对风险不断地进行识别、诊断和分析，以回避、降低或消除风险，并在项目及组织层次上建立有效的风险管理机制。
- 子合同商管理，是指选择合格子合同商并对其进行管理。

（2）基础设施过程，是指建立和维护其他过程所需的基础设施的过程。例如，软件工具、技术、标准及开发、支持、运行与维护所需的设施。其主要活动是定义并建立各个过程所需要的基础设施，并在其他相关过程执行时维护其所建立的基础设施。

（3）改进过程，是指建立、评估、度量、控制和改进软件生命周期过程的过程。其主要活动是制订一套组织计划，评估相关过程，并实施分析和改进过程。

（4）培训过程，是指为软件产品提供人员培训的过程。其主要活动是制订人员培训计划、开发培训资料及培训计划的实施。

1.2.2 软件生命周期模型

软件的无形和可延展特性允许使用各种各样的软件开发生命周期模型（简称开发模型）。在线性开发模型中，软件开发的各个阶段按照需要依次通过反馈和迭代完成，然后集成、测试和交付单个产品。在迭代开发模型中，软件是在迭代周期上以增加功能的方式进行开发的。在敏捷开发模型中，经常需要向客户或用户代表演示工作产品，由客户或用户代表在短的迭代周期内指导软件的开发，从而产生可运行的、可交付的软件的小增量。如果需要，增量、迭代和敏捷等开发模型可以将工作产品的早期子集交付到用户环境中。

线性开发模型有时被称为预测型开发模型，而迭代和敏捷开发模型被称为自适应开发模型。应该注意的是，在软件产品生命周期期间的各种维护活动可以使用不同的开发模型来进行，视适用情况而定。

各种开发模型的一个区别特征是软件需求的管理方式。线性开发模型通常在项目启动和计划期间开发完整的软件需求集，然后严格控制软件需求；对软件需求的更改是基于由变更控制委员会处理的变更请求（参见第 8 章）的。增量开发模型基于在每个增量中待实现软件需求的划分，产生可运行、可交付的软件的连续增量；就像在线性开发模型中一样，软件需求可以被严格控制，或者随着软件产品的发展，在修改软件需求方面也有一定的灵活性。敏捷开发模型最初可以定义产品范围和高级特性；然而，敏捷开发模型的设计目标是在项目过程中促进软件需求的演化。

必须强调的是，从线性到敏捷的开发模型的连续体并不是一条直线。不同方法中的元素可以合并到一个特定的模型中。例如，增量开发模型可能包含顺序的软件需求和设计阶段，但在软件构建过程中也允许相当灵活地修改软件需求和架构。

常见的开发模型有：瀑布模型、增量模型、演化模型、原型模型、螺旋模型、统一过程模型和敏捷过程模型。

1．瀑布模型

瀑布模型是典型的软/硬件开发模型，该模型也称传统软件生命周期模型。如图 1-9 所示，它包括需求分析、设计、编码、集成与系统测试、运行与维护几个阶段。在每个阶段分别提交以下产品：软件需求规格说明、系统设计说明、实际代码和测试用例、最终产品、产品升级等。工作产品流经“正向”开发的基本步骤路径。“反向”的步骤流表示对前一个可提交产品的重复变更。由于所有开发活动都具有非确定性，因此是否需要重复变更，仅在下一个阶段或更后的阶段才能认识到。这种“返工”不仅在以前阶段的某个地方有需要，而且对当前正在进行的阶段也同样重要。

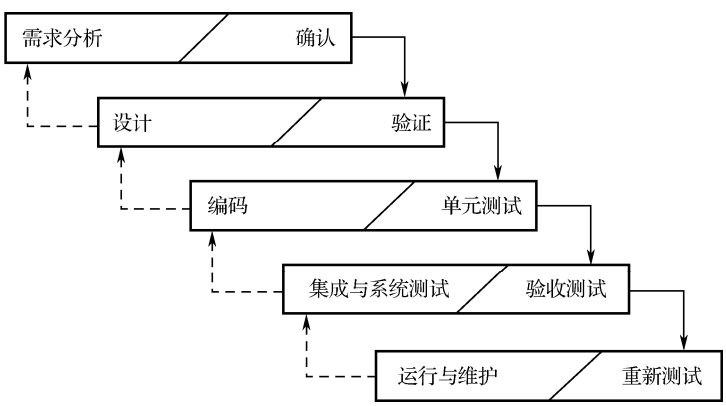

图 1-9　瀑布模型

该模型的主要特点是：

- 每个阶段都以验证/确认/测试活动作为结束，其目的是尽可能多地消除本阶段产品中存在的问题。
- 在随后阶段里，尽可能对前面阶段的产品进行迭代。

瀑布模型是第一个被完整描述的过程模型，是其他过程模型的鼻祖。其优点是：

- 容易理解、管理成本低。瀑布模型的主要成果是通过文档从一个阶段传递到下一个阶段。各阶段间原则上不连续也不交叠，因此可以预先制订计划来降低计划管理的成本。

- 它不提供有形的软件成果，除非到生命周期结束时。但文档产生并提供了贯穿整个生命周期的进展过程的充分说明，允许基线和配置在早期接受控制，并且前一个阶段作为下一个阶段被认可的、文档化的基线。

它的缺点表现为：

- 用户必须能够完整、正确和清晰地表达其需要。但在实际系统开发中，经常会出现用户与开发人员沟通存在巨大差异，用户提出的需求含糊又被开发人员随意解释，以及用户需求会随着时间推移不断变化等问题。
- 可能要花费更多的时间来建立一些用处不大的文档。
- 在开始的两个或三个阶段中，很难评估真正的进度状态。
- 在一个项目的早期阶段，过分强调基线和里程碑处的文档。
- 开发人员一开始就必须理解其应用范围。
- 当接近项目结束时，会出现大量的集成和测试工作。
- 直到项目结束之前，都不能演示系统的能力。

瀑布模型是传统过程模型的典型代表，因为管理简单，所以常被获取方作为合同上的模型。在一个阶段完成后，生产出一个具体的产品；如果需要的话，可以对这一产品进行独立的检验。获取方可以按阶段向开发方支付费用，这意味着双方必须客观地对其完成情况进行核实。

当一个项目有稳定的产品定义且很容易被理解的技术解决方案时，可以使用瀑布模型。在这种情况下，瀑布模型可以帮助及早发现问题，降低项目的阶段成本。它提供开发者渴望的稳定需求。若要对一个定义得很好的版本进行维护或将一个产品移植到一个新的平台上，那么瀑布模型是快速开发的一个恰当选择。

对于那些容易理解但很复杂的项目，采用瀑布模型比较合适，因为这样可以用顺序的方法处理复杂的问题。在质量需求高于成本需求和进度需求的时候，瀑布模型表现得尤为出色。由于在项目进展过程中基本不会产生需求的变更，因此，瀑布模型避免了常见的、巨大的潜在错误源。

瀑布模型的一个变体就是 V 模型，如图 1-10 所示。它在每个环节中都强调了测试（并提供了测试的依据），同时又在每个环节中都做到了对实现者和测试者的分离。由于测试者相对于实现者的关系是监督、考察和评审，因此测试者相当于在不断地做回顾和确认。

在图 1-10 中，左半部分是分析和设计，是软件设计实现的过程，同时伴随着质量保证活动——审核的过程，也就是静态测试过程；右半部分是对左边结果的检验，是动态测试的过程，即对分析和设计的结果进行测试，以确认它们是否满足用户的需求。

- 需求分析和功能设计对应验收测试，说明在做需求分析、功能设计的同时，测试人员就可以阅读、审查需求分析的结果，从而了解产品的设计特性、用户的真正需求，确定测试目标，准备用例并策划测试活动。
- 当设计人员在做系统设计时，测试人员可以了解系统是如何实现的，以及基于什么样的平台。这样就可以设计系统的测试方案和测试计划，并事先准备系统的测试环境，包括硬件和第三方软件的采购。因为这些准备工作实际上是要花去很多时间的。
- 当设计人员在做详细设计时，测试人员可以参与设计，对设计进行评审，找出设计的缺陷，同时设计功能、新特性等各方面的测试用例，完善测试计划，并基于这些测试用例来开发测试脚本。

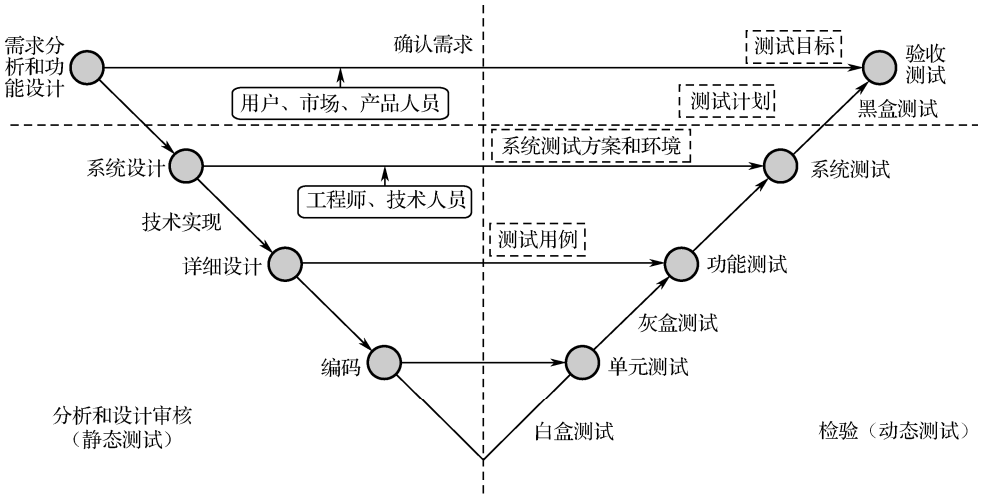

图 1-10　V 模型

- 在编程的同时进行单元测试是一种很有效的办法，这样做可以尽快找出程序中的错误。充分的单元测试可以大幅度提高程序质量，降低成本。

从图 1-10 中可以看出，V 模型中的质量保证活动和项目是同时展开的。项目一启动，软件测试的工作也就启动了，从而避免了瀑布模型在代码完成之后才进行软件测试的弊端。其特点如下：

- 图 1-10 中，虚线上面表明，其需求分析、定义和验收测试等主要工作是面向用户的。要与用户进行充分的沟通和交流，或者是和用户一起完成。相对来说，虚线下面的大部分都是技术性工作，在开发组织内部进行，主要由工程师、技术人员完成。
- 图 1-10 中，越向下，白盒测试使用得越多。单元测试、功能测试、系统测试大多将白盒测试和黑盒测试结合起来使用，形成灰盒测试。在验收测试中，因为用户一般都要参与，所以使用黑盒测试。

V 模型被广泛应用于软件外包中。由于劳动力短缺等多种原因，很多企业把项目直接外包给国内/国外的开发团队。项目成果的阶段性考查成为第一要务，因为这直接决定了何时、如何，以及由谁来进入下一个环节。

因此，V 模型变得比其他模型更为实用。模型的左半部分由接受外包任务的团队或者公司负责，而右半部分则由企业中有丰富经验的工程人员负责。这样既节省人力，又可以保证工程质量。事实上，即使图 1-10 左半部分的外包任务是由多个团队同时承接的，负责右半部分的工程人员也不需要更多的投入。

2．增量模型

增量模型是由瀑布模型演变而来的，它是对瀑布模型的精化。该模型有一个假设，即需求可以分段，成为一系列增量产品，对每个增量可以分别进行开发，如图 1-11 所示。

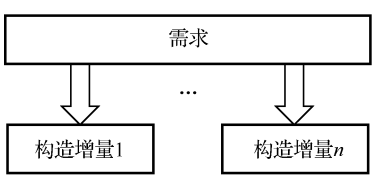

图 1-11　增量模型

在开始开发时，需求就很明确，并且软件还可以被适当地分解为一些独立的、可交付的产品，称为构造增量；在开发中，希望尽快提交其中的一些增量产品。例如，一个数据库系统，它必须通过不同的用户界面，为不同类型的用户

提供不同的功能。在这种情况下，首先实现完整的数据库设计，并把一组具有高优先级的用户功能和界面作为一个增量，然后陆续构造其他类型用户所需求的增量。

图 1-12 表达了如何利用瀑布模型来开发增量模型中的构造增量。尽管该图表示对不同增量的设计和实现完全可以是并发的，但在实际中，可以按任意期望的并行程度进行增量开发。例如，可以在完成了第一个增量设计之后，吸取经验教训，再转向第二个增量的设计。

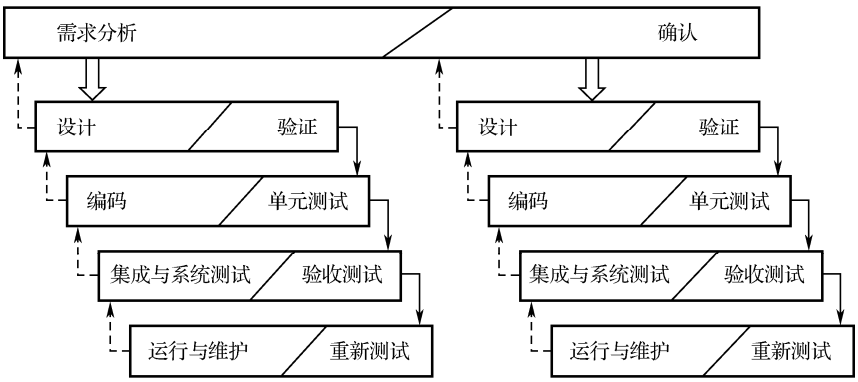

图 1-12　增量模型示意图

如果一个增量并不需要交付给用户，这样的增量通常被称为一个“构造增量”。如果增量需要被交付，它们就被认为是发布版本。在编写软件生命周期计划时，不论是正式的还是非正式的，都要注意使用用户期望的术语，其表达要与合同和工作陈述保持一致。

增量模型作为瀑布模型的一个变体，具有瀑布模型的所有优点。此外，它还有以下优点：

① 第一个可交付版本所需要的成本和时间是很少的。

② 开发由增量表示的小系统所承担的风险是不大的。

③ 由于很快发布了第一个版本，因此可以减少用户需求的变更。

④ 允许增量投资，即在项目开始时，可以仅对一个或两个增量投资。

然而，如果增量模型不适于某些项目，或使用有误，则有以下缺点：

① 如果没有对用户的变更要求进行规划，那么产生的初始增量可能会造成后来增量的不稳定。

② 如果需求不像早期考虑的那样稳定和完整，那么一些增量可能需要重新开发、重新发布。

③ 管理成本、进度和配置的复杂度，可能会超出组织的能力。

从缺点第①点和第③点可以看出，如果用户的变更要求与以前的增量矛盾，则双方容易发生冲突。因此在采用此模型时，开发方需要有合适的配置管理和成本计算系统，并在合同中明确给出变更条款。如果出现问题，就可以依据合同进行处理，化解矛盾。

如果采用增量投资方式，用户就可以对一些增量进行招标。然后，开发人员按提出的期限进行增量开发，因此，可以用多个契约来管理组织的资源和成本。

当需要以增量方式开发一个具有已知需求和定义的产品时，可以使用增量模型。其优点是，产品的各模块在很大程度上可以并行开发，从而可以在开发周期内尽早地证明操作代码的正确性而降低产品的技术风险。注意，如果在项目中并行执行的活动数量增大，则管理项目的复杂度就会加大。

3．演化模型

演化模型显式地把增量模型扩展到需求阶段。从图 1-13 可以看出，为了得到构造增量 2，使用了构造增量 1 来精化需求。这一精化可以有多个来源和路径。

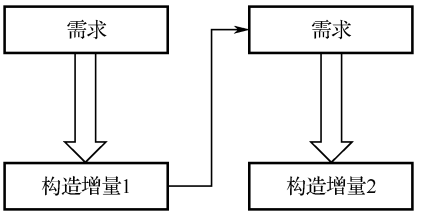

图 1-13　演化模型

首先，如果一个早期的增量已向用户发布，那么用户会以变更要求的方式提出反馈，以支持以后增量的需求开发。其次，实实在在地开发一个构造增量，为以前还没有认识到的问题提供了可见性，以便实际开始这一增量工作。

在演化模型中，仍然可以使用瀑布模型来管理每个增量。一旦理解了需求，就可以像实现瀑布模型那样开始设计阶段和编码阶段。

使用演化模型不能够成为弱化需求分析的借口。在项目开始时，应考虑所有需求来源的重要性和风险，对这些来源的可用性进行评估。只有采用这一方法，才能识别和界定不确定需求，并识别第一个增量中所包含的需求。此外，合同条款应该反映出所采用的开发模型。例如，对每个增量的开发和交付，双方应该按照合同进行协商，包括下一个增量的人力成本和费用的选择。

同样，成本计算、进度控制、状态跟踪和配置管理系统必须能够支持这一模型。由于演化的增量具有明确的顺序，因此与增量模型相比，演化模型面临的挑战通常是较弱的。但应该认识到，一定程度的并发总是存在的，因此系统必须允许某一层次的并行开发。

演化模型的优点和缺点与增量模型类似。特别地，演化模型还具有以下优点：

- 在需求不能予以规范时，可以使用演化模型。
- 用户可以通过运行系统的实践，对需求进行改进。
- 与瀑布模型相比，需要更多用户/获取方的参与。

演化模型的缺点表现为：

- 演化模型的使用仍然处于初步探索阶段，因此具有较大的风险，需要进行有效的管理。
- 该模型的使用很容易成为不编写需求分析或设计文档的借口，即便能够很清晰地描述需求分析或设计也是如此。
- 用户/获取方不理解该方法的自然属性，因此当结果不够理想时，可能产生抱怨。

当需求和产品定义没有被很好地理解，并需要快速地开发和创建一个能展示产品外貌与功能的最初版本时，特别适合使用演化模型。这些早期的增量能帮助用户确认和调整需求并帮助他们寻找相应的产品定义。

演化模型与增量模型具有许多相同的优点，而且具有能使产品适合需求变更的显著优点，它们还引进了附加的过程复杂性和潜在的更长的产品生命周期。

4．原型模型

原型模型是增量模型的一种形式。在开发真实系统前，首先需要构建一个简单的系统原型，实现用户与系统的交互，用户在对原型进行使用的过程中，不断发现问题，从而达到进一步细化系统需求的目的。开发人员在已有原型的基础上，通过逐步调整原型来确定

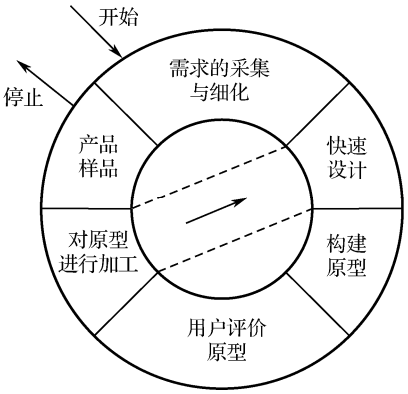

图 1-14　原型模型

用户的真正需求，进而开发出用户满意的系统。如图 1-14 所示为原型模型。

原型模型可以克服瀑布模型的缺点，减少因为需求不明确造成的开发风险。它的关键之处在于，能尽可能快速地建立原型。一旦确定了用户的真正需求，所建造的原型将被丢弃。因此，使用原型模型进行软件开发，最重要的是必须迅速建立原型，随之迅速修改原型，以反映用户的需求，而不是系统的内部结构。

5．螺旋模型

螺旋模型是由 Boehm 提出的另一种开发模型。在这一模型中，开发工作是迭代进行的，即只要完成了开发的一个迭代过程，另一个迭代过程就开始了。该模型关注解决问题的基本步骤，由此可以标识问题，标识一些可选方案，最终选择一个最佳方案，遵循动作步骤并实施后续工作。尽管螺旋模型和一些迭代模型在框架与全局体系结构上是等同的，但它们所关注的阶段及其活动是不同的。

开发人员和用户使用螺旋模型可以完成如下工作：

- 确定目标、方案和约束。
- 识别风险和效益的可选路线，选择最优方案。
- 开发本次迭代可供交付的内容。
- 评估完成情况，规划下一个迭代过程。
- 交付给下一步，开始新的迭代过程。

螺旋模型扩展了增量模型的管理任务范围，因为增量模型基于以下假定：需求是最基本的，并且是唯一的风险。在螺旋模型中，决策和降低风险的空间是相当广泛的。

螺旋模型的另一个特征是，实际上只有一个迭代过程用于真正开发可交付的产品。如果项目的开发风险很大，或用户不能确定系统需求，螺旋模型就是一个好的生命周期模型。

螺旋模型强调了原型构造。需要注意的是，螺旋模型不必要求原型，但构造原型比较适合这一过程模型。

在螺旋模型中，把瀑布模型作为一个嵌入的过程，即需求分析、设计、编码和交付的瀑布过程，是螺旋一周的组成部分。

螺旋模型是一种以风险为导向的生命周期模型，它把一个软件项目分解成一个个小项目。每个小项目都标识一个或多个主要风险因素，直到所有主要风险因素都被确认。“风险”的概念在这里是有外延的，它可以是需求或者架构没有被理解清楚、潜在的性能问题、根本性的技术问题等。在所有的主要风险因素被确定后，螺旋模型就像瀑布模型一样中止。

在螺旋模型中，越早期的迭代过程，其成本越低。规划概念比需求分析的代价低，需求分析比开发设计、集成和测试的代价低。

在螺旋模型中，项目范围逐渐增量展开。项目范围展开的前提是风险被降低到仅仅为下一步扩展部分的、可以接受的水平。在该模型中，要进行几次迭代，以及每次迭代中通常采用几个步骤完成并不重要，尽管那是很好的工作次序；重要的是，要根据项目的实际需求调整螺旋的每次迭代过程。

可以采取几种不同的方法把螺旋模型和其他生命周期模型结合在一起使用，通过一系列降低风险的迭代过程来开始项目。在风险降低到一个可以接受的水平后，可以采用瀑布模型或其他非基于风险分析的模型来推断开发效果。可以在螺旋模型中把其他过程模型作为迭代过程引入。如果遇到“不能确定性能指标是否能够达到”的风险，可以使用原型模型来验证是否能达到目标。

螺旋模型最重要的优势是，随着成本的增加，风险随之降低。时间和资金花得越多，风险越低，这恰好是在快速开发项目中所需要的。

螺旋模型提供至少和瀑布模型一样多或更多的管理控制。该模型在每个迭代过程结束前都设置了检查点。模型是风险导向的，对于无法逾越的风险是可以预知的。如果项目因为技术和其他原因无法完成，可以及早发现，这并不会使成本增加太多。

螺旋模型比较复杂，需要有责任心、更加专注，并具有管理方面的知识，通过确定目标和可以验证的里程碑，来决定是否启动下一轮开发。在有些项目中，产品开发的目标明确、风险适度，就没有必要采用螺旋模型提供的适应性和风险管理。

6．统一过程模型

统一过程模型吸取已有模型的优点，克服了瀑布模型过分强调序列化和螺旋模型过于抽象的不足，总结了多年来软件开发的最佳经验。其优点如下：

- 迭代化开发，提前认识风险。
- 需求管理，及早达成共识。
- 基于构件，搭建弹性构架。
- 可视化建模，打破沟通壁垒。
- 持续验证质量，降低缺陷代价。
- 管理变更，有序积累资产。

统一过程模型在此基础上，通过过程模型提供了一系列的工具、方法论、指南，为软件开发提供了可操作性指导，使开发者能较容易地按照预先制订的时间计划和经费预算，开发出高质量的软件产品以满足用户的最终需求。它是以用例驱动的、以构架为中心的、风险驱动的迭代和增量的开发过程。

如图 1-15 所示，在统一过程模型中，其横向按时间顺序来组织，将软件开发周期分成 4 个阶段，并以项目的状态作为开发周期阶段的名字：初始、细化、构造和移交。每个阶段目标明确。每个阶段的结束都有一个主要里程碑，如图 1-16 所示。实质上，每个阶段就是两个主要里程碑之间的时间跨度。在每个阶段结束时进行评估（生命周期里程碑审核），以确定是否实现了此阶段的目标。良好的评估可使项目顺利进入下一阶段。每个开发周期都将给用户提供产品的一个新版本，称为一个增量。在每个阶段，为了完成阶段目标可以进行多次迭代。每次迭代都要执行需求、设计、编码、测试、管理等多个软件开发中的主要活动。为了区别于瀑布模型，统一过程模型把软件开发中的这些主要活动称作核心工作流，以使人们能明确地认识到所有阶段中活动的连续性。

该模型的纵向按项目的实际工作内容——工作流来组织，如图 1-15 所示。工作流通常表示为一个内聚的、有序的活动集合。在统一过程模型中有以下 9 个核心工作流。

- 业务建模工作流：对整体项目业务建模。
- 需求工作流：分析问题空间并改进需求产品。

- 分析与设计工作流：解决方案建模并进化构架和设计产品。
- 实现工作流：编程并改进实现和实施产品。
- 测试工作流：评估过程和产品质量的趋势。
- 部署工作流：将最终产品移交给用户。
- 配置与变更管理工作流：统一化管理软件配置，并降低变更的损失。
- 项目管理工作流：控制过程并保证获得所有项目相关人员的取胜条件。
- 环境工作流：自动化过程并改进维护环境。

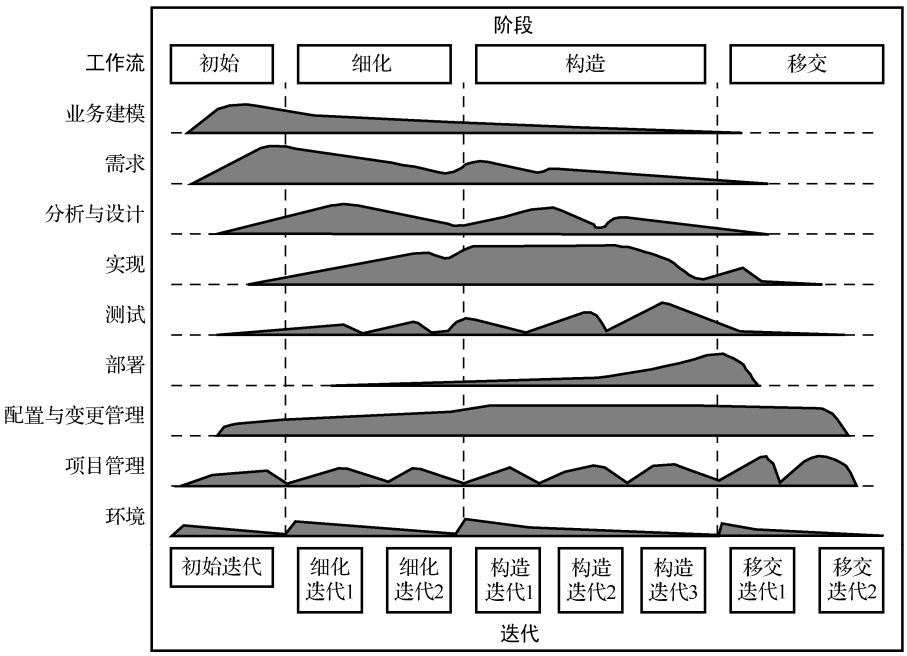

图 1-15　统一过程模型

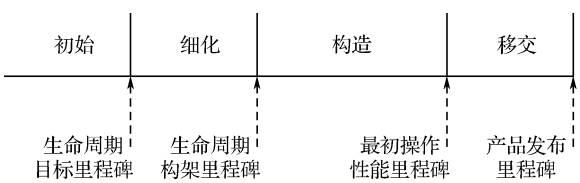

图 1-16　项目阶段和里程碑示意图

在整个生命周期中，涉及的所有重要活动都包含在这些工作流中，它们构成了项目的所有工作内容。但这些工作内容在不同阶段的相对工作量是不同的，如图 1-15 所示。

- 初始阶段：项目启动，确定生命周期目标里程碑，主要明确系统“做什么”。活动主要集中在需求工作流中，有少部分工作延续到分析与设计工作流中。该阶段的工作几乎不涉及实现和测试工作流。
- 细化阶段：构造构架基线，确定生命周期构架里程碑。虽然该阶段前期的活动仍然着重于实现需求，但分析和设计工作流中的活动更趋于活跃，为构架的创建打下基础。为了建成可执行的构架基线，有必要包含实现和测试工作流中的一些活动。
- 构造阶段：形成系统的初步可运行能力，确定在用户环境中初步运行的软件产品，

即确定最初操作性能里程碑。在此阶段，需求工作流中的活动趋于停止，分析工作流中的活动也减少了，大部分活动属于设计、实现和测试工作流。

- 移交阶段：完成产品发布，即确定产品发布里程碑。在用户的运行环境中安装并运行软件系统。该阶段工作流的混合程度依赖于验收测试或 β 测试的反馈。如果 β 测试没有覆盖实现中的缺陷，则重复进行的实现和测试工作流中的活动会相当多。

通常，软件系统在其生命周期中要经历几个开发周期。其中，第一个开发周期交付的第一个增量版本是最难开发的。它奠定了系统的基础及系统的构架，是对有可能存在严重风险的新领域的探索。一个开发周期随着它在整个软件开发生命周期中所处位置的不同有着不同的内容。在后期版本中，如果系统构架有了较大变化，则意味着开发过程早期阶段需要做更多工作。但如果系统最初的构架是可扩展的，那么在后期版本中，新的项目只是建立在已有产品的基础之上，也就是说，产品的后期版本将建立在其早期版本基础之上。

软件开发实践表明，在每个开发周期的早期解决问题往往比把问题留到晚期去解决更为有利。因此使用“迭代”的方法来获得初始阶段和细化阶段中的问题解决序列，以及构造阶段中的每个构造序列。

统一过程模型的迭代和增量是风险驱动的，但风险的到来并没有任何明显的提示，必须识别风险、限定风险范围、监控风险状况，并尽可能地降低风险。最好首先处理最重大的风险。同时，还必须仔细考虑迭代的顺序以首先解决最重要的问题。总之，要先做最难做的事。

在统一过程模型中，明确体现了：

- 计划、需求和构架以明确的同步点一起进化。
- 风险管理，以及如何客观地度量进展和质量。
- 借助提高功能的演示使系统能力得以进化。

7．敏捷过程模型

敏捷过程强调短期交付、用户的紧密参与，强调适应性而不是可预见性，强调为满足当前的需要而不考虑将来的简化设计，只将最必要的内容文档化，因此也被称为“轻量级过程”。敏捷开发保留了基本的框架活动：沟通、策划、建模、构建和部署，但将其缩减到一个推动项目组朝着构建和交付方向发展的最小集。

敏捷联盟定义了“敏捷”需要遵循的 12 条原则：

（1）最优先要做的事是，通过尽早和持续交付有价值的软件使用户满意。

（2）欢迎需求的变更，即使在软件开发的后期也是如此。敏捷过程利用项目需求变更为用户提升市场竞争优势。

（3）频繁地向用户交付可运行的软件产品。从几周到几个月，交付的时间间隔越短越好。

（4）在整个项目开发周期中，业务人员和开发团队应该天天在一起工作。

（5）围绕有积极性的个人来构建项目，为他们提供所需的环境和支持，并信任他们能够完成工作。

（6）在开发团队内部，效率最高、成效最大的信息传递方法是面对面地交流。

（7）可运行软件是进度的首要度量标准。

（8）敏捷过程提倡可持续开发。

（9）不断地关注优秀的技能和优秀的设计会增强敏捷能力。

（10）简单——把不必要的工作最小化的艺术——是根本。

（11）最好的构架、需求和设计源自组织团队。

（12）每隔一段时间，团队应该反省如何才能更有效地工作，并相应调整自己的行为。

并不是每个敏捷过程模型都同等地使用这 12 条原则。一些模型可以选择忽略或淡化一条或多条原则的重要性。

1.2.3 软件过程适应

预定义的软件开发生命周期、软件产品生命周期及单个软件过程通常需要被修改以更好地满足本地需求。组织环境、技术创新、项目规模、产品关键性、法规要求、行业惯例和企业文化可能决定需求的适应性。对单个软件过程和软件生命周期模型（开发和产品）的适应可能包括在软件过程、活动、任务和程序中添加更多细节，以解决关键问题。它可能包括使用一组替代的活动来达到软件过程的目的和结果。适应可能还包括从开发或产品生命周期模型中删除一些软件过程或活动，因为它们显然不适合要完成的工作范围。

1.2.4 实践考虑

在实践中，软件过程和活动常常是交叉、重叠和同时应用的。定义离散软件过程的软件生命周期模型，具有严格指定的输入/输出标准和规定的边界及接口，应该被认为是一种理想化情况。很多时候，必须对其加以调整，以反映组织环境和业务环境中软件开发和维护的现实情况。

另一个实践的考虑因素是：软件过程（如配置管理、构造和测试）应该可以调整，以方便软件的操作、支持、维护、迁移和退役。

在定义和裁剪软件生命周期模型时，需要考虑的补充因素包括：所需标准、指令和策略的一致性，用户需求，软件产品的临界性，组织的成熟度和能力。其他因素还包括工作的性质（如现有软件的修改与新开发软件的关系）和应用领域（如航空航天与酒店管理）。

1.3 软件过程评估与改进

软件过程评估用于评估软件过程的形式和内容，可以通过一套标准集来指定。在某些情况下，使用“能力评估”和“绩效评估”来代替过程评估。能力评估通常由购买方（或潜在的收购者）或外部代理（代表购买方或潜在的收购者）进行，其结果用于指示供应商（或潜在的供应商）所使用的软件过程是否为购买方所接受。绩效评估通常在组织内进行，以确定需要改进的软件过程，或者确定流程是否满足给定水平的过程能力或成熟度的标准。

过程评估是在整个组织、组织内单位和单个项目中执行的。评估可能涉及待评估软件过程的输入/输出标准是否被满足，如何评估风险因素和进行风险管理，或如何吸取经验教训。过程评估采用评估模型和评估方法进行。该模型可以为组织内部和组织之间的项目比较提供一个基准软件过程规范。

另外，过程评估与过程审计不同。评估是为了确定能力或成熟度级别，并确定要改进

的软件过程。审计通常是为了确定是否符合政策和标准。审计为组织内执行的实际操作提供了管理可视性，以便在影响开发项目、维护活动或软件相关主题的问题上，做出准确和有意义的决策。

组织内部软件过程评估和改进的成功因素包括管理赞助、计划、培训、有经验和有能力的领导者、团队承诺、期望管理和变更代理的使用，再加上项目试点和工具试验。额外的因素包括评估的独立性和评估的及时性。

1.3.1 软件过程评估与改进模型

软件过程评估模型通常包括软件过程的评估标准，这些过程被认为是良好的实践。这些实践可能只处理软件开发过程，也可能包括软件维护、软件项目管理、系统工程或人力资源管理等主题。

软件过程改进模型强调持续改进的迭代周期。软件过程改进周期通常包括测量、分析和更改的子过程。计划—行动—检查—执行（Plan-Do-Check-Act）模型是一种众所周知的软件过程改进的迭代方法。其改进活动包括：确定和优先考虑所需的改进（计划）；引入改进，包括变更管理和培训（行动）；与以前或示范的软件过程结果和成本相比较，评估改进情况（检查）；做进一步的修改（执行）。该模型可用于改进软件过程，以增强缺陷预防。

1.3.2 软件过程评估方法

软件过程评估方法可以定性或定量。定性评估依赖于专家的判断；定量评估通过对客观证据的分析为软件过程打分，这些客观证据表明已确定的软件过程的目标和结果的实现情况。例如，对软件检查过程的定量评估，可以通过检查遵循的程序步骤和获得的结果，加上有关发现缺陷的数据，以及与软件测试相比发现和修复缺陷所需的时间来进行。

软件过程评估的典型方法包括计划、事实调查（问卷调查、访谈和观察工作实践）、收集和验证过程数据、分析和报告等。软件过程评估依赖于评估人主观的、定性的判断，或者客观存在或未定义的工件、记录和其他证据。

根据软件过程评估的目的，在软件过程评估期间执行的活动和评估活动的工作分配是不同的。软件过程评估可以用于开发对软件过程改进提出建议的能力级别，或者用于获得软件过程成熟度级别，以便获得合同或授予的资格。

软件过程评估结果的质量取决于软件过程评估方法、获得数据的完整性和质量、评估团队的能力和客观性，以及评估过程中审查的证据。软件过程评估的目标是获得洞察力，确定一个或多个软件过程的现状，并为软件过程改进提供基础；通过遵循一致性检查表来执行软件过程评估。

1.3.3 连续式和阶段式软件过程评估

软件过程的能力和软件过程的成熟度通常使用 5 或 6 个级别进行评级。连续式评级表示为每个软件过程分配一个评级；阶段式评级表示为指定流程级别内的所有软件流程分配相同的成熟度级别。表 1-1 给出了软件过程评估等级表。连续式评级表示通常有第 0 级，阶段式评级表示则没有。

表 1-1　软件过程评估等级表

级　　别	连续式评级表示 能力级别	阶段式评级表示 成熟度级别
0	不完整级	
1	已执行级	初始级
2	已管理级	可重复级
3	已定义级	已定义级
4		已管理级
5		优化级

在表 1-1 中，第 0 级表示软件过程未完全执行或不能执行。在第 1 级，软件过程（能力级别 1 级或者成熟度级别 1 级）正在执行，是在临时、非正式的基础上进行的。在第 2 级，软件过程（能力级别 2 级或者成熟度级别 2 级）正在以一种方式执行，该方式为中间工作产品提供管理可视性，并且可以对软件过程之间的转换施加一些控制。在第 3 级，软件过程（能力级别 3 级或者成熟度级别 3 级加上成熟度级别 2 级）已明确定义（可能在组织策略和项目过程中），并在不同项目中重复。能力级别 3 级或成熟度级别 3 级为整个组织的软件过程改进提供了基础，因为软件过程（或流程）以类似的方式进行。这允许跨多个项目以统一的方式收集性能数据。在第 4 级（成熟度级别 4 级），定量的测量方法可以用于过程评估，也可以采用统计分析方法。在第 5 级（成熟度级别 5 级），应用了持续过程改进的机制。

软件过程评估的连续式评级表示和阶段式评级表示可用于确定软件过程改进的顺序。在连续式评级表示中，不同软件过程的不同能力级别为确定软件过程的改进顺序提供了指导。在阶段式评级表示中，在成熟度级别内满足一组软件过程的目标是为该成熟度级别定制的，这为在下一个更高成熟度级别上改进所有软件过程奠定了基础。

1．连续式评级表示

（1）不完整级。不完整级的流程是未执行或部分执行的流程。因为无法满足流程领域的一个或多个特定目标，以及没有将部分执行流程进行制度化，所以连续式表示能力级别 0 级没有一般目标。

（2）已执行级。已执行级的流程是一个能完成产出工作产品所需工作的流程，流程领域的特定目标都被满足。由已执行级所导致的重大改善可能会随着时间推移而失去，因为它们没有被制度化。应用制度化（已管理级和已定义级的一般执行方法）可以确保维持改善。

（3）已管理级。已管理级的流程是一个已执行的流程。它会根据政策规划与执行流程，任用具备技能的人员，并给予足够的资源以产出可控制的产品；纳入相关的关键人员；进行监督、控制及审查；评估遵循流程说明的程度。已管理级所反映的流程规范，确保现有的执行方法都在有压力的情况下，仍维持运作。

（4）已定义级。已定义级的流程是一个已管理级的流程。流程根据组织的指引调试组织标准流程，流程说明需要维护，并将流程相关经验纳入组织流程资产。已管理级与已定义级间的重要差异在于标准、流程说明与程序的范围。在已管理级中，每个流程特定案例中的标准、流程说明与程序都可以有相当的差异；在已定义级中，项目的标准、流程说明