高速公路机电工程软件开发技术

◎ 邹国平　黄　铮　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高速公路机电工程软件开发技术

邹国平　黄铮　编著

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

内 容 简 介

本书主要介绍高速公路机电工程软件开发技术，分上、下两篇。上篇为基础理论篇，系统地阐述了软件开发的方法与技术；下篇为软件开发的工程实践，详细介绍了高速公路机电工程中收费系统、监控系统和隧道监控系统的软件开发技术及实例。

书中从软件工程的相关概念、软件工程技术及其发展趋势入手，介绍了面向对象的软件开发方法与技术、分布式计算技术、新型软件开发方法与技术，进而结合高速公路机电工程中的应用实例，介绍了高速公路收费系统软件开发、监控系统软件开发、隧道监控系统软件开发，并且给出了相应章节关键技术的实现代码和实例程序的框架代码。

本书选材新颖，内容丰富，理论与实践相结合，实用性强，可作为交通信息工程、软件工程等相关专业高校师生的教学及学习用书，也可作为相关工程技术人员、软件开发人员的工作参考书。

图书在版编目（CIP）数据

高速公路机电工程软件开发技术 / 邹国平，黄铮编著.—北京：电子工业出版社，2008.10
ISBN 978-7-121-07434-9

Ⅰ. 高… Ⅱ. ①邹…②黄 Ⅲ. 高速公路—机电工程—软件开发 Ⅳ. U412.36-39

中国版本图书馆 CIP 数据核字（2008）第 148937 号

责任编辑：朱清江
特约编辑：郭茂威
印　　刷：
装　　订：北京市李史山胶印厂
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：16.5　字数：400 千字
印　　次：2008 年 10 月第 1 次印刷
定价：40.00 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

高速公路机电工程是高速公路的重要组成部分，是高速公路的“神经中枢”，在高速公路运营管理中发挥着重要的作用。机电工程软件又是高速公路机电工程的灵魂，它的开发成功与否影响到交通机电工程系统功能的发挥，也就决定了交通机电工程系统的成败。

交通机电工程软件开发涉及高速公路机电工程和软件开发方法与技术两个完全不同的专业领域。目前市面上有涉及高速公路机电工程的书籍，但这些书籍只是介绍高速公路机电工程原理或系统集成技术等方面的知识，而关于高速公路机电工程软件开发的专业书籍却没有。考虑到市面上缺乏高速公路机电工程专业软件开发方面的资料，笔者编写了这本书。编写本书的出发点是将笔者长期从事高速公路机电工程软件开发和应用的实践经验加以总结归纳，并结合软件工程技术的最新进展，尽量用浅显易懂的语言将复杂的工程软件开发技术讲得透彻，期望能给相关专业的工程技术人员、交通信息工程专业师生及广大软件开发人员以参考。

本书共七章，分上、下两篇。上篇系统地阐述了软件开发方法与技术，下篇介绍了高速公路交通机电工程中收费系统、监控系统和隧道监控系统的软件开发。上篇是本书的基础理论部分，下篇则是软件开发的工程实践部分。下面简略地介绍各章的内容。

第 1 章　软件工程技术概述

首先介绍了软件工程技术中的一些基本术语，然后介绍了软件相关的一些基本概念，阐述了软件工程技术的总体框架、软件工程技术的发展历程、CASE 工具及环境和软件过程技术。最后，以 Internet 的快速发展和普及为背景，探索了软件技术的发展趋势。通过本章的学习，读者对软件工程技术有了一个基本的了解。

第 2 章　面向对象的软件开发

从面向对象软件工程方法、统一建模语言 UML 和统一建模过程 RUP 三个方面详细介绍了面向对象的软件开发方法。面向对象的软件开发方法与技术至今还是工程软件开发的主流方法与技术，下篇中的实例代码很好地体现了面向对象的编程思想。

第 3 章　分布式计算技术

首先介绍了分布式计算的概念，剖析了其内涵，分析了分布式软件体系结构和分布式计算技术中极其重要的中间件技术。

分布式软件开发方法和面向对象技术的结合极大地推动了软件工程技术的进步，产生了一些主流的软件开发方法和技术，如 OMG 的公共对象请求代理体系结构（CORBA）、Microsoft 的分布式组件对象模型（DCOM）、Sun Microsystems 的 Enterprise Java Beans（EJB）等。本章对上述三种主流的软件开发方法和技术进行了详细的介绍。

第 4 章　新型软件开发方法与技术

介绍了一些新型软件开发方法与技术。它们是：敏捷软件开发方法、面向 Aspect 的软件开发、面向 Agent 的软件开发、软件重用技术和基于构件的软件开发技术。

第 5 章　收费系统软件开发

首先介绍了收费系统的基本概念，然后对路网环境下收费的必要条件和关键技术、路网环境下收费系统总体构成及其数据流模型进行了分析，最后从软件体系结构、网络互联和关系型数据库等方面对路网环境下的收费系统应用软件实现技术进行了剖析。

其次对高速公路收费系统的功能需求进行了分析。需求分析是系统软件设计的基础，接下来从硬件环境、软件运行平台、软件实现主要结算、软件模块组成及其功能描述和设计要

求、程序接口规范和数据库等方面对收费系统软件进行了总体设计。

收费系统软件的关键技术主要有车道收费软件中的关键技术、收费系统的数据传输技术和收费系统后台管理系统中的报表技术，本章对这些关键技术进行了详细介绍并给出了Delphi环境下的实现代码。

本章最后给出了车道收费系统的完整实例。由于篇幅所限，书中只给出了系统程序的框架代码，代码展示了很多Delphi的高级编程知识和技巧。通过实例，读者能深入理解和掌握车道收费系统的软件开发技术。

第6章 监控系统软件开发

先从概念、功能、组成等方面系统地介绍了高速公路监控系统，让读者对高速公路监控系统有个整体性的认识，以便更好地理解本章下面的内容。

接下来对高速公路监控系统的需求进行了分析，在需求分析的基础上，对监控系统软件进行了总体设计。此外，对系统所需的数据库部分的设计也作了简单的介绍。

监控系统软件最关键的部分是图控子系统和通信子系统，本章对其软件实现技术进行了详细分析，并给出了Delphi环境下的核心代码。本章最后给出了监控系统软件开发的一个完整实例，并详细注释了图控程序和通信程序的框架代码。

第7章 隧道监控系统软件开发

首先介绍了隧道监控系统的监控模式及其组成，让读者对隧道监控系统有一个大概的了解；然后分析了可编程控制器（PLC）、现场总线、工业以太网的技术，剖析了工业控制领域重要的软件平台和开发环境：组态软件以及OPC规范。这些都是隧道软件开发的基础。

隧道监控软件的开发分为PLC的开发和上位机的编程两部分。本章的最后通过一个实例，从需求分析、PLC选型、PLC组网、PLC和上位机编程等方面展示了隧道监控软件开发的全过程。

本书得到了江西省交通厅、江西赣粤高速公路股份有限公司和江西方兴科技有限公司的支持和资助，是作者在从事科研项目过程中，研读了大量国内外相关技术资料和文献，并结合工程实践经验写成的。本书选材新颖，内容丰富，实用性强，概念清晰，通俗易懂，可作为相关专业的工程技术人员、交通信息工程专业的师生及具有一定编程基础的软件开发人员的参考用书。

本书代码来源于实际工程项目，考虑到单位的技术保密，作者隐含了某些程序的实现细节，如设备类中的DLL代码、数据库中表单的具体定义等，希望读者谅解。书中除特殊说明外，所有代码在Delphi7.0环境下调试通过。

本书第1、2、3、4章由邹国平撰写，第5章由叶见勇、邹国平撰写，第6章由李卫星、黄铮撰写，第7章由郝国昌、胡孝望撰写，全书由邹国平、黄铮统稿、定稿。

由于作者水平有限和时间仓促，书中难免有不足和疏忽之处，恳请各位同仁和读者批评指正。欢迎将对本书的任何指教或宝贵意见通过E-mail发送给作者，也欢迎就有关问题和作者联系。

作者E-mail地址：zgping@jxjt.gov.cn。

编 者

2008年9月于南昌

目　录

第1章 Chapter 1

软件工程技术概述

1.1 软件与软件工程

1. 基本术语

在学习软件开发的基本原理之前，我们有必要先了解软件工程技术中的几个基本术语，以便更好地理解本章下面的内容。

计算（Computing）：解决问题的手段。

算法（Arithmetic）：解决问题的步骤。

指令（Instruction）：表示算法的符号。

程序（Program）：程序=算法+数据结构。

软件（Software）：软件=程序+文档。

软件开发（Software Development）：构造软件的过程。

软件工程（Software Engineer）：用工程方法构造软件。

软件过程（Software Process）：设计、开发、应用和维护软件产品的一组相互关联的活动、方针、组织结构、技术方法、规程以及工作产品。它定义了对软件开发进行组织、管理、度量、支持和改进的途径。

软件过程模型：使用适当的方法表达的一个软件过程的抽象描述。

软件工具：为支持软件开发、维护、管理而研制的计算机程序系统。

计算机辅助软件工程（Computer-Aided Software Engineering，CASE）：一个软件系统，用于对软件过程的活动提供自动化支持。

2. 软件概述

计算机软件（Software）是指与计算机系统操作有关的程序、规程、规则及任何与之有关的数据和文档资料。它由两部分组成：一是使计算机硬件能完成计算和控制功能的有关计算机指令和计算机设计定义的组合，或机器可执行的程序及有关数据；二是机器不可执行的，与软件开发、运行、维护、使用和培训有关的文档。因此，IEEE 将软件定义为“计算机程序和相关的文档”。

程序（Program）是用程序设计语言描述的、适合于计算机处理的语句序列。程序设计语言编译器可以将程序翻译成一组机器可执行的指令，这组指令也称机器语言程序，它将根据用户的需求，控制计算机硬件的运行、处理用户提供的或机器运行过程中产生的各类数据并

输出结果。

文档（Document）是一种数据媒体和其中所记录的技术数据和信息，包括计算机的列表和打印输出。文档用来记录计算机软件的要求、设计或细节，解释软件的能力和限制条件，或提供软件运行期中使用或保障计算机软件的操作命令。文档记录软件开发的活动和阶段成果，不仅可以用于专业人员和用户之间的通信和交流，而且还可以用于软件开发过程的管理和运行阶段的维护。

3. 软件开发的演变过程

软件开发经历了程序设计阶段、软件设计阶段和软件工程阶段的演变过程。

（1）程序设计阶段

程序设计阶段出现在 1946～1955 年。此阶段的特点是：尚无软件的概念，程序设计主要围绕硬件进行开发，规模很小，工具简单，无明确分工（开发者和用户），程序设计追求节省空间和编程技巧，无文档资料（除程序清单外），主要用于科学计算。

（2）软件设计阶段

软件设计阶段出现在 1956～1970 年。此阶段的特点是：硬件环境相对稳定，出现了“软件作坊”的开发组织形式。开始广泛使用产品软件（可购买），从而建立了软件的概念。随着计算机技术的发展和计算机应用的日益普及，软件系统的规模越来越庞大，高级编程语言层出不穷，应用领域不断拓宽，开发者和用户有了明确的分工，社会对软件的需求量剧增。但软件开发技术没有重大突破，软件产品的质量不高，生产效率低下，从而导致了软件危机的产生。

（3）软件工程阶段

自 1970 年起，软件开发进入了软件工程阶段。由于软件危机的产生，迫使人们不得不研究、改变软件开发的技术手段和管理方法。从此软件产生进入了软件工程时代。此阶段的特点是：硬件已向巨型化、微型化、网络化和智能化四个方向发展，数据库技术已成熟并广泛应用，第三代、第四代语言出现。

4. 软件危机

20 世纪 60 年代，随着第三代计算机的产生，计算机的硬件性能发生了翻天覆地的变化，运行大型的复杂软件系统已经成为可能，然而，相应的软件开发技术却难以满足大型软件系统的开发需求。这种软件技术跟不上硬件发展而造成的诸多问题被称为软件危机（Software Crisis）。

（1）软件危机的表现

① 软件成本日益增长

在计算机发展的早期，大型计算机系统主要是被设计应用于非常狭窄的军事领域。在这个时期，研制计算机的费用主要由国家财政提供，研制者很少考虑到研制代价问题。随着计算机市场化和民用化的发展，代价和成本就成为投资者考虑的最重要的问题之一。20 世纪 50 年代，软件成本在整个计算机系统成本中所占的比例为 10%～20%。但随着软件产业的发展，软件成本日益增长。相反，计算机硬件随着技术的进步、生产规模的扩大，价格却不断下降。这样一来，软件成本在计算机系统中所占的比例越来越大。到 20 世纪 60 年代中期，软件成本在计算机系统中所占的比例已经增长到 50%左右。

② 开发进度难以控制

由于软件是逻辑、智力产品，软件的开发需建立庞大的逻辑体系，这是与其他产品的生产不一样的。例如，工厂里要生产某种机器，在时间紧的情况下可以要工人加班或者实行“三

班倒”，而这些方法都不能用在软件开发上。

在软件开发过程中，用户需求变化等各种意想不到的情况层出不穷，令软件开发过程很难保证按预定的计划实现，给项目计划和论证工作带来了很大的困难。

Brook 曾经提出：“在已拖延的软件项目上，增加人力只会使其更难按期完成”。事实上，软件系统的结构很复杂，各部分附加联系极大，盲目增加软件开发人员并不能成比例地提高软件开发能力。相反，随着人员数量的增加，人员的组织、协调、通信、培训和管理等方面的问题将更为严重。

③ 软件质量差

软件项目即使能按预定日期完成，结果却不尽人意。1965 年至 1970 年，美国范登堡基地发射火箭多次失败，绝大部分故障是由应用程序错误造成的。程序的一些微小错误可以造成灾难性的后果。

在“软件作坊”里，由于缺乏工程化思想的指导，程序员几乎总是习惯性地以自己的想法去代替用户对软件的需求，软件设计带有随意性，很多功能只是程序员的“一厢情愿”而已，这是造成软件不能令人满意的重要因素。

④ 软件维护困难

正式投入使用的软件，总是存在着一定数量的错误，在不同的运行条件下，软件就会出现故障，因此需要维护。但是，由于在软件设计和开发过程中，没有严格遵循软件开发标准，各种随意性很大，没有完整的真实反映系统状况的记录文档，给软件维护造成了巨大的困难。特别是在软件使用过程中，原来的开发人员可能因各种原因已经离开原来的开发组织，使得软件几乎不可维护。

另外，软件修改是一项很“危险”的工作，对一个复杂的逻辑过程，哪怕做一项微小的改动，都可能引入新的问题。

（2）软件危机产生的原因

从软件危机的种种表现和软件作为逻辑、智力产品的特殊性，可以发现软件危机产生的原因：

- 用户需求不明确；
- 缺乏正确的理论指导；
- 软件的规模越来越大；
- 软件复杂度越来越高；
- 软件产品开发的效益跟不上计算机硬件的复杂以及用户需求的增长。

（3）解决软件危机的途径

人们在认真地研究和分析了软件危机背后的真正原因之后，得出了“人们面临的不光是技术问题，更重要的是管理问题，管理不善必然导致失败”的结论，便开始探索用工程的方法进行软件生产的可能性，即用现代工程的概念、原理、技术和方法进行计算机软件的开发、管理和维护。于是，计算机科学技术的一个新领域——软件工程诞生了。

软件工程的三要素是方法、工具和过程。

- 软件工程方法为软件开发提供了“如何做”的技术，是完成软件工程项目的技术手段；
- 软件工具是人类在软件开发活动中智力和体力的扩展和延伸，为软件工程方法提供自动或半自动的软件支撑环境；
- 软件工程的过程则是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。

迄今为止，软件工程的研究与应用已经取得很大成就，它在软件开发方法、工具、管理等方面的应用大大缓解了软件危机造成的被动局面。

5. 软件工程

为了解决软件危机，1968 年在德国召开的国际学术会议上，北大西洋公约组织（NATO）的计算机科学家第一次提出“软件工程”的概念，希望通过系统化、规划化、数量化等工程原则和方法来实现复杂软件系统的开发和维护。

按照 Webopedia 词典中的定义，软件工程是“研究如何开发大型应用系统的计算机科学学科。软件工程不仅覆盖构建软件系统的相关技术层面问题，还包括诸如指导开发团队、安排进度以及预算等管理层面问题”。由这个定义可以看出，软件工程不仅仅包括编写程序代码所涉及的技术，它还包括所有对软件开发能够造成影响的问题。Brook 在 1987 年指出，不存在任何一个单一的开发技术或管理技术能够解决软件工程所面临的所有问题。因而软件工程是一个包括一系列概念、理论、模式语言、方法以及工具的综合学科。图 1.1 给出了一个软件工程技术的总体框架。可以看出，软件工程技术可以分为产品实现层技术以及开发管理层技术。其中，产品实现层技术涉及与特定软件系统开发相关的问题，为在软件生命周期的各个阶段实现软件产品提供技术支持；开发管理层技术通常不针对特定的某个软件开发项目，而是为管理和改进软件组织所有的业务活动提供技术支持，例如如何使用适当的方法管理软件开发过程中所需要执行的各个活动，以便在特定的软件项目中系统地展开软件工程各层技术，支持软件组织的业务实现，从而控制软件产品开发的成本，提高生产效率，保证和改进软件产品的质量。

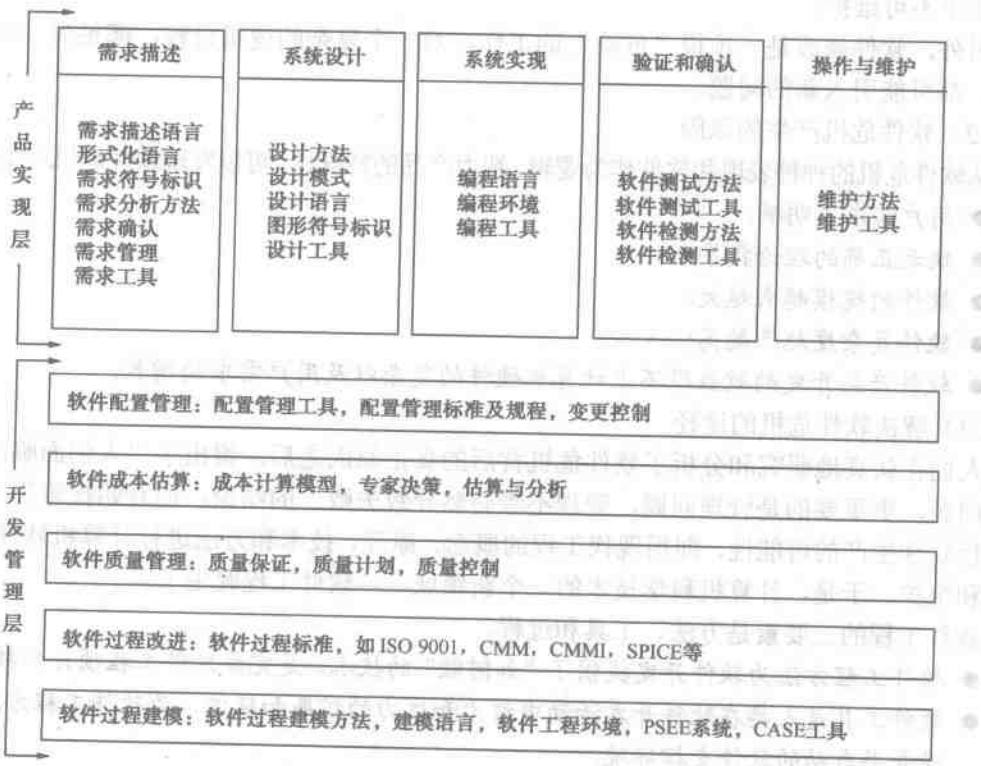

图 1.1 软件工程技术的总体框架

1.2　软件工程技术发展历程

1. 结构化程序设计

结构化程序设计（Structured Programming，SProg）的概念最早由荷兰著名学者 E.W.Dijkstra 提出。1965 年他在一次国际会议上指出，“可以从高级语言中取消 goto 语句”，同时他认为“程序的质量与程序中所包含的 goto 语句的数量成反比”。1966 年 Bohm 和 Jacopini 证明了结构定理。结构定理指出：任何程序逻辑都可以以顺序、选择和循环三种基本控制结构实现，并且是具有单入口、单出口的。

结构定理奠定了结构化程序世界技术的理论基础，到了 20 世纪 60 年代末期人们认识到结构化程序设计不是简单地去掉 goto 语句的问题，它应该是一种新的程序设计思想、方法和风格，可以显著地提高软件的生产率和降低软件维护的代价。但随着系统复杂度的提高，单独使用结构化方法并不能保证软件的质量。尽管使用了结构化方法，开发出来的软件依然难以理解和使用，于是导致了功能分解技术的出现。

2. 功能分解

功能分解（Functional Decomposition，FD）技术是一个过程方法，它将要实现的最终系统分解成一系列逐步细化的概念化（Conceptualization）的模块。概念之间的关系用结构图（Structure Chart）来表示。FD 通常在面向过程的范例（Paradigm）中使用。这些系统的概念模块是以面向过程的方式定义的（每一个模块代表一个过程或子过程）。FD 的目标提供一种方法通过抽象来逐步求精地理解系统，其开发的产品具有良好的结构。系统的概念模型和表示与源代码的结构是一致。这种方法至今依然在使用，但结构图已经不能提供足够的信息来保证可以得到一个结构良好、准确的解决方案了。为了增加一些必要的信息，出现了结构化分析与设计方法。

3. 结构化分析与设计

结构化分析与设计（Structured Analysisand Design，SAD）的降临标志着第一个软件工程方法的诞生。结构化分析（Structured Analysis，SA）方法是面向数据流自顶向下逐步求精进行需求分析的方法；结构化设计（Structured Design，SD）方法是根据需求阶段对数据流的分析（一般用数据流图和数据字典表示）设计软件结构的方法；SAD 用一组技术共同来表示软件开发的过程。SAD 基于 SPrag 和 FD，并进一步用抽象的技术来产生模块化的输出。随着 SAD 的引入，最终实现系统的交付变成一系列的里程碑而不是一个里程碑。分析要解决的问题以及解决办法的设计都被认为是软件开发过程的重要步骤。

4. 以数据为中心的设计方法

以数据为中心的设计方法（Data-Centered Design Method，DCM）的贡献是在结构化分析中扩充了数据模型，其目的是确定整个组织的数据需求，创建一个中心的、集成的数据库，单独的应用程序开发并从中心数据库取数据。数据模型用 ER 模型表示。ER 最初的目的是为关系数据库设立的，建立了设计模型之后，应用程序的开发就可以用结构化的分析和设计来关注中心数据库的数据。

5. 面向对象的设计方法

面向对象的方法（Object-Oriented Method，OO）是软件过程方法的又一次飞跃。对象是一个具有一组状态的实体，并封装了附加于这些状态的操作。状态描述了对象的属性或特征，

操作描述了对象改变其状态的方法以及该对象为其他对象所提供的服务。面向对象方法认为，人类生活在一个由对象（Object）组成的世界中。对象可以被归类、描述、组织、组合、创建和操纵。面向对象方法是一种模型化设计的抽象方法，结构上具有良好的高内聚低耦合特性。采用面向对象技术设计和开发的软件系统更易于维护，在对系统进行修改时，能够产生较少的副作用。同时，面向对象技术提出了类、继承、封装、接口等概念，从而为对象的复用提供了良好的支持机制，因而采用面向对象技术对软件产品进行设计和开发，也能够有效地提高软件组织的开发效益。

面向对象方法包括面向对象编程（OOP）、面向对象设计（OOD）和面向对象分析（OOA）。它是一种自底向上的归纳和自顶向下的分解相结合的方法，通过对象模型的建立，能够真正建立基于用户的需求，而且系统的可维护性大大改善。因此 OO 方法与技术在需求分析、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，基本解决了软件开发在这方面存在的严重问题，成为软件组织分析、设计和开发软件产品的首选范型。当前业界关于面向对象建模的标准是 UML（Unified Modeling Language）。

6. 以过程为中心的软件开发方法

严格地说，前面的各种软件工程方法都属于软件产品的实现层技术，提供了一系列方法、技术和工具支持软件被设计为具有良好定义的结构，使得其复杂度可以得到控制、软件易于实现、易于维护和移植。这些方法和技术在一定程度上可以改进软件质量、可靠性、结构以及成功交付软件的概率。但事实上，单纯依靠实现层技术并不能保证软件组织高质量并高效率地开发软件产品，即使到了今天，软件危机依然存在，甚至由于软件应用领域的普及、需求的频繁变化，危机有愈演愈烈的趋势。

随着信息技术的飞速发展，软件的应用范围、复杂度和规模都在急剧增大，传统软件工程基于实体驱动和确定目标、有序控制的开发模式开始让位于 Internet 环境下以过程为中心、基于协同驱动和动态目标、实体聚合的开发模式。在 20 世纪 80 年代中期，以提出过程成熟模型 CMM、个人软件过程 PSP 和小组软件过程 TSP 为标志，软件工程逐步进入以过程为中心的生产阶段。而从 20 世纪 90 年代开始，软件工程进入了以软件过程、面向对象和构件重用三种软件开发方法与技术并驾齐驱的软件工业化生产阶段。

1.3 CASE 工具及环境

1.3.1 计算机辅助软件工程

软件工具是指在软件开发、维护和分析中使用的程序系统。具体来讲就是开发人员在系统分析、设计、编程、测试过程中应用的一套辅助工具。例如，在设计、分析阶段有 PSL/PSA、AIDES、SDL/PAD 等；在编程阶段有 BASIC 编译器、PASCAL 编译器等。软件工具通常由工具、工具接口和工具用户接口 3 部分组成。工具通过工具结构与其他工具、操作系统或网络操作系统及通信接口、环境信息库接口等进行交互作用，当工具需要与用户进行交互作用时，则通过工具的用户接口来进行。软件工具能在软件开发各个阶段帮助开发者控制开发中的复杂性，提高工作质量和效率。随着软件工程思想的日益深入，计算机辅助软件开发工具和开发环境得到越来越广泛的应用。

在软件工程活动中，软件工程师和管理员按照软件工程的方法和原则，借助于计算机及

其软件工具的帮助，开发、维护、管理软件产品的过程，称为计算机辅助软件工程（Computer-Aided Software Engineering,CASE）。CASE 系统常常用作方法支持，其有效性只有通过“集成”才能达到。一般的 CASE 环境需要网络的支持，允许若干软件工程师在这个环境中同时使用相同或不同的软件工具相互通信协同工作。CASE 技术是软件技术发展的产物，它既起源于软件工具的发展，又起源于软件开发方法学的发展，同时还受到实际应用发展的驱动。CASE 环境的核心是软件工程信息库。CASE 工具及其环境的开发和使用是软件工业化的产物，已引起世界各国的普遍重视，成为软件工程的重要研究领域。

1.3.2 CASE 工具

支持软件工程活动的软件工具品种多、数量大。人们可以根据软件工具的功能、作用、使用方式等多种原则进行分类。一般来说，按照 CASE 工具的功能，可将其划分为以下 9 类。而所有这些工具都是在软件工程信息库的支持下工作的。

（1）事务系统规划工具（Business System Planning Tools）

事务系统规划工具为定制事务信息系统规划提供“元模型”。利用“元模型”可以生成专用事务信息系统模型，该模型反映了一个单位各部门之间的信息流程。建立专用事务信息系统模型需要提供系统资源、模型运行方式和管理方法。

（2）项目管理工具（Project Management Tools）

项目管理类工具主要用于管理人员有效地估算软件项目所需的工作量、成本和研制周期等，定义功能分解结构，并制定可行的项目开发计划。多数 CASE 的项目管理工具只支持项目管理的某一活动。项目管理员从 CASE 工具箱中挑选适当的工具，估算项目的工作量、成本、制定进度计划等。项目管理工具只有包括：项目计划工具、需求追踪工具、度量和管理工具等。目前常用的项目管理工具有 Rational Clear Case，CVS 和 Microsoft Project 2000 等。

（3）支撑工具（Support Tools）

支撑工具支持软件开发和维护的全过程。只要包括文档工具、操作系统、网络系统软件、质量保证工具、软件配置管理工具、数据库管理工具等。

（4）分析与设计工具（Analysis and Design Tools）

分析与设计工具主要用于建造系统模型。该模型包括数据的表示、数据内容、控制流、控制规格说明、进程表示等。分析设计软件工具不仅能帮助建造模型，而且能帮助评价模型的质量，检查模型的一致性和正确性。主要工具包括结构化分析/结构化设计（SA/SD）工具、面向对象分析/面向对象设计（OOA/OOD）、原型/模拟（PRO/SIM）工具、界面设计和开发工具等。

（5）程序设计工具（Programming Tools）

程序设计工具用于软件开发过程中的编码活动。主要包括传统的程序设计工具（如各种编辑器、编译器、调试器）、第四代程序设计工具（如数据库查询系统代码产生器、四代语言 4GL）、面向对象的程序设计工具（C++、Smalltalk、Eiffel）。

（6）测试与分析工具

测试与分析工具支持软件开发过程中的测试活动。包括：测试数据获取工具、程序静态（非执行状态）测量工具、程序动态（执行状态）测量工具、硬件或其他外部设备的模拟工具、测试管理（测试流程管理、缺陷跟踪管理、测试用例管理）工具等。

（7）原型建造工具（Prototyping Tools）

原型建造工具通常支持某一领域的原型建造，带有一定的专用性（如通信、航空、航天）。

较低级的原型可以用手工或机器描述系统的结构、功能和人机界面等，这样的原型是静态的、不能执行的。较好的原型工具不仅能描述系统的特征和功能，还可以生成可执行代码，演示系统的动态行为和功能。近年来，某些原型工具开始借助知识库来理解应用领域的知识，建造可执行的原型系统。

（8）维护工具（Maintenance Tools）

维护工具支持软件维护。按功能划分，包括从程序到规格说明的逆向工程工具、代码的重构和分析工具、在线系统的重新工程化工具（如修改在线数据库系统）。

（9）框架工具（Frame Tools）

框架工具是数据库管理、配置管理和 CASE 工具集成的软件工具。

1.3.3 集成化的 CASE 环境

工具的集成与提高工具的互操作性代表了当前 CASE 发展的主要趋势。20 世纪 90 年代，CASE 工具逐渐发展成为集成化的 CASE 环境。

1. CASE 集成环境的定义

“集成”的概念首先用于术语 IPSE（集成工程支持环境），而后用于术语 ICASE（集成计算机辅助软件工程）和 ISEE（集成软件工程环境）。工具集成是指工具协作的程度。集成在一个环境下的工具的合作协议，包括数据格式、一致的用户界面、功能部件组合控制和过程模型。

（1）界面集成

界面集成的目的是通过减轻用户的认知负担而提高用户使用环境的效率和效果。为达到这个目的，要求不同工具的屏幕表现与交互行为要相同或相似。表现与行为集成，反映了工具间的用户界面在词法水平上的相似（鼠标应用、菜单格式等）和语法水平上的相似（命令与参数的顺序、对话选择方式等）。更为广义的表现与行为定义，还包含两个工具在集成情况下交互作用时，应该有相似的反应时间。界面集成性的好坏还反映在不同工具在交互作用范式上是否相同或相似。也就是说，集成在一个环境下的工具，能否使用同样的比喻和思维模式。

（2）数据集成

数据集成的目的是确认环境中的所有信息（特别是持久性信息）都必须作为一个整体数据能被各部分工具进行操作或转换。衡量数据的集成性，我们往往从通用性、非冗余性、一致性、同步性、交换性五个方面去考虑。

（3）控制集成

控制集成是为了能让工具共享功能。我们给出了两个属性来定义两个工具之间的控制关系。供给：一个工具的服务在多大程度上能被环境中另外的工具所使用；使用：一个工具对环境中其他工具提供的服务能利用到什么程度。

（4）过程集成

过程为开发软件所需要的阶段、任务和活动序列，许多工具都是服务于一定的过程和方法的。我们说的过程集成性，是指工具适应不同过程和方法的潜在能力有多大。很明显，那些极少做过程假设的工具（如大部分的文件编辑器和编译器）比起那些做过许多假设的工具（如按规定支持某一特定设计方法或过程的工具）要易于集成。在两个工具的过程关系上，具有三个过程集成属性：过程段、事件和约束。