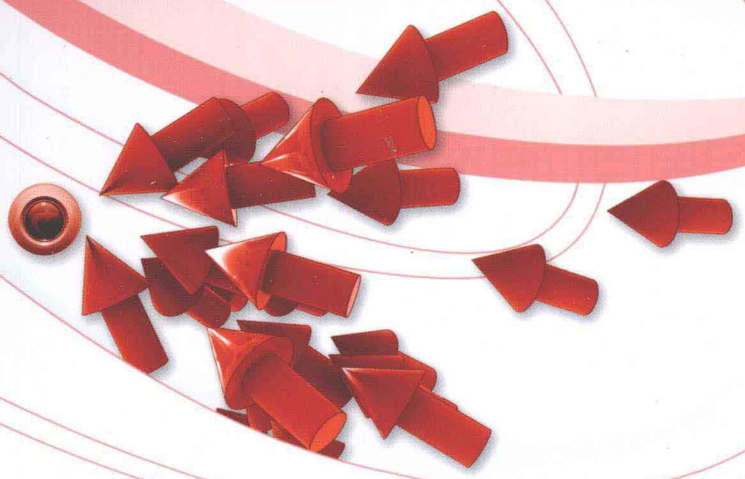
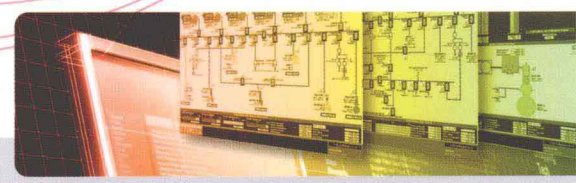
普通高等教育“十二五”规划教材

卓越工程师教育培养计划教学成果





高等学校计算机规划教材

软件开发环境与工具

相　洁　吕进来　主编

林福平　王会青　张　辉　李爱萍　副主编

陈俊杰　主审

電子工業出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

http://www.phei.com.cn

普通高等教育“十二五”规划教材
卓越工程师教育培养计划教学成果
高等学校计算机规划教材

软件开发环境与工具

相　洁　吕进来　主编
林福平　王会青　张　辉　李爱萍　副主编
陈俊杰　主审

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书以软件工程理论为指导，系统介绍软件开发过程中常用的工具。在保证教材内容的先进性和实用性的原则上，详细介绍系统分析设计工具 PowerDesigner、较流行的数据库管理系统与常用的数据库工具软件 SQL Developer、集成开发环境 Visual Studio、系统测试工具 LoadRunner、项目管理工具 Project 2007、软件配置管理工具 VSS。内容较为全面，覆盖了软件生产不同阶段的辅助开发工具以及软件过程管理工具。

本书包含两篇，第一篇基础篇，介绍软件开发过程的基础知识，重点介绍不同阶段的软件开发工具；第二篇案例篇，通过 3 个不同软件体系结构(单机应用程序、C/S 结构和 B/S 结构的网络应用程序)的综合案例，详细分析、介绍软件开发过程及相应的开发工具。本书通俗易懂，每章均有适当的习题，用来帮助读者巩固所学知识。本书配有 PPT、案例源代码、软件开发文档、习题答案等教学资源。

本书语言通俗，既有理论的概括与探讨，又有实际的经验方法总结。本书可作为高等院校计算机相关专业“软件开发环境与工具”课程的教材或教学参考书，也可作为软件工程实践课的教材，同时也可作为软件开发人员的学习和使用参考书。

图书在版编目(CIP)数据

软件开发环境与工具 / 相洁等主编. —北京：电子工业出版社，2012.5
高等学校计算机规划教材
ISBN 978-7-121-13667-2

I. ①软… II. ①相… III. ①软件开发－高等学校－教材 IV. ①TP311.52

中国版本图书馆 CIP 数据核字(2011)第 100341 号

策划编辑：史鹏举
责任编辑：史鹏举
印　　刷：
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：15.75　字数：455 千字
印　　次：2012 年 5 月第 1 次印刷
定　　价：29.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010) 88258888。

前　言

随着计算机软、硬件技术的高速发展，计算机软件应用领域越来越宽，软件系统规模越来越大，客户对软件开发周期的要求越来越高。软件工程理论随之逐渐成熟，软件开发技术也不断推陈出新，对软件开发人员的经验与能力的要求显著提高。

软件开发过程中，合理使用工具软件可以显著提高工作效率。一方面，工欲善其事，必先利其器，要求软件开发人员熟练掌握各种软件开发技术和相关工具，但是目前各类工具软件名目繁多，用途各异，给学生及软件开发人员带来更高的要求和诸多挑战。另一方面，通过对计算机科学专业、软件工程专业毕业学生的跟踪调查，多数学生认为本科学习期间虽然学习了很多与软件开发相关的课程，但是难以对软件开发建立一个全面的认识，不能充分认识到这些课程在软件开发中的作用和地位，导致缺乏这些课程学习的主动性和针对性。学生毕业后，仍然没有一套完整的软件开发思想和工具。针对这种现状，在教育部卓越工程师教育培养计划的支持下，我们组织了一批拥有丰富教学经验和软件开发实践经验的教师完成了本书的编写工作。

软件开发需要系统掌握软件工程、程序设计语言、算法分析与设计、操作系统、数据库原理、软件体系结构等相关课程的基本知识，软件开发工具只是辅助软件开发的工具，没有很扎实的基础理论指导，就不能很好地进行软件分析与设计、数据库管理、程序设计、软件测试与项目管理等工作，使用再好的工具也是徒劳。考虑到知识点衔接问题，本书适当增加了一些基础理论知识，便于读者能够在理解基本理论的基础上，在相关环节的软件开发过程中掌握软件工具的选择和使用方法，并能在后续的软件开发相关课程学习中主动使用这些工具。

与同类其他教材相比，本书具有以下特色：

- 系统介绍软件开发环境与工具的理论体系，突出软件开发工具的选择和具体使用方法的介绍，对工具的介绍并不是泛泛而谈；
- 软件开发工具类型齐全，覆盖软件开发各个环节，且全部是软件开发过程中的主流工具；
- 提供进行案例驱动教学的案例，这些案例代表了不同的软件体系结构和分析设计理论。

全书分为两篇，第一篇(基础篇)共 9 章，第二篇(案例篇)共 3 章。第 1 章主要介绍软件开发过程以及 CASE 工具的概念和分类；第 2 章和第 3 章介绍软件分析与设计基本理论，以及常用的分析与设计工具；第 4 章介绍数据库管理系统与数据库开发工具；第 5、6、7 章介绍软件集成开发环境 Visual Studio，在简要介绍 Visual C#语言基础之后，重点介绍使用如何 Visual Studio 进行应用软件开发；第 8 章介绍软件测试基础及测试工具；第 9 章介绍软件项目管理工具。案例篇通过 3 个案例介绍如何使用 CASE 工具辅助软件开发，这 3 个案例分属单机应用程序、C/S 结构和 B/S 结构的网络应用程序。教学时可以使用这 3 个案例进行案例驱动式教学。

本书语言通俗，既有理论的概括与探讨，又有实际的经验方法总结。本书可作为高等院校计算机相关专业“软件开发环境与工具”课程的教材或教学参考书，也可作为软件工程实践课的教材，同时也可作为软件开发人员的学习和使用参考书。

本书涉及的内容繁多，参考教学学时为 40～50 学时，授课教师可参照下表并使用本书配套资源完成教学任务。

教 学 内 容	学　时
第 1 章　软件概论	2
第 2 章　软件分析与设计基础	4
第 3 章　软件分析与设计 CASE 工具	4
第 4 章　数据库工具	4
第 5 章　Visual Studio 集成开发环境	2
第 6 章　Visual C#语言基础	4
第 7 章　Visual C#程序设计	8
第 8 章　软件测试基础与工具	4
第 9 章　软件项目管理与工具	2
第 10 章　案例 1：学籍管理系统	2
第 11 章　案例 2：基于 C/S 结构的聊天系统	2
第 12 章　案例 3：基于 B/S 结构的在线考试系统	2
实验	10

由于各学校教学计划有所不同，授课教师可以根据情况适当调整内容。如果已经完成软件工程导论的学习，可以减少第 2、8、9 章的学时，这些基础部分的内容可以安排学生自行复习回顾。

本书提供 PPT、案例源代码、软件开发文档、习题参考答案等配套教学资源，可登录华信教育资源网(www.hxedu.com.cn)，免费注册、下载。

本书第 1、4 章由林福平编写，第 2 章由武淑红编写，第 3、10 章由相洁编写，第 5 章由李爱萍编写，第 6 章由张辉编写，第 7、12 章由吕进来编写，第 8、9、11 章由王会青编写，全书由相洁统稿。本书由陈俊杰教授主审。书中部分案例由王白石等参与完成，在此表示衷心的感谢！

本书的出版得到了电子工业出版社的大力支持，在此表示诚挚的感谢！

由于作者水平有限，书中难免有疏漏、欠妥之处，敬请读者批评指正。

编　者

目　录

第一篇　基础篇

第一篇 基 础 篇

第1章 软 件 概 论

信息时代，软件无所不在。计算机系统的广泛应用，使得人们的生活、工作、学习及娱乐都离不开软件。

今天的人们能够亲身感觉到许多事情。人们出行时购买火车票、飞机票更加方便，城市里几乎遍布代售点。人们使用、管理财物更加方便了，ATM 机 24 小时工作并且随处可见。人们可以使用各种各样的卡来购物和支付交通费用，省去持币的不便和找零的费时，也可以通过网络管理自己的金融资产。人们上班时使用各种软件完成工作任务，工程师和设计师使用 CAD 等工具进行设计工作，职员使用办公软件进行文件资料编写。学校老师通过网络布置作业，学生通过网络提交作业，大家通过网络交流学习经验共享学习成果。人们在网络上听音乐、观赏电影、玩网络游戏的现象已经普遍化。所有这些都有赖于软件的广泛普及应用。

软件应用领域广泛，可以说几乎找不到一个不使用软件的领域。最初计算机的发明是为了计算炮弹弹道，然而今天的计算机及软件的应用程度，完全超越了当初人们的想像，尤其是互联网的出现，进一步扩大了计算机软件的应用范围。近年来由于大规模集成电路技术的不断进步，伴随着像手机这样的移动设备得以普及和广泛的使用，在这些移动设备上运行的软件也蓬勃发展。

1965 年，戈登·摩尔(Gordon Moore)提出的摩尔定律，至今似乎尚未过时。硬件方面，CPU 速度不断加快，内存容量不断提高，硬盘存储不断增大，网络通信技术的进步，以及手机等小型轻量移动设备的普及使用，特别是互联网技术的成熟与进步，使得软件的应用领域更加广泛，软件规模更加庞大，软件系统组成更加复杂。与此同时，软件系统中使用的计算机语言也趋于多样，由单一计算机语言开发的软件系统已经几乎难以找到。

计算机硬件和网络技术的进步，以及软件系统的广泛应用，导致软件系统复杂化、巨型化，也使得软件开发过程更加复杂，软件开发成本更高，软件的正确性和可靠性更加难以保障，软件项目管理难度更大，软件项目对开发人员的要求也更高了。与此同时，这样的挑战也带来了软件及其开发技术的进步。

1.1 软件基本概念

1.1.1 软件概念

众所周知，计算机系统是由硬件和软件组成的。通常硬件是计算机系统中所有物理组件的总称，完成计算机系统的基本功能，并实现了一个指令集。计算机指令集是编写计算机软件的程序员与硬件之间的接口界面，也称为机器语言。软件是为了特定目的，存储在计算机存储器中的计算机程序及相关数据的集合，这些程序由指令组成，告诉计算机做什么和如何做。另一种说法是，软件指程序、过程、算法和文档的集合。软件执行程序中实现的功能，无论是直接提供指令给计算机硬件运行或提供

服务将数据作为另一个软件的输入。不同于硬件(亦称物理设备)，软件是无形的。软件有时只是被单纯地理解为应用软件，有必要通过以下的软件相关的概念和实例，使我们对软件的认识更具体一些。

- 应用软件(Application Software)：包括最终用户的计算机应用，例如文字处理软件、视频游戏和为特定用户群开发的 ERP 软件。
- 中间件(Middleware)：用于控制和协调分布式系统。
- 编程语言(Programming Languages)：定义计算机程序的语法和语义。例如一些成熟的银行应用是 1959 年发明的 COBOL 语言编写的。新的应用通常用近期发明的编程语言编写。
- 系统软件(System Software)：包括操作系统，用于管理计算机资源。例如 Windows 操作系统是系统软件，负责管理计算机各类资源。
- 测试件(Testware)：用于测试硬件和软件的套件。
- 固件(Firmware)：底层软件，通常存储于电子可编程存储器上。固件如同其名，被看做硬件被其他软件调用执行。
- 设备驱动程序(Device Drivers)：建立软件与磁盘驱动器、打印机、光盘驱动器、计算机显示器等硬件设备的数据通信界面，使硬件之间能够方便地交换数据。
- 编程工具(Programming Tools)：用于上述软件的开发。程序员用它来调试程序，或对既有的系统进行逆向工程的分析和检查源代码兼容性。

1. 软件概述

计算机软件的说法是为了区别于计算机硬件。硬件实现物理的互连，并根据设备需要存储运行软件。在系统底层，特定的处理器执行机器语言指令的代码。机器语言是一组代表处理器指令的二进制值，可以改变计算机的状态。程序是指令序列，执行一个具体的指令序列将改变计算机的状态。程序通常用高级编程语言来书写，与机器语言相比接近于自然语言，更易于人们使用，效率也更高。高级语言被编译解释为机器语言的目标代码。早期软件也用汇编语言编写，是使用自然语言字母表的机器语言助记符。汇编语言程序必须通过汇编器组装成目标代码。

软件具有各种形式与作用，可以是数字化存储的可播放数据，可以是 CPU 可以执行的程序代码，也可以是其他信息。软件是范围广泛的产品，使用各种技术来开发，如普通编程语言、脚本语言、微指令(亦称微码)和 FPGA 配置等。

软件有网页软件和桌面应用软件之分。网页软件通常按照特定架构，使用 HTML、PHP、Perl、JSP、ASP.NET、XML 等语言开发。桌面应用软件有 OpenOffice、Office 等，使用 C、C++、Objective-C、Java、C#或 Smalltalk 等语言开发。应用软件通常运行在位于底层的操作系统软件(如 Linux 或 Windows)之上。软件或固件也用于视频游戏及汽车、电视和消费电子产品等的逻辑系统配置中。

2. 软件分类

计算机系统的软件主要分成三大类：系统软件(System Software)、编程软件(Programming Software)、应用软件(Application Software)。有些情况下，准确界定一个软件属于哪一类是困难的，但是这样的分类也是合理的。以往大多将编程软件归类于系统软件，这样的分类有助于深度理解软件及其作用。

(1) 系统软件

协助计算机硬件和系统工作，实现计算机的基本功能，包括以下组件：

- 设备驱动程序
- 操作系统

- 服务器
- 通用工具
- 视窗系统

系统软件负责管理各种各样相互独立的硬件组件，使之能够很好地协同工作。其目的是让程序员可以忽视复杂的特定计算机的细节进行编程。这些细节包括通信设备、打印机、数据写入设备、显示器和键盘等。同样也可以使程序员不用区分计算机的资源，如内存和处理器时间，进行安全稳定的开发工作。

(2) 编程软件

编程软件通常是提供给程序员的工具，通过该工具可以方便地使用不同的编程语言来编写计算机程序和软件。集成开发环境(Integrated Development Environment，IDE)是一个单独的应用软件，试图管理所有这些功能。编程工具软件包括：

- 编译器
- 调试器
- 解释器
- 连接器
- 文本编辑器

(3) 应用软件

应用软件种类繁多，用于完成某种计算任务。例如网页浏览器是一种应用软件，可以显示网页服务器或者文件系统的HTML文件内容，并让用户与这些文件交互。虽然严密准确地对软件进行分类是一件十分困难的事情，但还是需要给出一种分类。应用软件大致包括以下几大类：

- 商业软件
- 计算机辅助设计
- 数据库
- 决策软件
- 教育软件
- 图像编辑
- 工业自动化
- 数学软件
- 医疗软件
- 分子模拟软件
- 量子化学和固体物理软件
- 仿真软件
- 电子表格
- 远程通信
- 视频游戏
- 文字处理

(4) 软件的其他分类方法

基于软件工作方式，软件可分为实时处理软件、分时软件、交互式软件和批处理软件。

基于软件运行环境，软件可分为单机运行软件、网络运行软件和嵌入式软件。

基于软件体系结构，软件可分为C/S结构、B/S结构和多层架构。

3. 软件特点

软件特点大致可以归纳如下：

软件是一种逻辑实体。软件是抽象的、无形的，没有物理实体，但可以记录在介质上。软件必须通过测试、分析、思考、判断去了解它的功能、性能及其他特性。软件正确好坏与否，需要等到在机器上运行之后才能知道。这给软件的设计、生产和管理带来诸多困难。

软件是人类智力产品。软件是人们通过智力劳动，依靠知识和技术等手段生产的信息系统产品，是人类有史以来生产的高复杂度、高成本、高风险的工业产品。软件涉及人、社会和组织的行为和需求，涉及几乎所有领域的知识。

软件开发过程复杂。20 世纪 60 年代末 70 年代初爆发的软件危机，使人们充分清楚地认识了软件开发的复杂性。所有软件开发必须按照软件工程管理的方法进行，严格管理软件项目的进度、质量和成本。有必要使用有效的软件开发环境和工具，以提高软件开发效率。

软件需要长期维护。软件维护与硬件维修维护有着本质的差别，不能简单地通过更换部件来实现。在软件生命期中，需要随时对暴露出来的故障进行修改。随着社会及技术的变化进步，人的需求、社会的行为规范、组织的需求和业务流程、国家的法律等也会发生变化，随着这些变化都需要对既有软件进行修改和维护。

软件成本昂贵。由于软件应用范围广泛和需求复杂等原因，许多软件往往是一个巨型系统，需要投入大量的人力、物力和财力进行开发，导致软件成本昂贵。

软件可以复制。软件一旦开发成功，就不需要再制作，可以无限地复制同一内容的副本。所以软件质量必须在开发阶段得以控制。由于软件功能和性能可以通过修改而改变，因此软件通常有多种版本。

1.1.2 软件应用领域

今天的计算机及软件的应用程度，完全超越了当初人们的想像，尤其是互联网的出现，进一步扩大了计算机软件的应用范围。以下列举了若干应用领域：

- 金融(包括银行、证券和保险等)
- 企业(主要是制造业和服务业等)
- 物流(包括商业零售业及其物流系统)
- 政府
- 电信
- 能源
- 交通
- 医疗
- 农业

1.1.3 软件危机

20 世纪 60 年代末 70 年代初，软件经历了一场“软件危机”。软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。这些问题不仅是不能正常运行的软件才有，几乎所有软件都不同程度地存在这些问题。

1. 软件危机主要表现

软件开发成本和进度失控。实际成本比估计成本有时高出一个数量级，实际进度比预期进度拖延

数月甚至数年的现象并不罕见。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，因而难免引起用户的不满。

用户对软件产品(软件系统)不满意。软件开发人员常常在对用户要求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，便匆忙着手编写程序。软件开发人员和用户之间的信息交流往往很不充分，“闭门造车”导致最终的产品不符合用户的实际需要。

软件产品质量差。软件可靠性和质量保证的确切定量概念刚刚出现不久，软件质量保证技术还没有坚持不懈地应用到软件开发的全过程中，这些都导致软件产品发生质量问题。

软件产品可维护性差。很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需要在原有程序中增加一些新的功能。“可重用的软件”还是一个没有完全做到的、正在努力追求的目标，人们仍然在重复开发类似的或基本类似的软件。

软件缺少相应文档资料。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新的”(即和程序代码完全一致的)。这些文档资料可以作为软件开发“里程碑”，软件开发组织的管理人员通过这些文档来管理和评价软件开发工程的进展状况；文档资料也是软件开发人员之间的交流通信工具，可以在软件开发过程中准确地交流信息；对于软件维护人员而言，这些文档资料更是至关重要、必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

软件成本比例上升。由于微电子技术的进步和生产自动化程度不断提高，硬件成本逐年下降，然而软件开发需要大量人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而持续上升。软件成本占计算机系统的比例逐年上升。

软件开发生产率低。软件开发生产提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。软件产品“供不应求”的现象，使人类不能充分利用现代计算机硬件提供的巨大潜力。

2. 软件危机产生原因

软件的核心是逻辑，是看不见摸不着的。软件开发进程难以准确描述，软件质量也难以评价，因此管理和控制软件开发过程困难。软件故障(即 BUG)是开发编写程序时编写出来的，同时测试阶段没能检测出来的程序错误。软件维护意味着需要修改原设计，客观上造成软件维护困难。

软件的一个显著特点是规模庞大，然而程序复杂性与程序规模之间是指数增长关系。想要按计划开发完成规模庞大的软件，需要大量软件开发人员分工合作，如何保证协同工作完成一个高质量的大型软件系统，是一个极端复杂困难的问题。这不仅涉及诸如分析方法、设计方法、 形式说明方法、版本控制等技术问题，更需要严格科学的管理。

没有准确把握用户需求也是软件开发失败的主要原因之一。没有完整准确了解用户要求便匆忙着手编写程序，这是早期软件开发普遍存在的现象。没有做好软件定义就着手开发，往往造成编写的程序不符合需求，同一功能内容的编程工作多次重复，造成时间和成本的浪费。

一个软件从定义、开发、使用和维护到废弃，经历一个漫长的时期，通常称为生命周期。现在的软件工程技术明确详细给出了软件生命周期中每一步的要点。没有在软件工程知识的指导下进行软件开发也是失败的原因之一。

3. 软件危机解决方法

软件危机发生之后，人们重新审视了软件开发的每一个细节和过程，产生了新的认识。特别是改正了“软件即程序”的错误观念，清晰地认识了软件与程序的不同。人们研究出各种各样的软件开发

方法，把方法论作为解决问题的一种途径。同时人们也开始注重软件文档的作用，将软件文档作为软件的一个组成部分来看待。

解决软件危机的主要方法可以归纳为以下两点，一是软件工程方法和技术的广泛普及与应用，使得人们按照一定原则和方法来开发软件，从而保证了软件的质量并节约了成本；二是软件项目管理普遍运用于软件开发过程之中，使得软件按照标准进行开发，保证了软件的开发进度，提高了软件开发质量，降低了软件开发成本，也加强了对软件开发人员的有效管理。

此外，一些技术与方法的使用，也极大地改进了软件开发。这些技术包括计算机辅助软件工程(Computer Aided Software Engineering，CASE)、代码自动生成技术、开源软件、软件架构、软件测试技术等。

1.2 软件架构

今天，一个软件系统往往是一个大型系统，甚至是一个巨型系统。软件系统的软件架构(Software Architecture)是构成系统的结构集合，包括构成软件的基本元素及其关系，也包括系统软件架构的文档。文档化的软件架构有助于与软件拥有者进行交流，更早地确定高层次的设计，并在项目中重用这些设计组件和模式。

1.2.1 软件架构概念

软件大型化导致的复杂性问题是计算机科学研究的一个主要课题。早期通过选择好的数据结构和算法，化解复杂性问题。随着软件开发技术的提高，软件架构的基本原理自 1980 年起就已经零星地运用于软件工程之中。软件架构通过抽象和分解达到简化系统复杂性的目的。

软件架构是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。软件架构是指在一定的设计原则基础上，从不同角度对组成系统的各部分进行搭配和安排，形成系统的多个结构而组成架构，它包括该系统的各个组件，组件的外部可见属性及组件之间的相互关系。组件的外部可见属性是指其他组件对该组件所做的假设。

软件架构是一个系统的草图，所描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述组件之间的通信。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在面向对象领域中，组件之间的连接通常用接口来实现。

1.2.2 软件架构的历史

有关软件架构的概念，Edsger Dijkstra 在 1968 年，David Parnas 在 1970 年，在研究工作中就已经提及。这些科学家强调软件系统的结构问题及其正确性是至关重要的。研究机构在软件架构研究中，充当重要角色。卡内基·梅隆大学的 Mary Shaw 和 David Garlan 于 1996 年写了《软件架构：展望一门新兴学科》(*Software Architecture Perspective on an Emerging Discipline*)一书，提出了诸多软件架构相关的概念，例如软件组件、连接器、风格等。加州大学埃尔文分校的软件研究院所做的工作则主要集中于架构风格、架构描述语言以及动态架构。

计算机软件的历史开始于 1950 年，历史非常短暂。而建筑工程则从石器时代就开始了，人类在几千年的建筑设计实践中积累了大量的经验和教训。建筑设计基本上包含两点，一是建筑风格，二是建筑模式。独特的建筑风格和恰当选择的建筑模式，可以使一个建筑独一无二。人类在建筑学所获得的理论和经验无疑也适用计算机软件。一个现代建筑中最为重要的是地基和框架。一个软件中最为重要的是软件平台和软件架构。这里所说的软件平台是运行软件的硬件、网络通信环境和包括系统软件在内的基础软件。

1.2.3 软件架构目标

软件架构设计需要达到以下的目标：

- 可靠性(Reliable)。软件系统多用于商业经营和管理，用于为人、社会和组织提供服务，所以可靠性非常重要。
- 安全行(Secure)。软件系统多数承担重要的交易处理，所以系统的安全性非常重要。
- 可扩展性(Scalable)。软件必须能够在用户的数目和使用率快速增长的情况下，保持合理的性能，才能适应用户市场扩展的需求。
- 可定制化(Customizable)。同一套软件，应该能够根据不同的客户群体及市场需求的变化进行调整，以适应个性化的需求。
- 可扩展性(Extensible)。随着新技术的出现，软件系统应当能够导入新技术，便于对现有系统进行功能和性能的扩展。
- 可维护性(Maintainable)。软件系统的维护包括两方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的成本。
- 客户体验(Customer Experience)。软件系统必须易于使用，让用户简单方便地学习使用。
- 市场时机(Time to Market)。软件用户面临同业竞争，软件提供商同样面临同业竞争，好的系统架构有助于软件快速夺取市场先机。

1.2.4 软件多层架构

由于互联网技术的普及，现在软件系统大多采用三层架构设计开发。这种三层架构也是一种软件设计模式，是C/S(客户/服务)架构。每一层作为独立的模块进行开发和维护。这三层分别是用户界面、功能处理逻辑(亦称业务规则)、数据存储访问。

模块软件的优势在于良好的接口定义。三层架构使得当需求和技术改变时，各个模块可以分别升级或重构。例如，操作系统的表现层的变化只会影响用户界面代码。

通常用户界面运行于台式机或工作站，并使用标准的图形用户界面；功能处理逻辑可以是一个或多个独立的模块，运行于工作站或服务器上；数据存储逻辑则运行于大型机或是一个运行于数据库服务器上的RDBMS。中间层可以进一步细分为多层(在这种情况下，整体架构称为“n层结构”)。

一般的三层架构内容如下：

- 表示层(Presentation Tier)。表示层位于应用的顶层，表示有关服务信息，如商品、采购物和购物车内容等。通过与其他层通信，输出结果给浏览器/客户层和网络中的所有其他层。
- 应用层(Application Tier)(业务逻辑、逻辑层、数据访问层或中间层)。逻辑层从表示层独立出来作为一层，通过执行具体的处理控制应用的功能。
- 数据层(Data Tier)。该层是数据库服务，提供信息存储和检索功能。同时保持数据独立，不依赖应用服务或业务逻辑。数据层独立增加了可扩展性，改进了运行性能。

1.2.5 软件架构相关概念

1. 设计分类

采用合理的软件架构是开发优秀软件的基础之一，因此也带来了不同的软件设计相关概念。根据IEEE的标准，一些术语定义如下：

架构设计：用硬件、软件组件及接口的集合确立开发一个计算机系统框架的过程。

详细设计：扩展系统或组件的初步设计，使得某种程度上该设计是可实现的。

功能设计：定义系统组件间相互协同工作的过程。

初步设计：分析设计方案，确定系统或组件的架构、组件、接口，以及估算执行时间和规模的过程。

软件架构设计是战略设计，考虑全局需求和管理解决方案做什么，例如编程规范、架构风格、组件式的软件工程标准、架构模式、安全性、规模、集成性、法律法规。功能设计是战术设计，考虑局部需求和管理解决方案如何做，例如算法、设计模式、程序设计风格、重构方法及底层实现。

2．分布式计算

由于互联网技术兴起，B/S 模式应用软件成为主流。下面简要介绍三种主要的分布式计算。

P2P(Peer-to-Peer)，又称为对等互联网技术，是一种网络新技术，依赖网络中参与者的计算能力和通信带宽，而不必依赖少数的几台服务器。该技术给网络上的数据交换和通信带来了便利和效率。

C/S 模式的软件由客户端(Client)和服务器端(Server)组成，运行于网络上。用户通过客户端软件使用应用软件。客户端将用户请求传递给服务器，服务器将处理结果回送给客户端并反馈给用户。

B/S 模式的软件由浏览器(Browser)和服务器(Server)组成，运行于互联网上。用户通过浏览器这种统一的接口使用应用软件，简化了学习过程。浏览器将用户请求传递给服务器，服务器将处理结果回送给浏览器并反馈给用户。B/S 模式满足了全球网络开放、互连、信息共享的需求。同时，B/S 模式只需管理服务器，这给系统运行维护带来了方便。多数 B/S 模式软件由浏览器、应用服务、数据库服务组成。

3．MVC

模型－视图－控制器(Model View Controller，MVC)是 Xerox PARC 在 1980 年为编程语言 Smalltalk－80 发明的一种软件设计模式，至今已经被广泛使用。MVC 原先用于桌面应用程序中，如今广泛用于 Web 应用软件的开发之中。模型层实现软件系统的业务逻辑，视图层实现与用户交互界面，控制层实现模型层与视图层之间信息传输，分配用户请求并选择恰当的视图显示，同时解释用户的输入并映射为模型层可执行的操作。

1.3 软件开发过程

软件生命周期(Systems Development Life Cycle，SDLC)是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。

1.3.1 软件开发活动

软件开发活动大体包括以下几个过程，这是一种较为粗糙的划分。

1．问题的定义及规划

此阶段是软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。

2．需求分析

在确定软件开发可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下良好的基础。需求常常是在整个软件开发过程中不断变化和深入的，因此必须制定需求变更计划来应付这种变化，以保护整个项目的