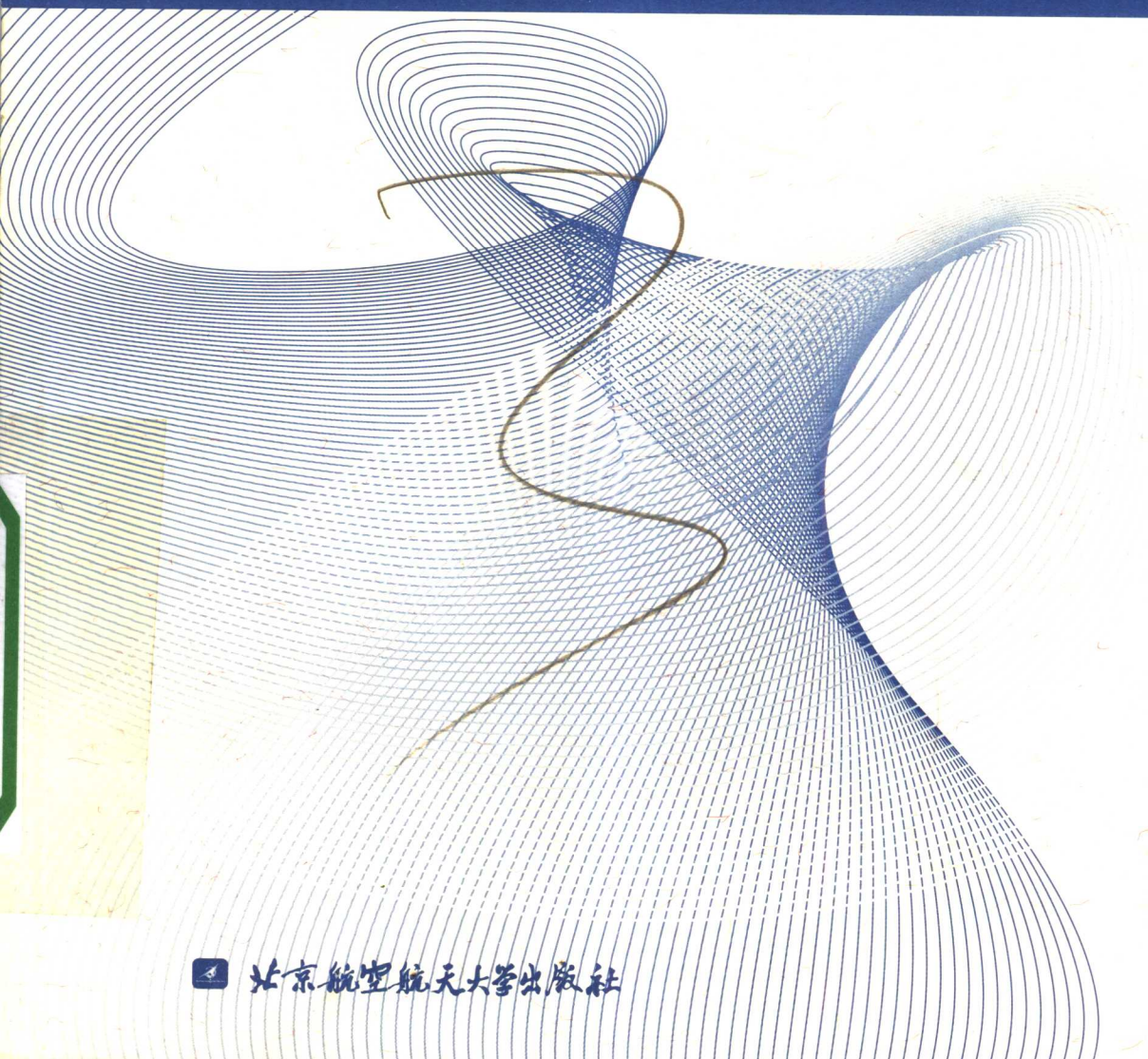
郭秋萍 焦 允 王全兰 著

大规模系统构架建模及其开发技术



北京航空航天大学出版社

大规模系统构架建模及其开发技术

郭秋萍　焦　允　王全兰　著

北京航空航天大学出版社

内容简介

从大规模系统构架建模的角度，研究基于构架/构件的系统开发方法及其相关技术。通过研究统一建模语言(UML)与形式化体系结构描述语言(ADL)的融合，建立一种基于构架/构件的大规模系统开发模型。主要内容包括：国内外研究现状；软件复用思想；系统构架理论与构架建模方法研究；软件构件的抽取、设计与实现方法；基于构架/构件的大规模系统开发模型；系统构架的评估等。可作为高校与软件工程相关专业或从事软件体系结构研究的高年级本科生、研究生、教师等的参考用书，也可为业内相关技术人员提供参考。

图书在版编目(CIP)数据

大规模系统构架建模及其开发技术/郭秋萍，焦允，王全兰著. —北京：北京航空航天大学出版社，2008.3

ISBN 978-7-81124-097-9

Ⅰ.大… Ⅱ.①郭…②焦…③王… Ⅲ.系统建模 Ⅳ.945.12

中国版本图书馆 CIP 数据核字(2008)第 019942 号

大规模系统构架建模及其开发技术

郭秋萍 焦 允 王全兰 著

责任编辑 许传安

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083)

发行部电话：010－82317024 传真：010－82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

北京宏伟双华印刷有限公司印装 各地书店经销

*

开本：690×960 1/16 印张：13 字数：232 千字

2008 年 3 月第 1 版 2008 年 3 月第 1 次印刷 印数：2 500 册

ISBN 978-7-81124-097-9 定价：26.00 元

前　言

面向对象方法的成熟，虽然为软件开发方式带来了一场技术上的变革，但基于网络的计算环境又为系统开发提出许多新课题。它要求系统实现跨空间、跨平台、跨用户的共享，导致软件在规模、复杂度、功能要求等方面的极大增长。基于构架/构件的系统开发，正是能充分满足这些新需求的有效方法。

本书从大规模系统构架建模的角度，研究基于构架/构件的系统开发技术。其主要内容均来源于河南省自然科学基金项目（“基于构架/构件的分布式多层信息系统建模及开发研究”，项目编号：0611052100）的研究成果。全书共分 8 章：第 1 章介绍了传统系统开发概况与基于构架的软件新认识观；第 2 章阐述了软件复用思想以及面向复用的系统开发方法；第 3 章深入探讨了系统构架理论，包括构架的构成要素、设计模型、形式化描述理论、体系结构风格等，重点研究了形式化描述与 UML 的集成建模；第 4 章为构件技术研究，包括构件模型、构件设计与实现、构件获取与评选、构件组装等；第 5 章为基于构件的系统建模与设计，包括基于用例的需求描述、基于 UML 的构件抽取和基于对象的构件设计等；第 6 章为主流构件实现规范介绍，包括 CORBA，J2EE，Microsoft COM/DCOM/COM＋等；第 7 章为基于构架/构件的大规模系统开发方法模型，包括系统开发生命周期中的构架建模、基于构架/构件的大规模系统开发方法以及系统构架模型的一致性保证等；第 8 章为系统构架的评估。

郭秋萍负责全书的总体组织、内容架构及定稿工作，并撰写了第 3 章和第 4 章；焦允承担具体研究工作，撰写了第 1 章、第 2 章、第 5 章和第 7 章；王全兰承担具体研究工作，撰写了第 6 章、第 8 章。廖建军、周新伟参与了一些章节的资料准备和文字校对工作。

本书的读者对象包括：与软件工程有关专业的本科生、硕士研究生、博士研究生，从事软件体系结构研究的学者或从事该领域相关研究的技术人员，以及其他对系统构架、软件构件感兴趣的读者。

在全书撰写过程中，参阅了大量文献资料，在此，谨向相关原著作者表示感谢。由于作者的学识与水平有限，书中难免存在不妥及疏漏之处，敬请读者批评指正。

作　者

2007 年 11 月

目　录

第1章

引 论

信息化是整合大规模企业各种主要业务的重要手段之一，能有效降低企业运行成本，提高管理效率。如今企业级应用系统正逐渐发展成在Intranet和Internet环境下的各种客户端可远程访问的分布式、多层次异构系统。尤其在电子商务、Web信息系统等领域，网络技术已成为现代企业管理的基础，并越来越深入地影响其未来走向[1]。

面向对象方法的成熟，虽然为软件开发方式带来了一场技术上的变革，但基于网络的计算环境又为系统开发提出许多新的课题。它要求软件实现跨空间、跨平台、跨用户的共享，导致软件在规模、复杂度、功能要求的极大增长，需要软件生产走异构协同工作、多层次集成、可反复重用的工业化道路[2]。

与此对应的是，传统软件开发过程不注重系统体系结构的总体规划，不注重利用以往软件开发所积累的知识和经验成果。一方面导致系统体系结构不优化，开发成功率低，适应性不强，难以扩展和维护；另一方面导致在分析、设计、编码、测试等软件开发各阶段重复劳动，极大降低系统开发效率，而且由于未充分重用现有的高质量软件，容易引入各种错误，从而影响软件质量。

为解决传统软件开发中存在的一系列问题，一种基于构架/构件的系统开发思想被提出，是目前企业级应用系统建设的主流方向[3]。在过去几年中，已产生了许多支持构架、构件的软件开发方法或相关工具，如RUP开发过程、UML建模语言、Rational ROSE建模工具等。本书从覆盖整个软件开发生命周期的角度，围绕软件开发的三个关键元素：过程、表示法和技术，探讨应该如何将其系统地加以应用，以一种系统化的方式有效地指导，并实现大规模软件的构件化开发。

1.1 传统软件开发方法概况

软件工程自提出以来，发展了许多有价值的软件开发模型，如较著名

的有瀑布模型、快速原型模型、螺旋模型、自动程序设计模型、增量模型和演化模型等。时至今日，这些开发模型仍然被各种应用软件系统开发所采用。

1.1.1 瀑布模型

瀑布模型是最常用的软件开发模型，如图 1.1 所示。

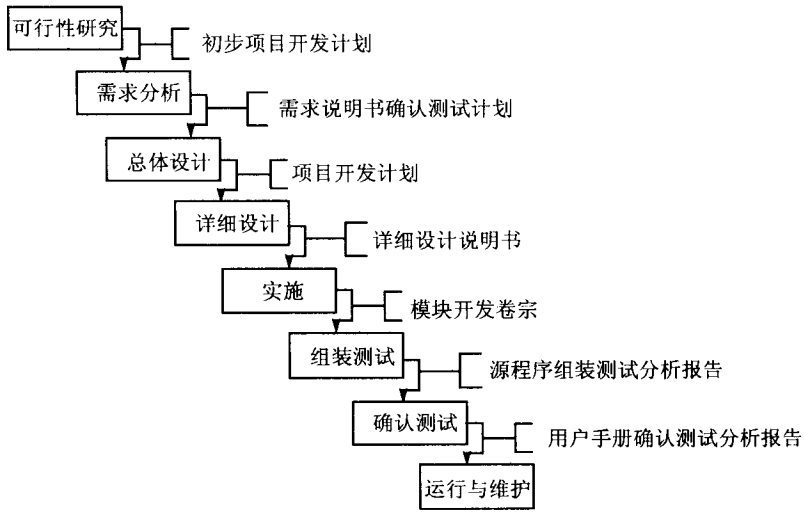


图 1.1 瀑布模型

该模型采用分阶段任务划分的方式，形如瀑布，似飞流逐级而下。其显著特点是：

1）各阶段间具有顺序性和依赖性，前一阶段的输出是后一阶段的输入；

2）推迟实现，即推迟具体的工程开发过程；

3）质量保证，每阶段均形成规范、齐全的文档，并进行审查、确认、管理复审和技术复审等，可以发现，并及时纠正当前阶段的错误，避免其在后续阶段的传递和放大。

在瀑布模型中，软件开发的各项活动严格按照线性方式进行。当前活动接受上一项活动的处理结果，并对之实施操作以完成本阶段的工作。此外，需要对当前活动的处理结果进行验证。如果验证通过，则该结果作为下一项活动的输入，继续进行下一项活动，否则返回修改。瀑布模型十分强调文档的作用，要求每个阶段都要对阶段性处理结果进行仔细验证。但是，这种模型的线性过程过于理想化，已不再适合现代软件的开发模式，几乎被业界所抛弃。

1.1.2 快速原型模型

快速原型模型适合用户需求不明确或欲快速了解用户需求的场合。它把软件开发分为两个阶段：原型开发阶段和目标系统实现阶段，如图1.2所示。

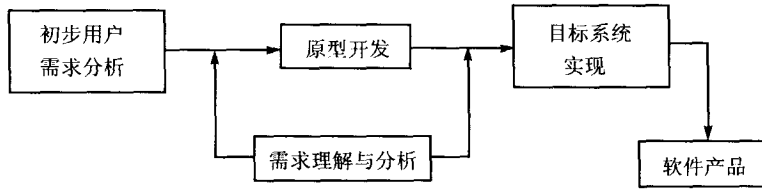


图1.2 快速原型模型

快速原型模型第一阶段的任务是快速开发一个原型。它包含目标软件的关键问题，反映目标软件的大致面貌，也可以反映目标软件开发的可行性，即开发目标软件的风险程度，从而可以正确做出下一步的决策。该阶段可通过加深对用户需求的理解，充分反映用户对系统的功能要求，为开发实用系统奠定可靠基础。第二阶段将依据前一阶段形成的界面，甚至部分算法、代码来构造实用系统。这种模型较适宜开发中小型软件。对大型软件来说，快速原型法可能会因为系统本身过于复杂而难以快速形成。

与瀑布模型相比，快速原型模型更符合人类认识问题和思考问题的过程，是目前较为流行的一种软件开发方法。在许多应用软件的开发过程中，由于客观上存在开发者和用户之间难以充分沟通的现象，开发者不能完全了解用户的确切需求，导致系统设计不能很好地反映实际需求。采用快速原型模型在开发者和用户之间架起了一条桥梁，可使系统建立有条不紊地进行。但对于结构复杂的大型应用系统来说，利用快速原型模型一般很难准确勾画出能够反映系统实际情况的构架模型，导致构造的原型模型与实际系统之间存在较大差异，因此不适合于大规模系统的开发。

1.1.3 螺旋模型

螺旋模型是在快速原型模型的基础上扩展而成的。这种模型把整个软件开发过程分成了多个由风险分析和原型开发组成的阶段。有人把这种模型也归为快速原型模型。螺旋模型的各个阶段均从确定阶段目标、制定可选方案和约束条件开始。然后评价可选方案，对风险进行识别和制定消除风险的办法，并开发出一个原型。接下来在此基础上开发和验

证下一级产品，同时制订下一阶段计划。最后用户对所得到的产品进行评审，检验是否符合要求。螺旋模型如图 1.3 所示。

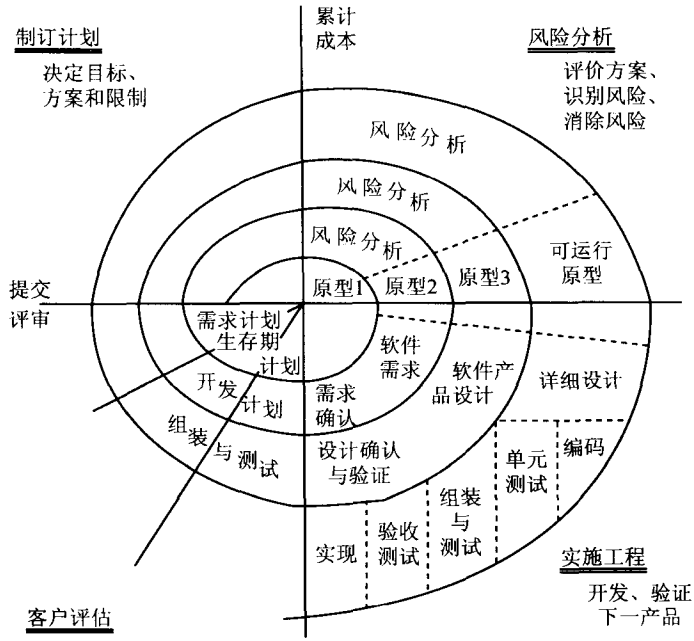


图 1.3 螺旋模型

由图 1.3 可知，螺旋模型沿着螺线旋转，在直角坐标系的 4 个象限上分别表达 4 个方面的活动。沿螺线自内向外，每旋转一圈便开发出更为完善的一个新软件版本。例如，在第一圈，确定了初步的目标、可选方案和约束条件之后，转入坐标系的右上象限，对风险进行识别和分析。如果分析表明，需求有不确定性，那么在右下象限内，所建的原型会帮助开发人员和用户，考虑其他开发模型，并对需求做相应修正。用户对产品做出评审之后，给出修正建议。在此基础上需再次确定目标，并进行风险分析。在每一圈螺线上，各种活动一直继续下去，自内向外，逐步扩展，最终得到所期望的系统。

1.1.4 自动程序设计模型

自动程序设计模型把软件开发流程分为两个基本阶段：软件规格说明开发和在机器辅助下自动转化为程序代码。该模型以形式化的软件规格说明为核心。软件的整个开发过程包括规格说明开发、规格说明确认、规格说明维护、交互式转换、自动编译与调整设计等阶段，如图 1.4 所示。

自动程序设计模型从问题的非形式用户需求开始，经形式化的软件

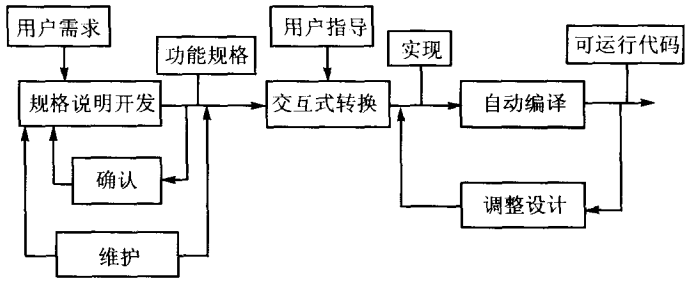


图 1.4 自动程序设计模型

功能规格说明、设计规格说明，最终到可执行的程序代码、系统调试、确认及交付使用。

1.1.5 增量模型

与建造大厦相同，软件也是一步一步构造起来的。在增量模型中，软件被作为一系列的增量元件来设计、实现、集成和测试。每一个元件是由多种相互作用的模块所形成的提供特定功能的代码段构成的。增量模型如图 1.5 所示。

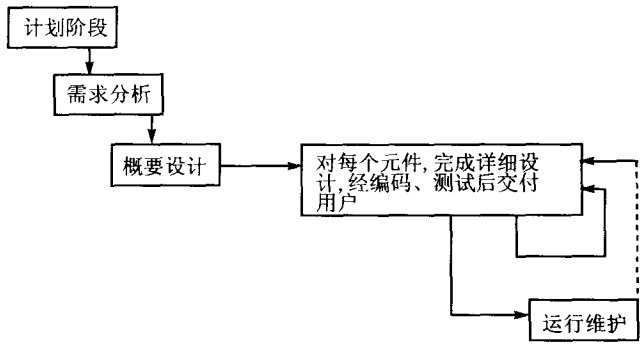


图 1.5 增量模型

增量模型在各个阶段并不交付一个可运行的完整产品，而是交付满足客户需求的一个子集的可运行产品。整个产品被分解成若干个模块，开发人员逐个模块地交付产品。这样做的好处是软件开发可以较好地适应变化，客户可以不断地看到所开发的软件，从而降低开发风险。但是，增量模型也存在以下缺陷：一是由于各个模块是逐渐并入已有的软件系统结构中，所以加入模块必须不破坏已构造好的系统部分，这需要软件具备开放式的体系结构；二是在开发过程中，需求的变化是不可避免的，增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速

原型模型，但也很容易退化为边做边改模型，从而使软件过程的控制失去整体性。

1.1.6 演化模型

演化模型主要针对事先不能完整定义需求的系统开发。用户可以给出待开发系统的核心需求，并且当看到核心需求实现后，能够有效地提出反馈，以支持系统的最终设计和实现。开发人员根据用户的需求，首先开发核心系统。当核心系统投入运行后，用户试用之，并提出精化系统、增强系统能力的需求。软件开发人员再根据用户的反馈，实施开发的迭代过程。每一个迭代过程均由需求、设计、编码、测试、集成等阶段组成，为整个系统增加一个可定义的、可管理的子集。在开发模式上采取分批循环开发的办法。每循环开发一部分功能，便成为这个产品原型的新增功能。于是，不断演化出新的系统。实际上，这个模型可看作是重复执行的多个“瀑布模型”。

以上各模型中，瀑布模型对用户需求要求最高，且不能随意改动。快速原型模型及螺旋模型可在开发过程中不断地完善用户需求，但滞后于用户需求的提出。自动程序设计模型可紧随用户环境及用户需求的变化而改变，开发出实际系统，但由于受多方面因素影响，例如，需求的正确理解、规格的标准化问题、自动编译系统的设计等，导致目前该模型还不是一种实用模型，缺乏典型的成功实例予以佐证。增量模型对非开放体系结构的软件适应性较差，且在开发过程中容易失去对过程的控制。演化模型未对演化的程度给出严格定义。

1.2 传统的软件认识观

长期以来，人们对应用软件及其开发过程逐渐形成了 3 种主要认识，分别是计算机解的观点、专业的观点和从零开始开发的观点。

1）**计算机解的观点** 这种观点认为，应用软件是客观世界中某种信息处理活动的计算机解，软件开发过程则是在计算机世界中的求解过程。该观点是由数十年的程序设计文化培植起来的，并且一直处于主导地位。究其原因主要是早期的应用软件大多被用于科学计算，软件设计和开发的目标是寻找合适的数据结构以及优化算法，程序设计被看成是一门艺术。计算机解的观点的典型代表是*软件＝程序＝算法＋数据结构*[4]。由于客观世界与计算机世界之间存在着巨大鸿沟，这种占主导地位的计算机解的观点不可避免导致应用软件系统与客观现实系统之间的差距。当今应用软件的范围越来越广泛、规模越来越庞大和复杂，而这种计算机

解的观点越来越显得与当今应用软件开发格格不入。

2）**专业的观点** 这种观点认为，系统开发过程是一个纯技术过程，是软件专业人员的开发过程，客户只负责提出系统需求。专业的观点是由计算机解的观点引伸出来的，强调应用软件的开发是计算机专业领域的事情，忽视了客户参与的重要性，不利于需求的正确获取、系统评审和最后验收。由于客户被置于开发过程之外，软件开发人员替代了客户的角色，开发者可以改动或新创用户业务范围内的任一成分和操作。其结果是，计算机专业人员开发的、自认为合理的应用软件并不一定是用户真正需要的。

3）**从零开始开发的观点** 这种观点认为，在传统的应用软件开发过程中，软件开发组织应该根据客户需求从零开始开发新系统。如典型的线性顺序模型，就是严格按照系统规划、需求分析、设计、编码、测试、实施与维护这一过程来开发的。从零开始的观点过于强调软件系统的特殊性，而忽略了不同系统之间可能存在的共性，这必然导致在开发过程中大量重复劳动、效率低下、可靠性不高等问题，无法促进应用软件的规模化生产。

1.3 支持复用的软件开发概况

在软件开发方法的发展过程中，目前的主要趋势是向着软件复用、构件集成等方向发展。下面分析几种有代表性的支持软件复用的系统开发方法。

1.3.1 Parnas 方法

最早的带有复用色彩的软件开发方法是由美国学者 D. Parnas 在 1972 年提出的。由于当时软件在可维护性和可靠性方面存在着严重问题，因此，Parnas 在其经典论文《论将系统分解为模块的评价准则》中给出一个信息隐蔽原则。核心内容是：在概要设计时列出将来可能发生变化的因素，并在模块划分时将这些因素放到个别模块的内部。这样，在将来由于这些因素变化而需修改软件时，只需修改这些个别的模块，其他模块不受影响。信息隐蔽原则不仅提高了软件的可维护性，而且也避免了错误的蔓延，改善了软件的可靠性。现在信息隐蔽原则已成为软件工程学中的一条重要原则。虽然 Parnas 对软件开发提出了深刻的见解，但遗憾的是，他没有给出明确的工作流程。所以 Parnas 方法不能独立使用，只能作为其他方法的补充。

1.3.2 面向对象的软件开发方法

面向对象的软件开发方法是软件技术的一次革命，在软件开发史上具有里程碑的意义。随着面向对象分析、面向对象设计、面向对象编程的发展，最终形成面向对象的软件开发方法。这是一种自底向上和自顶向下相结合的方法，而且它以对象建模为基础，从而不仅考虑了输入、输出数据结构，实际上也包含了所有对象的数据结构。所以面向对象的开发方法彻底改变了传统的软件开发过程。不仅如此，该方法还在需求分析、可维护性和可靠性这 3 个软件开发的关键环节和质量指标上有了实质性的突破，彻底地解决了这些方面存在的严重问题。

1.3.3 可视化开发方法

随着图形用户界面的兴起，用户界面在软件系统中所占的比例也越来越大，有的甚至高达 60%～70%。可视化开发就是在可视化开发工具提供的图形用户界面上，通过操作界面元素，诸如菜单、按钮、对话框、编辑框、单选框、复选框、列表框和滚动条等，由可视化开发工具自动生成应用软件。这类应用软件的工作方式是事件驱动。对于每一事件，由系统产生相应的消息，再传递给相应的消息响应函数。这些消息响应函数是由可视化开发工具在生成软件时自动装入的。可视化开发是软件开发方法上的一场革命。它使软件开发从专业人员的手中解放出来，对缓解 20 世纪 80 年代中后期爆发的应用软件危机有重大作用。

1.3.4 基于构件的软件开发方法

基于构件的软件开发方法 CBSD(Component Based Software Development)是软件复用思想在实践中的应用。早在 1968 年的北大西洋公约组织(NATO)的软件工程会议上就已提出建造可复用构件库的思想。构件及其较高的可复用性为人们展示了一种崭新的软件设计思路，以构件对象为中心的设计方法把硬件以芯片为中心的工艺思想恰如其分地融合于软件的面向对象分析、设计和实施之中，使面向对象的概念和方法从工具语言的层次一下子跃上了系统的应用层次。基于构件的软件开发的主要思想就是将现有的可复用构件集成到当前应用中来，使得应用无需从头开始，通过软件复用大大提高软件生产率。CBSD 能改变软件开发过程中的被动局面，使得开发者能够到构件市场购买所需构件便可组装应用程序，这将使软件产业发生革命性变化。

基于构件的软件开发方法是软件复用思想的最好体现。它将复用范围扩展到整个社会，真正实现软件开发社会化，是软件开发方式上的一次