大学计算机与新一代信息技术

云课堂版

安徽省高等学校计算机教育研究会推荐用书
新时代大学计算机课程分级分类教学创新系列教材
新形态数字化教材　丛书总主编 胡学钢 王 浩

主编　接 标 王秀友 蔡庆华 常雷琴

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

进阶篇　算法与数据管理

模块 8　软件开发与过程管理

案例导入

模块 8 资源

在当今计算机技术迅速发展、软件应用广泛普及的背景下，软件开发已经从最初的简单程序逐渐演变为构建复杂软件系统的过程。这不仅仅涉及程序员编写的代码，更是一个经过精心设计和管理的软件开发过程。然而，随着这一领域的蓬勃发展，所面临的挑战也与日俱增，包括项目延期、质量问题和项目成本超支等，合称为软件危机。

一个具体的例子发生在波音公司的“星际客船（Starliner）”的首飞中。Starliner 是为美国国家航空航天局（National Aeronautics and Space Administration，NASA）的商业载人计划设计的，旨在将宇航员送往国际空间站。然而，在 2019 年进行的首飞测试中，用来控制飞船活动的自动计时器错估了任务阶段，导致飞船提早消耗了过多燃料。地面控制中心曾试图发出指令覆盖计时器的程序，但由于又出现了通信延迟，该指令最终没能追上燃料的消耗速度，与空间站的对接任务因此搁浅，未能完成原定的任务目标。

为了克服软件危机，软件工程迅速崭露头角。软件工程致力于将工程化的原则、方法和最佳实践应用于软件开发过程，以提高开发的可控性、可预测性和可管理性。通过软件工程的方法，开发人员能够更好地理解用户需求，进行系统化的需求分析和设计，有效地进行项目管理和团队协作，并采用适合的开发模型和工具来提高开发效率和质量。

在本模块中，我们将深入探讨软件开发过程工程化的基本概念、原则和方法。首先，我们将了解软件危机的背景和原因，以及软件工程的发展历程。然后，我们将介绍软件生命周期的不同阶段和常见的开发模型，以及软件开发项目管理的关键要素和实践。此外，我们还将深入讨论软件开发过程中的过程管理、人员管理和制品管理等重要方面。最后，我们将介绍一些常用的软件工具和开发环境，以支持软件开发项目的需求和目标。

通过本章内容的学习，一方面，使读者了解软件开发过程工程化的核心概念和方法，从而能够应对软件开发中的各种挑战和问题；另一方面，使读者了解一些实用的工具和技术，以提高软件开发的效率和质量。

化的结构化软件开发方法学（structured development methodology）；开始采用定量的软件工程方法来指导软件开发、管理和质量保证等。

20 世纪 80 年代，软件工程逐渐发展为一门独立的学科，并在大学中设立了软件工程专业。同时，随着个人计算机出现，软件系统规模和复杂性不断增长。在结构化方法学、瀑布模型等成果的基础上，软件工程的研究与实践进一步发展，出现了一些软件工程的标准和规范。面向对象程序设计技术开始出现并流行，诞生了多种面向对象程序设计语言。

20 世纪 90 年代，互联网应用开始出现，软件经历了从单一计算环境转变为基于局域网的分布式计算环境。对于软件工程的研究和实践更为深入，面向对象方法成为软件开发的主流方法之一，提供了更加灵活和可重用的软件开发方式，促进了软件的模块化和可维护性；各种轻量级软件开发方法被提出，提高了开发团队的灵活性和适应性，适应了快速变化的需求和市场；提出了软件设计模式的概念和思想，开始关注软件体系结构的研究和设计技术，并应用于软件开发实践。

21 世纪前 10 年，互联网蓬勃发展，软件开发活动也越来越多的在互联网上开展。这一时期软件工程实践逐渐成熟，并呈现出多样化的发展趋势。主要进展和成果如下：敏捷方法的兴起，强调快速交付、灵活性和团队合作，帮助企业应对市场的快速变化和竞争的压力；研究与实践面向服务的软件工程；为了解决软件系统与社会日益融合所带来的安全性、私密性等问题，开展了软件可信技术的研究与实践。

近年来，随着云计算、开源软件实践和自动化技术的发展，DevOps 和持续交付成了软件工程中的重要概念。DevOps 和持续交付的理念与实践有助于解决传统软件开发中的一些挑战，如开发和运维之间的协作问题、交付周期的延长、人工操作的错误等。通过促进团队协作、自动化和持续改进，DevOps 和持续交付提供了一种更加敏捷、高效和可靠的软件交付方式。它们被广泛应用于云计算、Web 应用开发和大规模软件系统等领域。

总地来说，软件工程经历了从问题认知到学科确立、方法演进和实践成熟的过程。随着技术和市场的不断变化，软件工程领域仍然在不断发展和创新，以应对新的挑战和需求。

思考题

软件工程可以解决软件危机吗？

8.2 软件生命周期与软件过程模型

软件生命周期是指软件从概念到退役的整个过程，涵盖了需求分析、设计、开发、测试、部署和维护等阶段。软件过程模型则是在软件开发过程中采用的一种组织和管理方法，用于指导和规划开发活动。

8.2.1 软件生命周期概述

软件生命周期模型描述了软件在其整个生命周期中经历的各个阶段，包含需求分析、软件设计、软件实现、软件测试、软件交付和软件维护等阶段，如图 8-4 所示。每个阶段会产生不同的软件制品，同时不同阶段之间存在相关性。

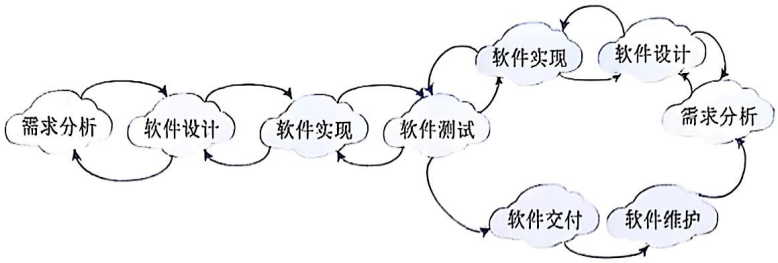

图 8-4　软件生命周期模型

1. 需求分析

需求分析阶段的成功与否对于后续的软件开发过程至关重要。准确理解和明确需求可以避免后期的变更问题，提高软件开发的效率和质量。在这个阶段，软件团队首先要与用户、客户、项目经理和其他利益相关者沟通，收集软件系统的需求信息，包括功能性需求和非功能性需求。软件团队需要对收集到的需求进行分析和整理，确保需求的一致性、完整性和可行性。这包括识别需求之间的关系和优先级、识别潜在的冲突或风险、将需求转化为明确、可验证的文档等。其中文档包括软件需求说明书、软件规格说明书、软件确认测试计划、用户手册等，用于指导后续的设计、开发和测试工作。当完成文档编写工作后，软件团队可以通过原型演示、用户反馈和需求评审等方式，与利益相关者共同审查和确认相关文档，确保可以准确反映用户的期望和系统的功能。

2. 软件设计

软件设计可以划分为概要设计和详细设计两个阶段，这种划分有助于更好地组织和管理设计过程。概要设计阶段关注系统的整体结构和组成部分的高层次设计。概要设计的目标是为详细设计提供一个基础框架，指导后续的设计和开发工作，通常以高级设计文档或设计说明书等软件制品的形式呈现。详细设计阶段关注系统的具体实现细节和算法设计。详细设计阶段的输出通常是详细设计文档、类图、时序图、数据库模式等软件制品。概要设计和详细设计相互关联，概要设计提供了整体架构和设计原则，详细设计则在概要设计的基础上进行具体的实现细节规划。这两者结果的好坏将直接决定软件系统的质量水平。

3. 软件实现

软件实现是将软件设计转化为实际的可执行代码。在软件实现阶段，软件团队首先会根据项目需求选择和使用合适的程序设计语言。其次，开发团队会根据概要设计和详细设计的指导，进行编码活动。在编码过程中，开发人员应遵循编程规范和最佳实践，确保代码的可读性、可维护性和可扩展性。软件实现阶段需要严格遵循软件设计的指导，保证实现的正确性和质量。同时，开发团队需要进行有效的版本控制和代码管理，以便跟踪和管理代码的变更和版本发布。这样可以确保软件实现的可靠性和可维护性，并为软件交付和部署做好准备。

4. 软件测试

软件测试的主要任务是发现潜在的缺陷和问题，并确保软件系统按照预期工作。软件团队需要制订测试计划，明确测试的目标、范围、资源需求、时间计划和测试策略等。软件测试有多种类型，包括单元测试、系统测试、回归测试、验收测试等。软件测试是一个迭代的过程，测试团队需要根据测试结果和反馈进行调整和改进。自动化测试工具和技术可以提高测试效率和覆盖范围。良好的测试实践可以帮助提高找出软件缺陷的效率。测试

过程中需要记录和跟踪错误报告，并确保及时修复和验证。通过有效的软件测试，可以提高软件系统的质量、稳定性和用户满意度。

5. 软件交付

软件交付是将已完成的软件系统交给客户或最终由用户使用的过程。在进行软件交付之前，需要确保软件系统经过充分的测试，以确保其质量和稳定性。在软件交付阶段应该为用户使用软件提供清晰的文档支持，包括软件系统的用户手册、安装指南、配置要求、技术文档等。在进行软件系统的部署和安装时，软件团队需要根据软件系统的要求和客户环境，确保软件系统能够在目标环境中正确运行，并与其他系统或组件进行适当的集成。为确保用户能够正确使用软件系统并了解其功能和特性，软件团队可以为用户提供必要的培训。在软件交付后，可能需要一段过渡期来确保软件系统的稳定性和用户适应。在过渡期内，软件团队可以提供必要的支持和维护，及时解决用户反馈的问题，以提高用户满意度。

6. 软件维护

软件维护是指在软件交付后，对软件系统进行修复、改进和支持的一系列活动。软件维护的目标是确保软件系统的正常运行、持续可用和适应变化的需求。在软件使用过程中，软件团队可能会因为软件中存在的缺陷、用户需求的变化、移植到不同的运行环境等原因，对这些软件进行维护，以确保软件系统的稳定性和可靠性、满足用户的新需求和期望、确保软件系统在新的环境中能够正常运行和兼容等。在进行软件维护时可能会经历需求分析、软件设计、编码、软件测试和部署运行这样的周期，以确保维护工作的高效性和质量。随着软件系统的变化和改进，相关文档也需要进行更新和修订。软件团队会及时更新用户手册、技术文档和培训材料，以确保用户获得最新的信息和指导。软件维护可以延长软件系统的寿命，提高用户满意度，并确保软件系统持续地满足业务需求。

8.2.2 常见的软件过程模型

软件过程模型是一种组织和管理软件开发过程的框架或模式，它描述了软件开发过程中的活动、任务、角色和交付物之间的关系和流程。软件过程模型可以帮助团队规范开发过程、提高效率和质量。常见的软件过程模型有瀑布模型、原型模型、螺旋模型等。每个过程模型都有其独特的特点和适用场景，软件团队可以根据项目需求和团队能力选择最适合的模型。

1. 瀑布模型

瀑布模型是最传统的软件过程模型之一。图 8-5 是瀑布模型的示意图。它将软件开发划分为一系列线性的阶段，如需求分析、设计、编码、测试、运行维护等。每个阶段依次进行，前一阶段的输出作为下一阶段的输入。整体开发过程步骤与软件生命周期相一致。从图中可以看出，开发的整体流程类似于瀑布，故因此得名。

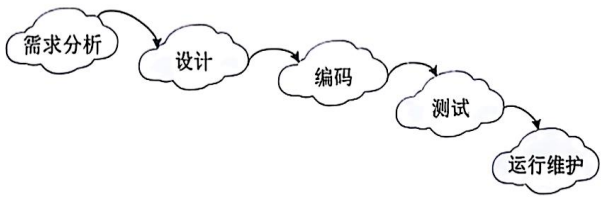

图 8-5　瀑布模型

瀑布模型整体清晰简洁，易于理解、掌握和使用。但其隐含了两项基本假设，一是软件开发活动完成后，不会出现其他问题；二是在需求分析阶段能够获取关于软件系统的完整软件需求。同时，瀑布模型的特点是阶段之间的线性流程和严格的阶段转换。每个阶段的输出作为下一个阶段的输入，而且通常每个阶段的工作是顺序进行的，即一个阶段完成后才能进入下一个阶段。因此，瀑布模型适用于需求稳定、项目规模较小且要求严格的项目。对于需求变化频繁的项目，可能不太适用。然而，瀑布模型仍然在某些特定的项目中得到应用，尤其是对于安全性和合规性要求较高的领域。

2. 原型模型

原型模型强调通过迭代开发快速构建原型以满足用户需求。在这种模型中，开发团队与用户紧密合作，通过构建原型来验证需求和设计。图 8-6 是原型模型的示意图。软件团队首先需要与用户和客户密切合作，收集系统需求和期望，理解用户的业务需求、功能要求和界面设计等。接着，根据收集到的需求，创建一个初步的系统原型。这个原型通常是一个简化的版本，展示系统的核心功能和用户界面。将原型展示给用户、客户和开发团队，并进行评估工作，收集他们的反馈和意见。根据原型反馈，对原型进行迭代开发。每个迭代周期都会增加新的功能或改进现有功能，以逐步完善原型。当用户认可原型所展示的软件需求时，可基于软件原型所反映的软件需求，开展后续系列的软件开发工作。

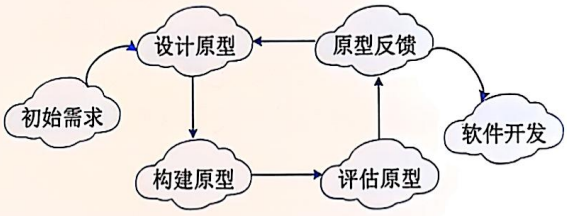

图 8-6　原型模型

通过使用原型模型，软件团队能够更早地获得用户的反馈，减少开发过程中的风险和错误，提高软件系统的可靠性和用户满意度。原型模型适用于需求不确定或需求易变的项目，可以快速验证和调整设计，提供更好的用户体验和系统质量。然而，由于注重快速迭代和用户反馈，原型模型可能会增加开发时间和成本。此外，在处理复杂系统和大型项目时，原型模型不够有效，需要结合其他开发方法和管理策略。

3. 螺旋模型

螺旋模型是一种面向风险管理的软件开发模型，它结合了瀑布模型的阶段性和原型模型的迭代性，通过周期性的评估和风险分析来指导开发过程。图 8-7 是螺旋模型的示意图。从图中可以得知整个软件开发分为多个周期，每个周期的开始任务是确定项目的目标、约束条件和需求。接着，进行风险识别和分析，对项目中的风险进行评估和管理。识别可能的风险和问题，并制订相应的解决方案和风险缓解策略，通常使用建造原型来排除风险。若不可以排除风险，则应当通过削减项目规模等方式降低风险，若可以排除风险，则启动软件系统的开发，开发阶段的工作过程相当于纯粹的瀑布模型。最后，对本阶段的工作成果进行评估和审查，并确定下一步的工作。

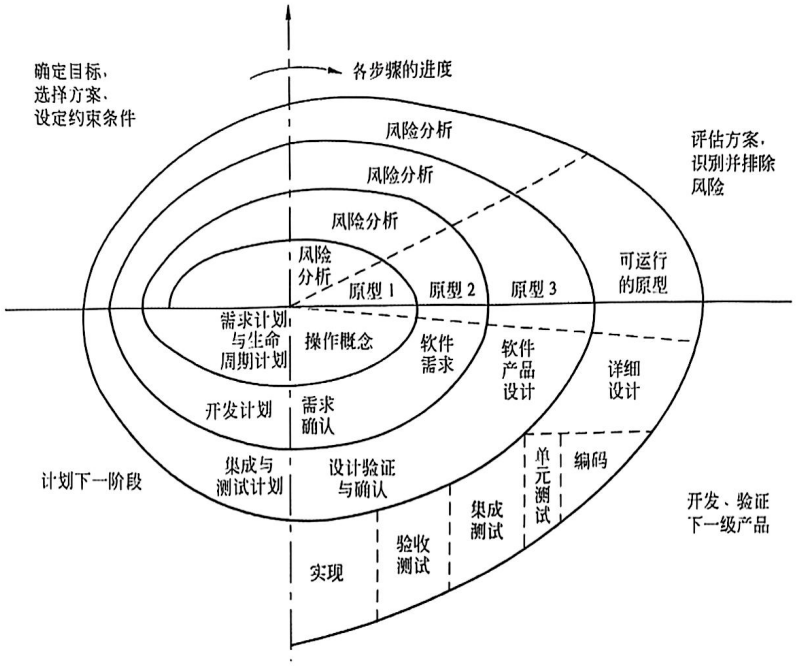

图 8-7　螺旋模型

通过不断地迭代和风险管理，螺旋模型可以逐步完善软件系统，并减少项目失败的风险。螺旋模型强调在开发过程中及时纠正问题和调整方向，以确保软件系统的质量和满足用户需求。螺旋模型适用于大型和复杂的软件项目，尤其是在需求不确定或变化频繁的情况下，它能够更好地应对项目中的风险和不确定性，并提供更灵活的开发方法。然而，螺旋模型的开发过程相对复杂，需要有经验的开发团队和有效的风险管理策略。

思考题

软件什么时候会彻底退役？

8.3　软件项目管理

在软件开发过程中，项目管理在确保项目按时、按质地交付，并满足客户需求和预期等方面起着关键的作用。本节将涵盖过程管理、人员管理和制品管理等项目管理的各个方面。

8.3.1　软件项目管理概述

软件项目通常由一个或多个软件开发团队组成，他们协同工作以实现项目的目标。软件项目的目标可以是开发一个新的软件产品，对现有软件进行改进或升级，或者解决特定的业务需求。软件项目管理是指在完成软件项目目标的过程中，对项目进行规划、组织、协调和控制的活动。它涉及对项目范围、时间、成本、质量、风险和资源等方面的管理，以确保软件项目能够按时、按质量、按预算地完成。

软件项目有明确的目标和交付物，如开发一个特定的软件应用程序或系统。

软件项目在时间、成本和资源方面都有限制。项目团队需要合理分配和管理这些资源，以确保项目的成功交付。