第 1 章　软件工程综述

软件工程是工程化软件开发与维护的方法论。软件的开发者、维护者或软件项目管理者都将是软件工程的实践者，并都需要掌握与应用软件工程方法。

- 软件工程及其发展
- 软件特点与分类
- 软件危机现象与原因
- 工程技术与工程管理
- 主流工程方法学

1.1　什么是软件工程

软件工程是关于软件开发、使用与维护的工程方法学，是一门涉及工程技术、工程管理与工程经济等诸多内容的综合性工程学科。软件工程建立在与软件有关的工程概念、原理与方法基础上，它是对现实软件问题的工程方法探索，具有鲜明的工程实用性。

软件工程是软件产业发展的必由之路。通常看来，软件产业的发展涉及软件技术的进步、软件应用域的拓宽、软件生产方式的改进，其犹如三匹骏马，拉动着软件在产业化大道上奔驰。大体上看，软件的产业化发展经历了程序设计、程序系统和软件工程三个时代。

程序设计时代（20 世纪 50 年代）是软件发展的早期阶段，这时的软件仅体现为程序，一般使用密切依赖于某特定计算机硬件的机器语言或汇编语言，直接面对机器编制。因此，这时的程序很难由一台设备移植到另一台设备。这时的程序则主要用于科研机构的科学工程计算，设计中通常没有设计文档的支持，而且程序的任务单一、规模小、结构简单，其主要设计问题是程序算法改进。这时的程序还没有成为产品，程序编制者大多就是程序使用者，基本上是个人设计、个人使用、个人操作、自给自足的个性化的软件生产方式。实际上，由于缺乏统一的程序标准，设计者完全可按照自己对问题的理解去构造程序，目标则是实现功能与追求高性能，硬件是唯一需要重视的条件限制，此外则似乎再无其他原则性约束。

程序系统时代（20 世纪 60 年代）是软件产业发展的关键阶段，这时的计算机无论硬件还是软件都出现了显著的进步。体现计算机硬件指标的半导体材料、集成电路迅速发展，并获得了很有成效的应用，其不断改善计算机的计算速度、可靠性和存储容量。计算机软件也在迅速发展，高级程序语言获得了有效应用，其提高了编程效率，并方便了大规模复杂程序系统的创建。操作系统也在这个时期出现了，其有效地改善了软件的工作环境，并使软件具有了很好的可移植性。程序系统时代的技术进步有力地推动了计算机应用的发展。一些大型商业机构开始使用计算机进行商业数据处理。应该说，这个时期的软件需求在飞速增长，软

件规模也在不断扩大。“软件作坊”在这个时期应运而生，它们专业从事软件生产，以满足迅速膨胀的软件需求。

程序系统时代出现了软件危机。软件作坊带来了不同于个人开发的工业化软件产品，但仅仅只是工业化软件生产的起步，成员合作需要依赖的行为准则、技术标准并没有建立起来，以致产品随意性大、质量难以保证。

应该说，软件产业发展过程中爆发的软件危机促进了软件工程的快速成长。1968 年在前联邦德国召开的计算机国际会议上，来自原北大西洋公约组织的计算机科学家针对软件危机问题进行了专题讨论。软件工程这个学术名词也在这次学术会议上被正式提出，其用来代表基于工程模式的软件开发，以谋求对软件危机的有效克服。

软件工程使身处软件危机黑暗中的软件开发者看见了曙光，并使软件产业发展步入到了软件工程时代（20 世纪 70 年代起）。无须质疑的是，软件的产业化发展也确实需要有软件工程方法的支持。实际上，自从软件成为了工业化产品，它所面临的就已不只是软件技术问题了，还必须考虑如管理、经济、应用等其他因素。软件工程即是基于工程角度对有关软件的诸多因素的综合考虑。

软件工程的最核心问题是，如何更有效率并更高质量地开发软件，其作用范围则贯穿于自软件的最初定义，到软件的开发实现，直到软件的使用与维护的整个软件生命过程。因此，无论是软件开发者，或是软件维护修复者，还是软件使用者，都有可能成为软件工程实践者。有许多人在专门从事软件工程研究，他们是专业的软件工程研究者，他们一直在探讨如何给软件工程下一个恰如其分的定义，以使该学科研究有一个较为明确的发展目标。

20 世纪 60 年代末，软件工程早期研究者 FritZ Baner 给软件工程的定义是：“为了经济地获得能在实际机器上可靠运行的软件，而需要合理建立，并有效应用的工程原则”。显然，这是一个在今天看来很不全面的定义，其工程特征仅体现为工程原则。但该定义把软件工程学科研究中所将面临的实质问题提出来了，即软件的成本、软件的质量、软件生产应该采用的工程方法。因此，该定义成为了以后软件工程方法学成长发展的基线。

1993 年，国际权威机构 IEEE 给出了关于软件工程的更加全面的定义，即“软件工程是：①将系统的、受规范约束的、可量化的方法应用于软件的开发、运行与维护，即将工程方法应用于软件；②对①中工程方法的研究”。应该说，这个定义对软件工程的工程学科特性有了更完整清晰的表述，如对软件工程的作用范围、软件工程基于工程应用的研究途径等都有了比较具体的说明。

1.2 软件有什么特点

计算机系统由硬件、软件两部分组成。硬件是物理部件，如处理器、存储器、主板总线，具有一定的物理形态，能够独立存在。软件则是物理硬件以外的逻辑部件，如程序、数据、文档，抽象无形，不能独立存在。

软件工程需要研究如何更有成效地研发软件、维护软件，要达到这个目标，则必然需要对软件有很好的认识。

20 世纪 50 年代，计算机主要用于解决复杂科学工程计算，核心问题是计算过程，因此，这时的软件就是程序指令。20 世纪 60 年代，为了适应计算机的大批量数据处理应用，数据从

早期程序中分离出来，因此，软件又多了数据这个逻辑要素。20 世纪 70 年代，为了适应软件系统的工程模式研发，用于描述软件结构、说明软件使用的软件文档又成为了需要独立对待的新的软件要素。

因此，目前看来，软件是由程序、数据、文档三个部分组成的。

1.2.1 软件特点

可以将软件与硬件进行比较，从中可以发现软件具有以下方面的特点：

（1）软件有对硬件不可缺失的依赖。

任何时候，软件都离不开硬件的支持。显而易见的是，必须要有磁盘、光盘、磁带这些有形的物理介质，抽象的逻辑软件才能存储；必须要有 CPU、内存的支持，抽象的逻辑软件才能进行物理运行；必须要有键盘、鼠标、显示器、打印设备的支持，软件中的抽象数据才能有物理的输入、输出。

（2）软件有不同于硬件的生产流程。

软件不是制造出来的，而是开发出来的。硬件生产的最大成本是复杂的制造工艺，而软件生产的最大成本则是分析与设计。例如，产品质量监控，硬件主要体现为对产品制造工艺的流程监控，而软件则主要体现为对产品的分析设计的流程监控。

软件生产还有不同于硬件生产的资源需求。通常，硬件生产对原材料、能源、生产设备等有较高的依赖度，并难免对周围环境带来污染。软件生产则主要依赖于人的智慧，而对原材料、能源、生产设备等有很低的依赖度，不会给周围环境带来污染。

（3）软件有不同于硬件的生命过程。

硬件在使用过程中往往会经历：磨合期、正常使用期、老化期三个阶段。

- 磨合期：最初使用阶段。由于硬件中的一些难以预料的缺陷，这个阶段往往会有一个相对较高的故障频率。
- 正常使用期：硬件在磨合之后的一个较长的稳定工作过程，故障频率相对较低。
- 老化期：硬件寿命接近终点时的阶段。由于长久使用，硬件将出现过度磨损，因此故障频率会越来越高，并逐渐失效而不能工作。

图 1-1 所示为硬件的生命过程曲线。显然，高质量硬件产品应有较低的磨合期故障频率与较长久的正常使用期。

软件也有类似于硬件的磨合期与正常使用期。

软件磨合期的高故障频率通常由软件中隐藏的错误或缺陷所致，并随着这些错误或缺陷的排除而进入正常使用期。然而，软件是用不坏的，因此不会出现磨损现象。实际上，只要软件的工作环境没有变化，并且软件本身不做改动，则在其进入正常使用期后，可以一直使用下去。

软件无被动磨损，但软件可主动进化。通常，软件在使用一定周期后，需要通过改版进化而获得新的生命力，以使软件能适应新的工作环境，能提供新的更完善的应用服务。必须注意的是，软件可因改版进化而引入新的错误或新的缺陷，并使软件需要重新经历磨合期。

图 1-2 所示为软件的生命过程曲线。显然，如同硬件一样，高质量软件也应该有较低的磨合期故障频率与较长久的正常使用期。

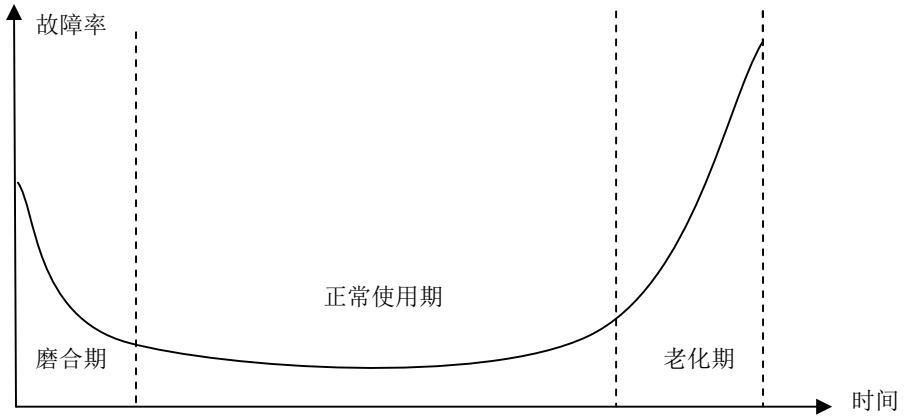

图 1-1　硬件产品生命过程曲线

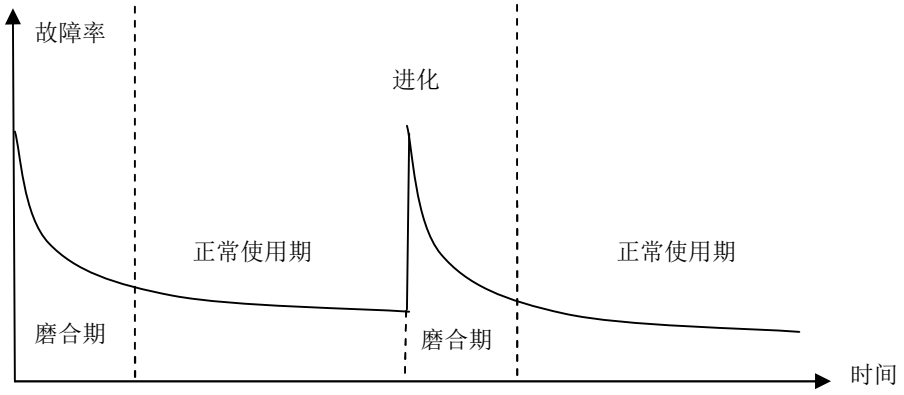

图 1-2　软件产品生命过程曲线

1.2.2　软件分类

软件分类有利于更好地认识软件。然而，同一个软件可以有多个不同的认识视角，如技术人员视角、管理人员视角、用户视角，因此也就可以按不同方式进行分类。

1. *按功能层次分类*

软件可依据功能地位而分为系统软件、支撑软件与应用软件，其关系如图 1-3 所示。

图 1-3　系统软件、支撑软件与应用软件之间的关系

（1）系统软件：为计算机底层软件，如操作系统、设备驱动程序、数据库引擎等。系统软件的特点是与计算机硬件紧密相关，并通过与硬件的频繁交互而对其他软件的运行提供底层

服务支持。例如操作系统，其建立在硬件环境基础上，并通过文件服务、进程服务、存储服务而给其他软件提供更加适宜的运行环境。

（2）支撑软件：介于系统软件与应用软件之间。工具软件是最常见的支撑软件，如程序编译器、程序编辑器、错误检测程序、程序资源库等。中间件软件也是支撑软件，根据用途不同可分为：数据中间件、对象中间件、通信中间件。可通过中间件的嵌入、组合或二次开发构造应用软件。

（3）应用软件：为最终用户提供应用服务的软件，通常由工具软件开发，并依靠系统软件的支持运行，如财务处理系统、生产控制系统、自动办公系统。早期的应用软件大多为离散结构，功能相互独立。但目前的应用软件则一般为多功能集成系统，如企业资源管理系统，即集成有企业中的各种资源的管理与协调，涉及人力、设备、资金、客户、原材料、生产线、半成品、产品等诸多方面的管理与控制。

2. 按工作方式分类

（1）实时软件：能够对外部事件或外部环境变化做出及时响应的软件，如飞机自动导航程序、导弹目标跟踪程序、自动生产线监控程序。实时软件一般由数据收集器、数据分析器、响应控制器等诸多部件组成，如图 1-4 所示。通常，数据收集器负责从外部环境获取格式化信息；数据分析器负责对获取的格式化信息进行分析、判断；响应控制器则负责按分析器对外部事件的判断进行有效的行为应对。例如导弹目标跟踪程序，当导弹跟踪目标发生移动时，导弹可通过热感应方式获取移动信息，并通过数据收集器将移动信息格式化，然后通过分析器的分析、判断确定目标新的位置，再由响应控制器调整运行轨迹，以达到有效跟踪的目的。

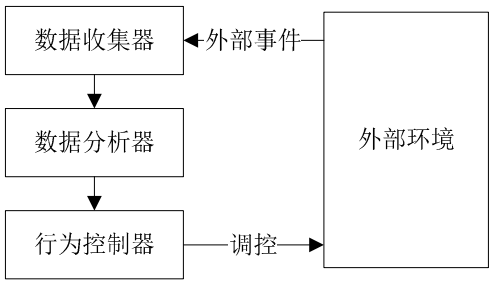

图 1-4　实时软件部件组成

（2）分时软件：能够把一个 CPU 工作时间段切片分配给多项任务的软件。如 Windows 操作系统，它即是一个具有分时功能的多任务操作系统，能使多个程序同步运行。与分时软件相关的软件技术是多线程机制，即在一个程序进程内执行多个线程任务。

（3）交互式软件：用于实现人机对话的软件，大多具有图形界面，如窗体程序、Web 页面。交互式软件的工作空间一般在客户端。窗体程序的特点是无论安装还是运行都在客户端。Web 页面则是以 HTML、XML 等文档格式安装于 Web 服务器，但运行空间仍在客户端，图 1-5 所示即为 Web 页面的工作特征。

（4）嵌入式软件：专门为特定智能设备提供自动控制的软件，如洗衣机流程控制程序、微波炉流程控制程序、汽车定速导航程序。嵌入式软件一般驻留在智能设备拥有的只读存储器中，因此与智能设备融为一体。

（5）批处理软件：能够成批处理大量数据或成批处理多项事务的软件。批处理软件通常

需要面对大量数据，并往往以某个时间点（如年终、月尾）为任务触发事件。可通过批处理而提高系统工作性能，如将可能影响系统日常性能的大量数据的处理安排在机器相对空闲时进行成批处理。可通过批处理而使事务操作延后，一些涉及多方协作而不便于同步操作的事务即可通过批处理而获得异步操作。

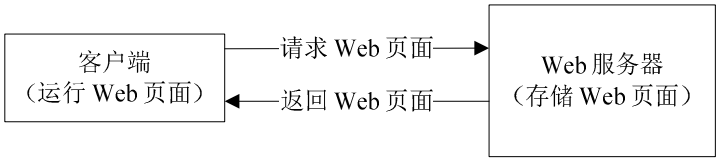

图 1-5　Web 页面的工作特征

3. 按规模分类

（1）微型软件：源程序代码行数 500 行语句以内的程序，通常一个人几天内即可完成开发。

（2）小型软件：源程序代码行数 2000 行以内的程序，通常一个人半年之内即可完成开发，大多为用户自主开发。

（3）中型软件：源程序代码行数 5 万行以内，并具有一定复杂度的软件系统，通常一年内可完成开发。

（4）大型软件：源程序代码行数 5 万行以上，具有综合功能，并涉及多用户任务协作的相当复杂的软件系统。

4. 按服务对象分类

（1）用户定制软件：开发机构受特定客户委托而开发的软件，如某特殊设备的控制系统、某专门企业的业务管理系统、某特定大厦的智能监控系统、某城市的交通监管系统。大多以招投标方式委托开发，并需要通过合同确定开发机构与客户之间的责权。

（2）通用商业软件：开发机构根据市场需求自主开发的面向广大用户的软件，如通用办公系统、通用财务系统。要求有面向用户的应用配置，以对各种复杂工作环境有良好的适应性。

1.3　为什么会发生软件危机

20 世纪 60 年代中期发生了软件危机，其严重影响了软件开发者的自信心。通常看来，是软件危机催生了软件工程，并通过软件工程克服了软件危机，重建了软件开发者的自信心。因此，软件危机一直被软件工程研究者重视。

软件危机主要表现在以下几个方面：

（1）软件开发成本、进度失控。许多软件尽管在开发之前已有了看似周密的成本估算，但实际开发费用则往往远远超出前期预算。开发进度难以控制，已定的进度计划可能是三个月提交产品，但实际则是半年过去了，而能够真正投入使用的最终产品却仍是遥遥无期。

（2）软件质量不能获得有效保证。花费大量资金、人力开发出来的软件可能不能正常使用，经常出故障，性能不稳定。不重视分析设计，只希望能尽快编写出可运行程序交用户使用，而为了压成本、赶进度，一些必须要有的软件测试也被省去了。显然，以这种方式开发出来的软件，其质量很难有保证。

（3）软件不能满足用户应用需要。尽管开发者自己觉得开发出了一个还算不错的软件，然而用户不买账，认为提交给他们的软件与开发者在合同中承诺的软件并不一致，一些必须要有的功能给遗漏了，程序工作流程与实际业务流程也可能不一致。

（4）软件可维护性差。软件难免需要进行错误修正或功能修补。然而，当开发者需要修正或改进软件时，则发现这是一件非常头痛的事情，原因是维护中需要依赖的设计文档很不完整，很多程序根本就找不到对应的设计说明。有些程序尽管有设计说明，但源程序已有多处改动，与设计说明严重不符。

为什么会发生软件危机呢？通常看来，有主客观两个方面的原因。

1. 客观原因

（1）软件的逻辑特性。软件是计算机系统中的逻辑部件，这使得软件缺陷具有了较大的隐蔽性，缺少硬件缺陷所具有的直观表象，难以发现。通常，硬件缺陷会有发热、噪音等不正常现象相伴随，它有一个逐渐变坏的过程，大多可以在没有引发严重故障之前被发现。软件缺陷则缺乏硬件缺陷的直观表象，必须等到错误发生前提条件成立时故障才会出现，如软件中的编码错误就必须等到出错的这行代码被执行时才有可能被发现。

（2）软件的复杂性。软件成为了工业产品之后，其规模越来越大，复杂程度越来越高。实际上，许多庞大的软件系统，其复杂程度已经远远超出了常人的理解力。人们寄希望于软件工具支持高度复杂软件系统的开发。然而直到今天，用于软件开发的工具仍很不完善，许多复杂软件问题的解决仍还不得不依靠人基于经验的决策判断。

（3）软件的产业发展状况。20 世纪 60 年代，软件产业才刚刚起步，缺乏工程规范、技术标准。就算是今天，即使有了一定的工程规范，但实施起来仍还是难度很大，并没有发挥其应有的作用。以 CMM 认证体系为例，这是一个有关软件工程过程的达标认证，用来反映软件企业的产业化完善程度，有 5 个认证级别，要求逐级认证。然而，在世界范围内，能够通过 3 级 CMM 认证的软件企业也不是很多。

2. 主观原因

（1）漠视用户需求。软件技术与用户需求是软件开发的两个支撑点，但软件开发者往往是一头热，他们的热情几乎都落到了软件技术上，因此用户需求成了冰点，被忽视甚至漠不关心。然而，软件毕竟是为用户开发，漠视用户需求必然使软件面临应用上的困难。漠视用户需求还给开发者带来了关门搞开发的陋习，很多通用软件就是按照这种方式开发出来的，开发之前没有进行广泛有效的市场调研，以致开发出来的产品不能获得市场的热情响应。这个问题也反映到了委托方式的定制软件上，形式上虽有一个需求分析期，但实质上也就是要求用户填写一封需求调研表格，紧接着就是软件开发，必须要有的现场调研、资料收集、需求论证被忽视了，以致开发出来的软件与用户的实际业务有冲突。

（2）重结果轻过程。软件开发的过程透明度一直很低，这既与软件的逻辑本性有关，也与早期开发中形成的个人创作工作模式有关。由于软件过程的低透明度，以致软件项目负责人、评审人员、用户都只能凭借最后结果对软件作出评价。问题的关键是，由于软件过程不透明，因此我们不关心过程，于是过程变得越来越不透明。但实质上，软件开发过程的透明度是可以提高的，只是我们为此需要花费更多的时间与精力，需要做更多的事情，如记录工作日志、加强配置管理。提高软件过程的透明度也是很重要的，过程决定结果，透明的过程将带来透明的结果，其有利于软件变更、维护与质量追踪。

（3）个人英雄理念。很少有产品如软件那样对个人素质有如此大的依赖，也正因如此，软件开发极易培养并强化开发者的个人权威。应该说，对于早期软件开发，个人英雄理念或许是不可避免的，甚至是必要的。然而在今天，面对既庞大又复杂的软件系统，个人英雄理念则带来了越来越多的弊端。由于个人英雄理念，软件产品开发的成败不得不与个人作为相关联。但个人的能力总是有限的，会受外部环境、自身情绪等诸多因素制约。个人也有可能离开软件开发团队。更加严重的是，过于强化的个人英雄理念还必然会对标准、规范带来冲击，会使本应该遵守的制度受到破坏。

1.4 软件工程技术

影响软件的技术因素很多，但从工程角度看，则主要反映在软件过程、工程方法与软件工具这三个方面。

1.4.1 软件过程

软件过程是软件开发与维护的实施路线与具体步骤，并是软件开发时的工程化框架，可作为工程方法与软件工具得以有效应用的基础。

软件过程将整个软件开发划分为诸多个过程域，每个过程域则包含任务项、方法、工具、标准、规程等诸多工程元素。

软件过程涉及的内容有：

（1）划分工程过程域。

（2）确定过程域中的任务项。

（3）过程域资源配置，包括任务项、方法、标准、工具等。

（4）过程域里程碑设置，如过程的启动前提、验收标准。

应该说，不同的软件项目将会面临不同的软件任务，因此需要不同的软件过程支持。但是，软件定义、软件开发、软件验证与软件维护则为四个基本过程域，是大多数软件项目都将经历的过程域。

四个基本过程域的特征是：

（1）软件定义：确定软件技术规格和使用规则。

（2）软件开发：按照软件定义开发软件产品。

（3）软件验证：按照软件定义对开发出来的软件成果进行验证确认。

（4）软件维护：修正软件缺陷，并随用户需求变化不断改进软件。

上述描述中，我们只是看到了软件过程的一些轮廓。实际应用中，为使软件过程有更多的操作细节，开发者往往会从过程模式中选择合适的软件过程应用于软件项目，如瀑布模式、原型进化模式、增量模式。

需要认识到的是，对于工程应用，诸多过程模式仅具有参考性，实际软件开发中往往需要根据具体的软件任务、软件开发机构的自身特点对过程模式进行过程改进，以使软件开发能够获得更加良好的过程支持。

良好软件过程的特征是：软件开发能够有效按照一定的工业流程作业，可对开发任务进行量化管理，可使软件方法与工具得到有效应用，可使软件质量得到有效监控，可显著提高软

件开发效率。实际应用中，开发者总是根据项目需要以上述特征的一个或几个为依据对模式进行过程改进。

1.4.2 工程方法

软件工程方法所指的是开发与维护软件时应该“如何做”的一系列技术性方法。

工程方法涉及的内容有：工程规范、工程策略、技术手段等。其中，工程规范是对工程行为的约束，用于规定哪些能做，哪些不能做；工程策略则体现为一种工程路线，以决定在诸多可行方案中最终将采用的是什么方案；技术手段则是在已定工程策略前提下的具体做法，如通过 ER 图分析系统数据关系，通过数据流图分析系统功能结构，使用组件技术构造分布式软件系统。

软件工程方法需要适应软件过程，因此，也就需要考虑不同过程中工程方法的关联性。显然，为使不同阶段的工程方法能有较好的关联性，工程方法需要形成体系（如结构化方法体系、面向对象方法体系），以支持从软件分析到软件设计、实现的全过程任务开展。

1.4.3 软件工具

软件工具是指对软件工程方法与软件过程的自动化或半自动化支持。软件工具也如工程方法一样，要求能够覆盖整个软件过程，如项目管理、软件分析、软件设计、程序创建、软件测试等，都要求有合适的软件工具支持。表 1-1 所示为一些常用的软件工具。

表 1-1 常用软件工具类别

工具类别	举例
项目管理工具	项目规划编辑器、用户需求跟踪器、软件版本管理器
软件分析工具	数据字典管理器、分析建模编辑器
软件设计工具	用户界面设计器、软件结构设计器、代码框架生成器
程序创建工具	程序编辑器、程序编译器、程序解释器、程序分析器
软件测试工具	测试数据生成器、源程序调试器

软件工具还可按高端工具与低端工具分类。

高端工具为软件前期开发中需要用到的分析、设计工具，如数据字典管理器、分析建模编辑器。

低端工具则为软件后期开发中需要用到的与程序创建直接相关的工具，如程序编辑器、程序分析器、程序调试器。

可以将诸多软件工具集成，以使软件开发时的各项工作，如软件项目管理、软件分析设计、软件实现部署、软件版本控制，能够在一个统一的工作平台上进行。通常，这样的集成有多种软件工具的开发平台被称为计算机辅助软件工程（CASE——Computer Aided Software Engineering），如图 1-6 所示。显然，软件工具的集成有利于软件开发资源的集中管理，有利于软件过程的自动化控制，可使软件工具更能发挥工程作用。

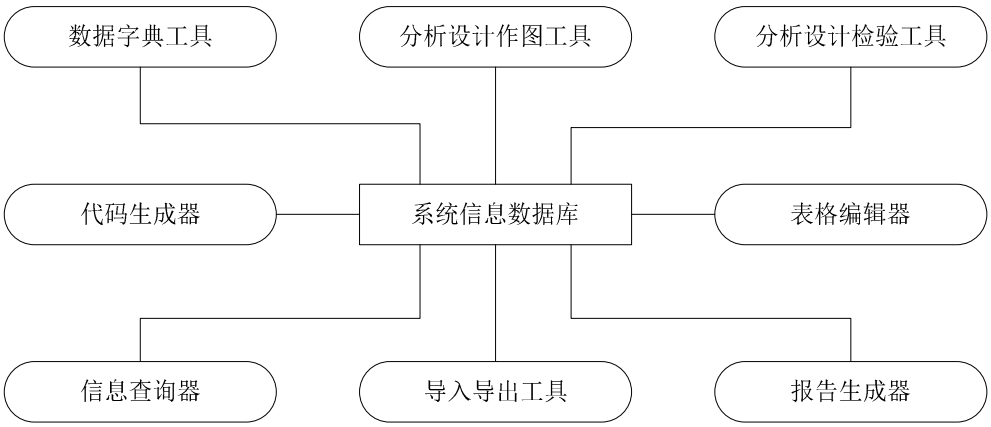

图 1-6　CASE 框架

1.5　软件工程管理

软件工程应用的好坏不仅依赖工程技术，也依赖工程管理。许多软件项目之所以失败，并不是因为缺少先进技术与优秀技术人才，而是因为管理混乱。因此，软件工程必须研究工程管理。

软件工程管理主要体现在四个 P 上，即项目（Project）、人员（People）、过程（Process）、产品（Product）。它们是软件工程管理中必然涉及的四个要素。

1.5.1　项目计划

项目是可量化的工程单位，所涉及的量化指标如项目周期、项目规模。软件开发需要以项目为单位实施。

软件项目使软件开发中的各种因素，如任务、人员、设备、费用、产品等集中到了一起，并通过建立一定的项目规则、制度，而使软件开发有了管理的便利。

项目管理的首要工作是制定项目计划。这即是说，在软件开发工作起步的时候，就应以项目任务、项目周期、工程环境为依据制定出科学合理的软件开发计划。

项目计划是软件开发的工作指南，内容涉及：项目任务分解、人员配置、资源配置、成本估算、进度安排等。

1.5.2　人员组织

软件开发是智力型劳动，必须由人来完成。应该说，开发人员，尤其是优秀技术人才是决定软件能否成功开发的最关键因素。然而，软件开发光有人还不行，还需要把人组织起来，并将人安排到软件项目中的合适岗位上承担开发任务。

软件开发通过软件项目组将人组织成开发团队，项目组成员则有项目负责人、开发人员、资料管理员、软件测试员等。

中小规模的软件开发往往由一个项目组即可承担全部开发任务。然而，对于较大规模的

软件系统，则需要把一个大项目分解为多个子项目，并需要组建多个项目组承担开发任务。软件项目组有许多不同形式的组织结构，如民主分散形式、核心领导形式。

民主分散形式的特点是小组成员完全平等，享有充分民主，通过协商做出项目决策，诸如项目计划、任务分配、工作制度、设计方案等都需要集体讨论。

核心领导形式的特点是项目组中有一位核心成员，他全面负责项目任务，起领导作用。比起民主分散形式，核心领导形式的组织结构更加严密，成员任务更加明确。

1.5.3 过程管理

前面我们已从技术层面对软件工程过程有过说明，它是软件工程的技术基础，是一个工程框架，可将软件开发分解为多个任务阶段或过程域。

过程管理的第一项工作是选择一个与所承担的软件项目相适应的过程模式。可供选择的过程模型有：瀑布模式、原型模式、增量模式、螺旋模式。许多时候，还可能需要根据软件任务、工程环境的特殊性对所选过程模式做一些必要的适应性改动，以获得更好的工程框架支持。

过程管理的第二项工作是基于所选过程模型制定出更加详细的里程碑过程计划，以便于软件开发能基于各个里程碑获得有效的过程控制。

软件开发需要有各种资源支持，如人员、设备、软件工具、技术资料等。因此，过程管理的第三项工作是基于里程碑过程计划的资源调配，以使得软件项目能够按计划顺利进行，同时又不会有太大的资源浪费。

1.5.4 产品管理

1. 产品质量保证

软件生产需要有较完善的质量保证体系，其涉及的要素有：

- 质量标准：有关软件质量的框架性规定，包括产品标准、过程标准。
- 质量计划：说明软件产品质量要求，规定软件产品质量的评定方法，包括产品质量计划、过程质量计划、质量目标。
- 质量控制：监督整个软件开发过程，以确保质量，保证规程和标准被严格执行。可采取的质量控制手段主要有质量评审、质量走查、质量评估。

2. 产品配置管理

为对软件生产进行过程追踪，还需要建立有效的配置管理，其主要内容有：

- 标识软件配置项：软件开发中需要存留的资源（程序、文档、数据），要求有配置项命名，以利于对其实施配置管理。
- 控制软件变更：建立软件变更制度，以达到对软件变更的有效控制。
- 控制软件版本：制定软件新版发行规则，以达到对软件版本升级的有效控制。

1.5.5 工程目标

软件工程目标是软件工程价值的体现，并是评价软件工程应用效果的基本依据。实际项目中，主要涉及以下几个方面的工程目标：

（1）降低软件开发与维护成本。

（2）使软件满足用户应用需要。
（3）尽量改善软件性能。
（4）尽量提高软件可靠性。
（5）使软件易于使用、维护与移植。
（6）能按时完成软件开发任务，并及时交付使用。

实际上，很难在一个软件项目中达到所有目标，然而必须有所考虑，并尽力追求。软件项目中的工程目标之间还有可能发生冲突。例如，软件成本、软件性能、软件可靠性、软件可维护性之间就有可能发生冲突。或许我们希望降低软件成本，但一味地降低成本，则难免影响软件质量，使软件的可靠性下降。或许我们希望软件有高性能，但为了获得高性能，则可能不得不使用低级语言实现软件，而这会使软件开发工作量加大，使软件成本上升，并会使软件难于理解、难于维护、难于移植。

一般的看法是，开发者应根据所承担的软件任务的特点，根据所处的工程环境，根据资金投入状况，根据开发人员的技术素质，选择一个合适的通过努力可以达到的工程目标作为工程定位，而不是盲目追求高目标。如果项目中存在目标冲突，则还需要根据实际情况对诸多工程目标进行优先排序。

1.6 主流软件工程方法学

1.6.1 结构化方法学

20 世纪 60 年代末，诞生了结构化程序语言（如 Pascal、C 语言），它们带来了以功能为目标的编程风格，可依靠程序函数建立程序模块，并且每一个程序模块都有确定的功能任务。结构化程序语言取得了极有成效的应用，成为了那个时期的主流程序语言。

结构化方法学即由结构化程序语言的广泛应用催生，以满足基于功能的软件开发，可支持整个软件生命周期，涉及：结构化分析（SA——Structure Analysis）、结构化设计（SD——Structure Design）与结构化实现（SP——Structure Particular）。

结构化方法是以实现软件功能为基本目标，结构化分析的任务是搞清楚软件需要哪些功能要素。数据流模型是结构化分析中最常使用的建模技术，可逐层描述软件系统内部的数据加工细节，可有效获取软件系统中的功能要素。

图 1-7 所示是某考卷处理系统数据流模型图，可描述该考卷处理系统数据的加工过程。

结构化设计则以分析获取的功能元素为依据，定义功能化程序模块，设计程序算法。

从数据流图中获取的功能元素可映射为程序模块。根据各功能元素对数据的加工顺序与协作关系则可映射出基于模块的程序结构。

程序结构图是软件设计时需要建立的程序结构模型，可描述程序系统基于程序模块的系统集成。图 1-8 所示是考卷处理系统程序结构图，可说明该考卷处理系统的程序构造。

1.6.2 面向对象方法学

面向对象方法源于 20 世纪 80 年代起逐步发展起来的面向对象程序技术，引入了许多新的程序概念，如类体、对象、消息等，并有不同于结构化的程序构造与工程方法。

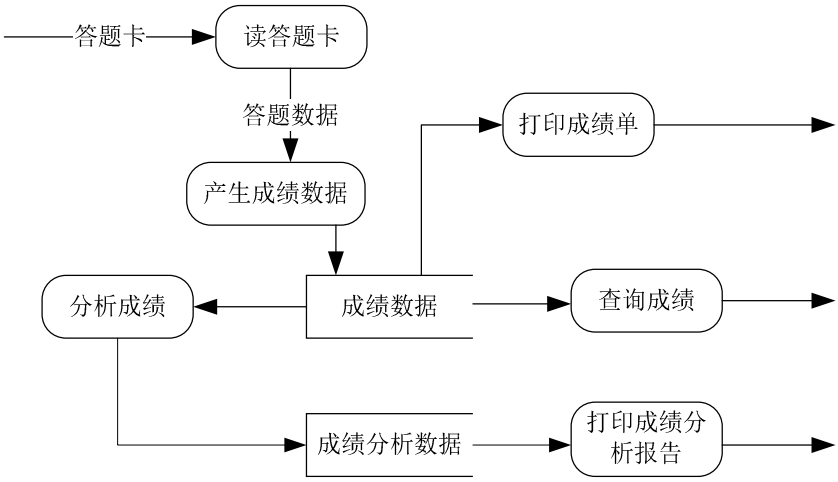

图 1-7　结构化数据流模型

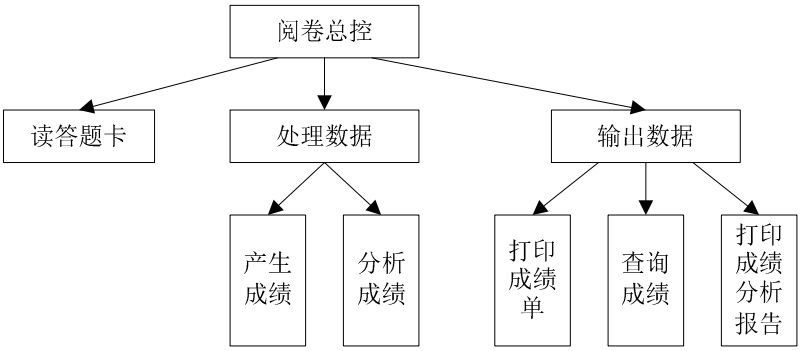

图 1-8　结构化程序结构

1. *面向对象程序*

类体是面向对象程序的基本模块单位。面向对象程序系统即由诸多类体构造。例如教学管理程序，其构造元素中即需要有教师、学生、课程等诸多与教学活动有关的类体。

类体涉及属性、操作两个方面的程序要素说明，其中的属性可定义为一组类体成员变量，操作则可定义为一组类体成员函数。例如，图 1-9 所示的学生类，其中的学号、姓名、专业即是它的属性，学籍注册、学业查询即是它的操作函数。

对象由类生成，是类模块的实例化结果。例如，图 1-10 中的 S 对象，它即由图 1-9 中的学生类生成。一个对象可调用另一个对象的操作函数，以实现对象之间的通信或交互，这被叫做消息发送。例如，图 1-10 中的教师对象 T 向学生对象 S 发出的学籍注册消息。

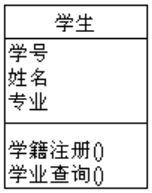

图 1-9　类体

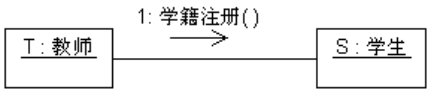

图 1-10　对象及其消息发送

数据封装、类体继承、操作多态被看做是面向对象程序技术的三个基本特性。

（1）数据封装：是指同一个类的不同对象将分别拥有各自独立的属性值，并且对象的属性值往往被限制为只能由其自身操作改变。

（2）类体继承：是指程序诸多类可组织成为一个具有等级关系的层次结构，处于上层位置的类叫做基类，处于下层位置的类叫做派生类，其中下层派生类除具有自己的属性、操作外，还可继承有上层基类的属性、操作。

（3）操作多态：是指多个同名函数可以有不同的功能用途。多态性的一种形态是操作函数重载，这即是在类中建立多个同名操作函数，但这些函数有不同的参数，并有不同的实现代码。动态联编是多态性的又一种表现形态，其体现为由抽象类定义的虚操作函数，仅用于提供消息格式，具体行为则取决于下级子类中的同名操作函数的具体代码实现。

2. *面向对象工程方法*

面向对象工程方法是为适应面向对象程序开发而产生的一整套工程规则。面向对象工程方法的基本特征是程序系统可基于现实实体构建。其中，类体被用来定义实体，由类体生成的对象则被用来进行实体行为仿真。

面向对象方法也涉及面向对象分析（OOA——Object Oriented Analysis）、面向对象设计（OOD——Object Oriented Design）与面向对象实现（OOA——Object Oriented Particular）三个任务阶段。

软件分析需要解决的核心问题是软件有多大的业务范围，有哪些业务步骤。诸多问题可通过建立分析模型给予解决，如业务用例模型、业务活动模型、类分析模型就是软件分析阶段需要建立的模型。

软件设计需要解决的核心问题是软件基于类体的内部构造，由对象扮演的动态元素的执行步骤。诸多问题可通过建立设计模型给予解决，如类设计模型、对象协作模型、组件模型就是软件设计阶段需要建立的模型。

比较结构化方法，面向对象方法可体现出以下方面的优越性：

（1）便利的由分析到设计的转换通道。

面向对象程序由类体构造。面向对象分析中即需要建立类关系模型，并且诸多从软件问题分析中获取的业务类可直接映射为软件设计中的实体类。显然，这有利于面向对象分析到面向对象设计的过渡。

（2）更加接近现实环境。

可以说，面向对象技术使计算机观点被更进一步淡化，现实世界模型成为了构造系统的最主要依据。因此，面向对象方法要求开发者更多地使用业务领域中的概念思考软件问题，并要求开发者与软件用户有更加广泛的交流。

（3）更加有效的程序复用手段。

结构化的软件重用技术是建立于标准函数库上的，而大多数的标准函数只能提供一些基本功能，因此复用粒度较小，不能很好地满足大型应用软件的复用需要。

面向对象的软件重用技术则建立于派生类对基类的继承上，因此有更大的复用粒度，可带来更高的开发效率。

（4）可使软件以迭代方式逐步完善。

面向对象程序类体具有继承性，其支持软件通过派生子类而逐步扩充功能。因此，面向

对象程序可逐步创建，程序产品可逐步提交与完善。项目初期可以只是建立一个具有基本用途的应用系统，但可通过建立派生类而逐步扩充与完善功能。

1.7 常用软件工具

软件开发需要有软件工具的支持。下面是软件开发中一些常用的软件工具，本书后面各章节的建模描述即通过这些工具创建。

1.7.1 Microsoft Visio

Visio 是 Microsoft 公司开发的一个通用的图形建模工具，能够创建很多领域的图形模型。Visio 可作为软件工具使用，并能建立软件开发中需要用到的各种图形，如数据流图、数据表关系图、程序结构图、UML 模型图、界面原型图、项目进程图、组织结构图等都能通过 Visio 创建。图 1-11 所示为 Visio 对软件开发的部分作图支持。

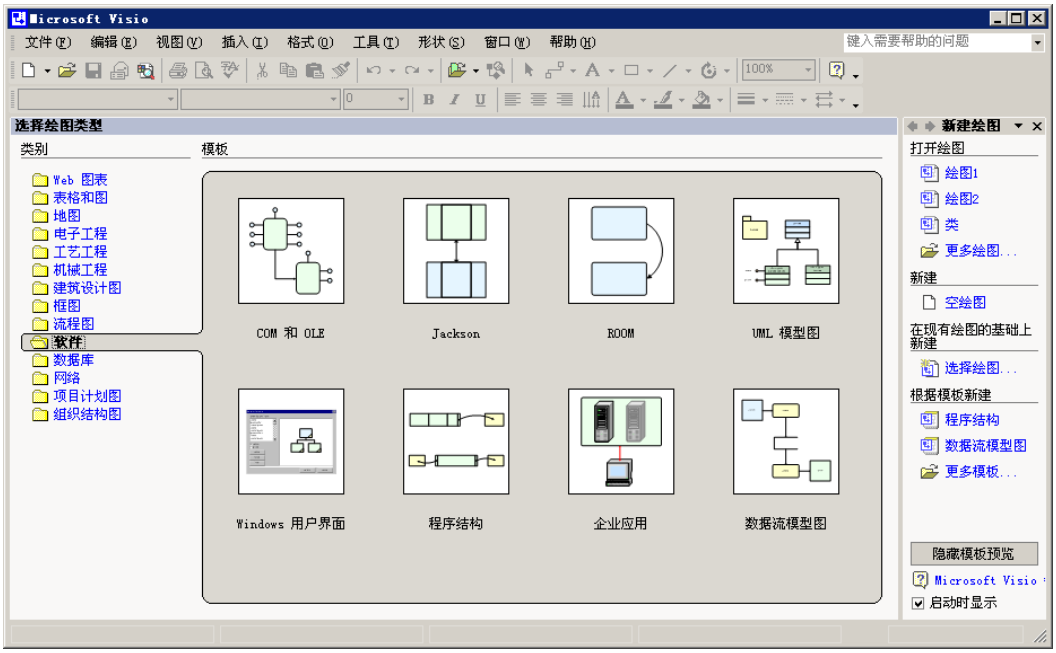

图 1-11 Visio 对软件开发的作图支持

Visio 的优点是简单实用，并能用来建立多种图形。本书结构化分析设计建模中的数据流图、程序结构图均是通过 Visio 创建。

Visio 还与 Visual Studio .NET 有很好的集成。例如，在 Visio 中创建的 UML 模型就能够通过正向工程转换为 Visual Studio .NET 中的代码框架，而在 Visual Studio .NET 中创建的程序系统则能够通过逆向工程转换成 Visio 中的软件模型。

1.7.2 Sybase PowerDesigner

PowerDesigner 由 Sybase 公司开发，是一个专门用于数据库的 CASE 工具。PowerDesigner

能够支持数据库的全过程建模，涉及的模型有：业务过程模型（BPM——Business Process Management）、概念数据模型（CDM——Coceptual Data Model）、物理数据模型（PDM——Physics Data Model）、面向对象模型（OOM——Object Oriented Model）。

PowerDesigner 还提供了极有成效的数据工程过程支持，可实现从概念数据模型到物理数据模型，再到面向对象模型的有效建模映射。

图 1-12 所示为 PowerDesigner 概念数据建模环境。本书中的数据库模型均是通过 PowerDesigner 建立。

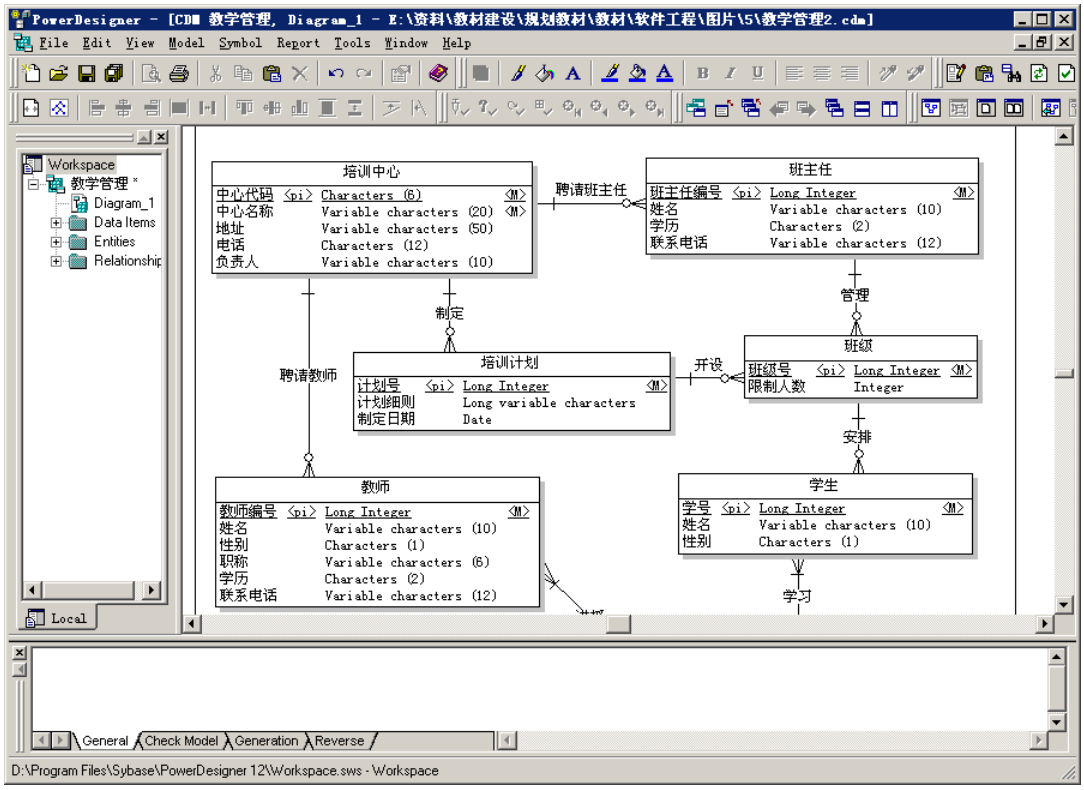

图 1-12　PowerDesigner 概念数据建模环境

1.7.3　IBM Rational Rose

Rose 由 Rational 公司开发，并随着 IBM 公司对 Rational 公司的收购而成为了 IBM 的产品。UML 建模语言的创建者 Booch、Rumbaugh 与 Jacobson 都曾服务于 Rational 公司与 IBM 公司，这使得 Rose 成为了最具影响力的 UML 建模工具。

Rose 提供了从用户业务领域到系统逻辑分析与设计，到系统物理设计与部署的对 UML 的全面建模支持，并提供了很好的模型映射机制，可使 UML 所倡导的基于迭代的统一开发过程得到有效的应用。

Rose 的作用还体现在对低端开发环境的正向工程与逆向工程的支持上，所建模型能够通过代码生成工具产生出低端开发环境所能识别的程序框架，而在低端开发环境中建立的程序清单则能够通过逆向工程而被抽象为系统模型。

图 1-13 所示为 Rose 中设计类图建模环境。本书中的诸多面向对象分析设计模型就是通过 Rose 建立。

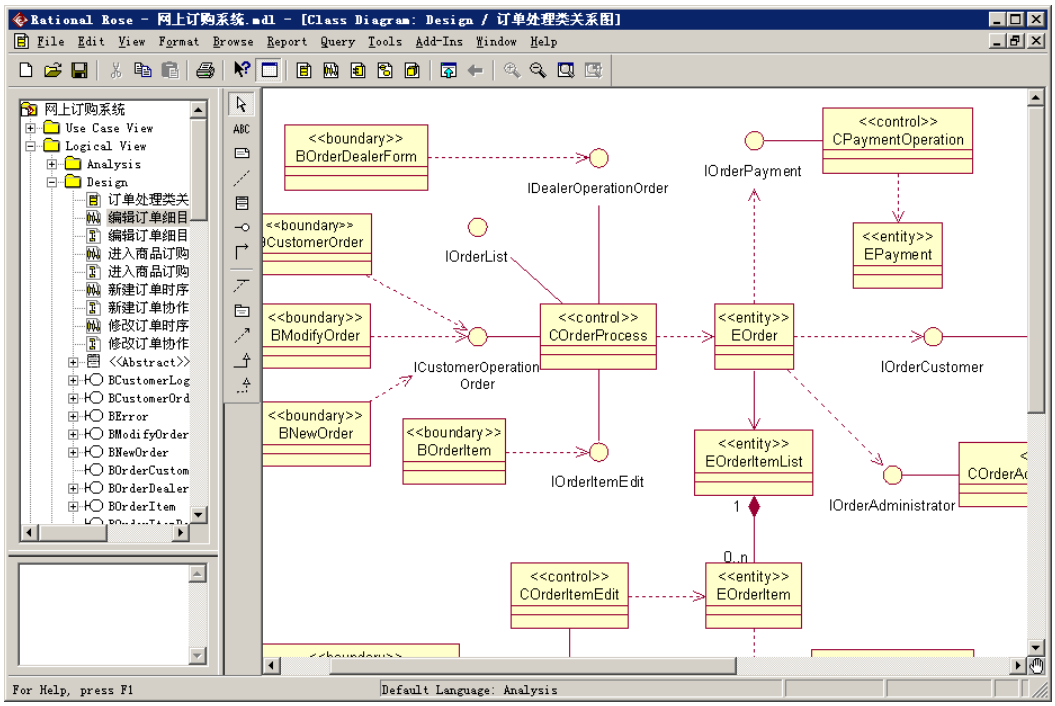

图 1-13　Rose 类图建模环境

小结

1. 软件工程

软件工程是关于软件开发、使用与维护的工程方法学，并是工程技术、工程管理与工程经济的有机综合。

2. 软件特点

软件是计算机系统中的逻辑成分，是程序、数据、文档等诸多逻辑元素的集合，需要有物理硬件的支持才能产生作用。软件不是制造出来的，而是开发出来的，其最大的生产成本是分析与设计。软件是用不坏的，然而却会因环境的改变而失效。因此，软件需要更新进化，以延长其生命周期。

3. 软件危机

20 世纪 60 年代中期，软件开发者遭遇到了软件危机。主要危机现象有：软件开发成本和进度难估算、软件质量没有保证、软件不能满足应用需求、软件缺乏可维护性。

危机的客观原因：软件的逻辑隐蔽性、软件的复杂性、软件的低产业发展水平。

危机的主观原因：漠视用户需求、重结果轻过程、个人英雄理念。

4. 软件工程技术、管理与目标

软件工程涉及的技术问题有：

- 软件过程：实现软件的步骤与工程框架。
- 工程方法：实现软件的技术性要素，如工程规范、工程策略、技术手段等。
- 软件工具：对工程方法与软件过程的自动化或半自动化支持。

软件工程涉及的管理问题有：项目管理、人员管理、过程管理、产品管理。

软件工程还必须考虑工程目标，以体现其工程价值。一些主要的工程目标是：降低成本、满足需求、改善性能、提高质量、及时交付。

5. 主流工程方法学

结构化方法学是传统的主流方法学，以功能为基本元素，包括结构化分析（SA）、结构化设计（SD）与结构化实现（SP），可对整个软件生命周期提供方法学支持。

面向对象方法学则是目前的主流方法学，包括面向对象分析（OOA）、面向对象设计（OOD）与面向对象实现（OOA），可对整个软件生命周期提供方法学支持。其以实体为基本元素，如类体、对象，并可使程序系统基于现实实体构建，更加接近现实环境。

6. 常用软件工具

软件工程还需要有软件分析设计工具的支持，常用的软件分析设计工具有：

- Microsoft Visio：通用图形建模工具，可支持结构化分析设计建模。
- Sybase PowerDesigner：专门的数据库建模工具。
- IBM Rational Rose：专门的 UML 建模工具。

习题

1．什么是软件工程？其对软件产业化发展有什么积极意义？

2．软件有哪些特点？按照功能层次，软件可分为：系统软件、支撑软件、应用软件。那么，SQL Server 是哪个层次的软件？ADO.NET 是哪个层次的软件？Visual C++是哪个层次的软件？

3．按照服务对象，软件可分为：用户定制软件、通用商业软件。试举例说明这两类软件的特点。

4．软件危机有哪些现象？为什么会发生软件危机？应该如何有效地预防？

5．软件工程涉及过程、方法、工具三个方面的技术问题，这三个方面存在怎样的相互关系？试举例说明它们之间的关系。

6．软件工程管理主要体现在四个 P 上，即项目（Project）、人员（People）、过程（Process）、产品（Product）。请简述这四个方面的管理，并谈一些自己的认识。

7．软件工程必须考虑工程目标，以体现其工程价值。一些主要的工程目标有：降低成本、满足需求、改善性能、提高质量、及时交付。这些目标谁更重要呢？请按照你所认识到的重要性对上述工程目标进行优先级排序。

8．结构化方法有什么特点？面向对象方法有什么特点？C 语言是结构化程序的代表，Java 是面向对象程序的代表，试以它们为依据说明结构化方法与面向对象方法的区别。

9．进入 Microsoft Visio，然后从其菜单“文件/新建/软件”中分别打开 UML 模型图、程序结构图、数据流模型图等模型工具，并进行你所能做的各种操作。对此，你有什么操作感受？