# 第3章

# 系 统 建 设

系统建设是一项涉及面广、投资规模大、建设周期长、存在一定风险的系统性工程，是组织开发战略中的重要组成部分，它必须服务和服从于组织的总体目标，与组织的整体发展战略协调一致。因此，在决定启动系统建设后，需要对系统建设做出规划，即从组织的战略高度出发，把组织作为一个有机整体，全面考虑组织所持的环境、组织本身的潜力、具备的条件以及组织进一步发展的需要，等等。从总体上把握系统建设目标，所具有的功能框架，研究论证其可行性，为系统的分析、设计、实施打下良好的基础，这是保证系统建设顺利进行和成功应用的前提。

## 3.1 系统建设是复杂的社会过程

### 3.1.1 系统建设的复杂性

国内外历史事实告诉人们，系统建设的道路坎坷，许多已经建立的系统所带来的效益，远远不及预先的承诺与期望。系统建设中耗资巨额、效益难望，或半途而废，或使建设单位背上沉重包袱等情况时有发生。因此，系统建设者必须深刻理解系统建设工作的复杂性，正确认识其特点与规律，并且运用科学的建设方法。这对于成功地建设系统至关重要。

软件系统是一个规模庞大、结构复杂、具备多种功能、实现多个目标的大系统。软件系统的建设涉及的组织管理背景和所用的技术手段都很复杂，工作量大，资源昂贵，这些都是一般的工程技术开发项目难以比拟的。为了卓有成效地进行管理系统的建设，近十年来，陆续提出了不少建设方法。这些方法各有特色，其中有些已在应用中取得了较好的效果。在现有的众多系统建设方法中，基于系统生命周期的一类方法，特别是其中结构化方法和面向对象方法在实践中起过重要的作用，得到了广泛应用。

系统建设的复杂性主要体现在建设环境复杂、用户需求多样、建设内容复杂、技术手段复杂、所需资源密集这几方面。

### 3.1.2 系统开发是一个社会过程

软件系统建设是一个复杂的社会系统工程，建设、运行和管理一个系统涉及管理科学、

计算机技术、通信技术、运筹学、一般系统论和信息理论等多种学科的理论知识，并对企业的各个业务领域要有深刻地理解。因此，系统建设的不同阶段需要掌握不同知识层次的工作人员，也需要不同层次的业务人员来配合。

### 1. 业务人员的构成

就软件系统建设而言，业务人员具有双重身份，一方面是软件系统的最终用户；另一方面又是软件系统的开发人员，如果没有最终用户真正参与软件系统开发，不可能成功地进行软件系统建设。在总体设计阶段，要解决软件系统的战略规划、战略数据规划和技术策略的制定。涉及到全用户的信息需求、信息规划、管理机构和管理方法的改进，机构之间的协调、控制和业务重组等全局性的问题。需要在用户最高领导的倡导、支持和强有力的组织下才能进行。需求分析时，认识软件系统的主要责任应该由企业承担。因此系统规划员的理想人选应是既有丰富的本企业的业务知识和管理经验，又是软件系统方面行家里手，掌握一套成熟的科学方法，最好具有总体数据规划的成功实践经验。

应用系统开发阶段，业务人员和开发人员密切合作，仔细研究在新的环境下，如何改进业务流程，提高工作效率，由计算机来取代日常的业务工作；如何利用计算机计算速度快、存储容量大和分布存储等优点，建立新模型、采用新算法，从而在更高更深的层次上改进，使用户的管理产生质的飞跃。

### 2. 开发人员的构成

总体设计阶段迫切需要精通本企业主要业务领域的业务、又精通管理科学、对信息处理和计算机技术有一定了解、具有相当的组织管理能力的人才。由这样的人员担任系统规划员，侧重于软件系统的社会系统特征。依靠他们需求分析才能做得扎实可信。诸如业务重组这种远见卓识的见解才能得出，才不会把软件系统建设成为手工管理的“仿真系统”。为了始终突出软件系统的数据特性和系统特性，还需要系统分析人员和数据管理人员。在系统分析人员和数据管理人员的指导下，应用系统设计人员负责应用系统的详细设计和专业数据库的设计，还要完成网络设计和施工。熟悉计算机软硬件系统、网络和通信；熟悉结构化方法、面向对象程序设计方法，数据库设计规范理论、数据管理、信息分类编码标准化和高效率、一致性使用数据的原则。

(1) 系统分析人员负责总体设计和应用项目计划的编制和审查，侧重于软件系统的技术特征。注重系统中各局部的信息联系和协调性、系统的约束，熟悉大系统开发的方法论，需要掌握系统分析与设计理论，熟悉结构化方法、面向对象方法和信息工程等主流方法论，熟悉计算机软硬件系统、网络和通信。系统设计时应突出系统品质，以整体最优为目标，局部利益服从整体利益。

(2) 数据管理人员负责数据管理规范的制定、修改、发布与监督执行，负责总体数据规划和数据库建设计划的编制或审查，负责全企业数据资源的使用与管理。需要掌握系统分析与设计理论，熟悉结构化方法、面向对象方法和信息工程等主流方法论，熟悉数据库设计规范理论，熟悉信息分类编码标准化和高效率、一致性使用数据的原则；有能力对大型系统数据资源的规划、使用提出设计，能监督其他人员进行数据的逻辑设计和数据管理。因此数据管理人员是企业系统最为重要的技术中坚，没有资格胜任的数据管理人员，就不可能有成

功的数据管理，也很难建设一个具有稳定的、有序的数据环境的系统。

（3）程序员负责用系统规定的某种程序设计语言实现应用系统的详细设计，要求程序员熟悉规定的程序设计语言，有丰富的编程经验、良好的编程风格。

（4）软件系统运行管理还需要相应运行人员监视系统运行，及时调整各种参数，优化系统品质，做好病毒防治，保证安全运行。

参与项目开发的人员一般分为 4 类：用户、开发人员、项目管理者和高级管理者。软件系统各阶段对人员有不同的要求。图 3-1 给出了各类人员参与情况的示意图。

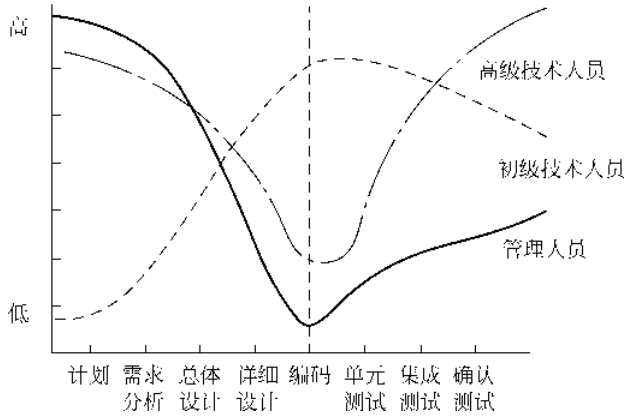


**图 3-1　各阶段技术与管理人员参与情况**

总之，软件系统是一个多学科、多领域、知识密集的高科技领域，特别是计算机、信息处理知识更新快，新理论、新概念、新方法、新工具层出不穷。网络设计和程序开发可以利用社会力量，寻求合作伙伴，但必须在企业自己系统级人员控制之下。这样做有两个明显的好处，一个是总体设计要指导、控制全部应用系统的开发，是相对稳定的，所以应该由企业自己的系统级人员来完成；另一个好处就是程序级的开发非常活跃，由有这方面特长的专业公司来开发，事半功倍，也有助于企业迅速跟上飞速发展的实现技术。我们要对软件系统有一个正确的、全面的认识，才能在当前变化纷繁的信息技术世界里少走弯路。

## 3.2　系统开发方法

软件开发模型是指开发软件项目的总体过程思路。软件开发方法是一种使用早已定义好的技术集及符号表示习惯组织软件生产过程的方法。其方法一般表述成一系列的步骤，每一步骤都与相应的技术和符号相关。软件开发的目标是在规定的投资时间内，开发出符合用户需求的高质量的软件。为了达到此目的，需要有成功的开发方法。

优秀的软件开发方法是克服软件危机的重要途径之一。因此，自软件工程诞生以来，人们重视软件开发方法的研究，已经提出了多种软件开发方法和技术，对软件工程及软件产业的发展起到了不可估量的作用。

### 3.2.1　结构化方法

结构化方法(structure method)是最早的、最传统的软件开发方法。20 世纪 70 年代初,就提出了用于编写程序的结构化程序设计方法,而后发展到用于设计的结构化设计(Structured Design,SD)方法、用于分析的结构化分析(Structured Analysis,SA)方法,以及结构化分析与设计技术(Structured Analysis and Design Technique,SADT)等; 面向数据结构的 Jackson 方法、Warnier 方法等。结构化方法由结构化分析、结构化设计和结构化程序设计构成,也称 Yourdon 方法。它适用于一般数据处理系统,是一种较流行的软件开发方法。在实际软件开发中使用的许多方法都是基于结构化分析与设计的改进方法。

所谓结构化分析,就是根据分解与抽象的原则,按照系统中数据处理的流程,用数据流图来建立系统的功能模型,从而完成需求分析。所谓结构化设计,就是根据模块独立性准则、软件结构准则,将数据流图转换为软件的体系结构,用软件结构图来建立系统的物理模型,实现系统的概要设计。所谓结构化程序设计,就是根据结构程序设计原理,将每个模块的功能用相应的标准控制结构表示出来,从而实现详细设计。

结构化方法总的指导思想是自顶向下、逐步求精。它是一种面向数据流的开发方法。它的基本原则是功能的分解与抽象。它是软件工程中最早出现的开发方法,特别适合于数据处理领域的问题。该方法简单实用,应用较广,相应的支持工具较多,技术成熟。

但是它不适应于规模大以及特别复杂的项目,该方法难以解决软件重用问题,难以适应需求变化的问题,难以彻底解决维护问题。这里提出的 3 个难点是由许许多多的惨痛事例总结出来的。最后的结论是不适应于规模大以及特别复杂的项目。那么,何为规模大? 何为特别复杂呢? 我们认为一个软件系统项目规模和复杂度可以参照下式。

- 不复杂/小规模: 系统项目规模/复杂度<8 人月。
- 中等复杂/规模: 8 人月<系统项目规模/复杂度<3 人年。
- 特别复杂/大规模: 3 人年<系统项目规模/复杂度。

### 3.2.2　Jackson 方法

Jackson 方法是一种面向数据结构的详细设计方法,也是一种较为流行的详细设计方法,Jackson 方法的发展可分为两个阶段。20 世纪 70 年代 Jackson 方法的核心是面向数据结构的设计,以数据驱动为特征。因为一个问题的数据结构与处理该问题数据结构的控制结构有着惊人的相似之处,该方法就是根据这一思想形成了最初的 JSP(Jackson Structure Programming)方法。该方法首先描述问题的输入/输出数据结构,分析其对应性,然后推出相应的程序结构,从而给出问题的软件过程描述。20 世纪 80 年代初开始,Jackson 方法已经演变到基于进程模型的事件驱动。许多软件设计书籍仍然将 Jackson 方法列为面向数据结构的设计方法。

Jackson 方法把问题分解为可由三种基本结构形式表示的各部分层次结构。这三种基本结构形式就是顺序、选择和重复。Jackson 方法提出一种与数据结构层次图非常相似的数据结构表示法,并提出一组基于这种数据结构到程序结构的映射和转换过程。

JSP 方法是以数据结构为驱动的，适合于小规模的项目。当输入数据结构与输出数据结构无对应关系时，难于应用该方法。基于 JSP 方法的局限性，又发展了 JSD(Jackson System Development)方法，它是 JSP 方法的扩充。

JSD 方法是一个完整的系统开发方法。该方法首先建立现实世界的模型，再确定系统的功能需求，对需求的描述特别强调了操作之间的时序性，它以事件作为驱动，是一种基于进程的开发方法，应用于时序特点较强的系统，包括数据处理系统和一些实时控制系统。

JSD 方法对客观世界及其同软件之间的关系认识不完整，所确立的软件系统实现结构过于复杂，软件结构说明的描述采用第三代语言，这不利于软件开发者对系统的理解以及开发者之间的通信交流，这些缺陷在很大程度上限制了人们实际运用 JSD 方法的热情。

### 3.2.3 维也纳开发方法

1969 年 IBM 公司维也纳实验室的研究小组为开发 PL/1 语言时，小组成员当时遇到如何对大型高级语言尽快用形式化说明来开发编译系统，使语法、语义的定义更严密、更系统化的问题。从软件系统最高一级抽象到最终目标代码生成，每一步都给出形式化说明的问题。最初提出了维也纳开发方法(Vienna Development Methad，VDM)。到现在维也纳开发方法已形成一种对大型系统软件形式化开发的较有潜力的方法，在欧洲及北美有相当大的影响，到 20 世纪 80 年代已将其应用到工程开发上。

VDM 是一种形式化的开发方法，软件的需求用严格的形式语言描述，把描述模型逐步变换成目标系统。VDM 是一个基于模型的方法，它的主要思想是：将软件系统当做模型来给予描述，具体说就是把软件的输入/输出看做模型对象，而这些对象在计算机内的状态可看做为该模型在对象上的操作。

VDM 从抽象说明开始，对软件系统功能条件给出定义，对其输入/输出用不同的数学域进行分类定义，这称为语法域说明。具体说明对象的真正含义，称为语义域说明。对系统在计算机内状态进行描述，称为加工函数(或语义函数)。前面的语义域和语法域都是用数学的域方程表示的，而加工函数是用数学函数形式表示的，所以 VDM 的软件系统模型是代数式的说明。

VDM 的每步开发借助于其强有力的描述工具语言 Meta_IV，开始在欧洲广泛应用。先是应用于开发程序语言的语义形式说明，以后变成一般软件的开发方法。

### 3.2.4 面向对象的开发方法

面向对象软件开发(Objected-Oriented Software Development)是近年来最流行的软件开发方法。但是，面向对象(Objected-Oriented，OO)的概念和思想却由来已久。有人认为，可以将 Dahl 与 Nygaard 在 1967 年推出的程序设计语言 SIMULA-67 作为面向对象的诞生标志。SIMULA-67 首先在程序中引入了对象概念。但是，面向对象真正的第一个里程碑应该是 1980 年 Smalltalk-80 的出现。Smalltalk-80 发展了 SIMULA-67 的对象和类的概念，并引入方法、消息、元类及协议等概念，所以有人将 Smalltalk-80 称为第一个面向对象语言。但是最后使面向对象广泛流行的则是面向对象的程序设计语言 C++。

什么是面向对象，面向对象应该具备哪些基本特征呢？面向对象方法是一种运用对象、类、继承、封装、聚合、消息传送、多态性等概念来构造系统的软件开发方法。

面向对象开发的基本出发点是尽可能按照人类认识世界的方法和思维方式来分析和解决问题。客观世界是由许多具体的事物、事件、概念和规则组成，这些均可看成对象。面向对象方法正是以对象作为最基本的元素，对象也是分析问题、解决问题的核心。由此可见，面向对象方法符合人类的认识规律。计算机实现的对象与真实世界的对象有一一对应的关系，不必做任何转换，这就使面向对象易于为人们所理解、接受和掌握。

面向对象开发方法包括面向对象分析、面向对象设计和面向对象实现。面向对象开发方法有 Booch 方法、Coad 方法和 OMT 方法等。

除了前面介绍的方法外，还有其他一些较成熟的软件开发方法，如适用于实时事务处理系统的有限状态机方法(Finite Status Machine，FSM)，适用于并发软件系统的 Petri 网方法等。有兴趣的读者可以参见有关资料。

## 3.3　软件开发工具

支持软件人员开发和维护软件活动而使用的软件一般称为软件工具。例如，项目估算工具、需求分析工具、设计工具、编码工具、测试工具和维护工具等。使用了软件工具可以提高软件生产率。

### 1. 工具箱

在软件开发的过程中，一般情况下是一种工具支持一种开发活动。开发过程中的活动较多，所以用的软件工具也多。于是将各种工具简单组合起来就构成工具包，人们将这种软件工具包形象地称为工具箱。工具箱的特点是工具界面不统一，工具内部无联系，工具切换由人操作。

因此，它们对大型软件的开发和维护的支持能力是有限的，即使可以使用众多的软件工具，但由于这些工具之间相互隔离、独立存在，无法支持一个统一的软件开发和维护过程。

### 2. 软件开发环境

工具箱的使用既有方便的地方，同时又存在着问题，为了使软件工具支持整个生存周期，人们将工具系统集成化，使之形成完整的软件开发环境。它不仅能支持软件开发和维护中的个别阶段，而且能支持从项目开发计划、需求分析、设计、编码、测试到维护等所有阶段，不仅支持各阶段中的技术工作，还要支持管理和操作工作，保持项目开发的高度可见性、可控制性和可追踪性。

### 3. 计算机辅助软件工程

为了软件工具在实现软件生存周期各个环节的自动化，软件工具正在发生很大的变化和不断建立新的软件工具。工具的共同点是让软件开发人员以对话的方式建立各种软件系统，因此称之为计算机辅助软件工程。可以将其定义为软件开发的自动化，简称为 CASE (Computer Aided Software Engineering)。为了实现软件生存周期各个环节的自动化并使

之成为一个整体，在自动化基础上，CASE 的实质是为软件开发提供一组优化集成的且能大量节省人力的软件开发工具。

CASE 技术是软件工具和软件开发方法的结合。它不同于以前的软件技术，因为它强调了解决整个软件开发过程的效率问题，而不是解决个别阶段的问题。由于它跨越了软件生存周期各个阶段，着眼于软件分析和设计以及实现和维护的自动化，因而在软件生存周期的两端解决了生产率问题。

CASE 工具与其他软件工具的区别体现在：支持专用的个人计算机环境；使用图形功能对软件系统进行说明并建立文档；将生存周期各阶段的工作连接在一起；收集和连接软件系统中从最初的软件需求到软件维护各个环节的所有信息；用人工智能技术实现软件开发和维护工作的自动化。

通常，结构化方法可使用瀑布模型、增量模型和螺旋模型进行开发；Jackson 方法可使用瀑布模型、增量模型进行开发；面向对象的开发方法一般是采用喷泉模型，也可用瀑布模型、增量模型进行开发，而形式化的维也纳方法只能用变换模型进行开发。

常用的开发环境包括 Microsoft 系列的 Visual C++、Visual Basic、Visual FoxPro 等，Borland 系列的 Delphi、C++ Builder 等，SyBase 系列的 PowerBuilder 等。下面将对部分开发环境和工具作简单概述。

### 3.3.1 Visual C++

Visual C++是略作扩展的 C/C++。

C++是完全支持面向对象的语言。C++是由 C 发展而来的，有很强的灵活性，也有自己的特长和不足。比如说，C++支持多重继承、模板、操作符重载、内联函数定义、预处理、宏、全局静态类变量、嵌套类定义等，但不支持虚构造函数、过程嵌套、内置集合类型、内置字符串类型、finally 构造等，在 rtti 方面，C++不如其他语言。

Visual C++作为开发平台，很重要的一点就是提供了应用框架——MFC。微软写 MFC 时必须考虑最大限度减少对语言本身的改动，而把功夫下在源代码级，以便能尽可能支持 ANSI 等标准，结果导致 MFC 的封装复杂而不直观(尤其是它对消息的封装)。太多的宏定义和含义模糊且自动生成、不得改动的注释使 MFC 乃至 VC 让很多新手望而生畏，不敢“下水”深入学习。

此外，Visual C++的编译和连接速度较慢，即使把 VC 的 incremental link 选项打开，Delphi 的编译和连接速度仍比 VC 快好几倍。并不是说微软的编译器不行，而是由 C++的复杂性决定的。模板的处理、预处理和宏的展开都是很费时的，为了克服编译的速度问题，C++编译器一般需要增强的连接器和预处理机制。

但是预处理机制仍然存在若干问题：

(1) 程序调试的断点行可能和代码行不同。

(2) 没有将最新的代码信息综合进去。

(3) 容易产生错误的逻辑。

(4) 因为读错文件头会容易产生类似 unexpected end of file 的错误。

C++编译器能识别无用的“死”代码，比如一个没有用的函数，等等。编译后的程序将不

包含这些多余的信息。

Visual C++编程使用 MFC 编译后的可执行文件通常很小，这相对其他平台而言比较小，主要是因为微软已经将系统运行库包含在 Windows 系统了。而 Visual C++的“edit and continue”功能，在调试中，是可以大幅度节省时间的。

在速度方面，Visual C++的编译和连接时的错误信息详细具体，特别是使用 atl 开发更加如此。

应用程序框架(application frame)，有时也称为对象框架。Visual C++采用的框架是 MFC。经过不断补充和完善，MFC 已经十分成熟。

在可移植性方面，MFC1.0 的程序也可以毫无障碍地在 VC 6.0 下编译通过。在集成界面方面，VC 在自动完成功能的智能化程度和提示详细程度很好，响应速度也比较快。Visual C++所带的 MSDN 是一部“开发者的百科全书”，信息庞大，查询方便，这方面非常专业，很多帮助项都有源程序示范。

调试方面，Visual C++调试功能非常强大，具有单步可视化调试、断点跟踪、运行时改变变量、鼠标指向可以得到变量值等功能。对动态链接库(Dynamic Linkable Library，DLL)的输入/输出也能方便地管理，能够进行源码级别的调试。

相对其他平台而言，Visual C++能够更加方便地看到变量的变化情况，这包括对结构可以展开成数据树，从而了解每一个变量的值，每一步调试，变化了的变量会加红，从而使调试更加方便。另外，Visual C++的块内存察看也很方便。

Visual C++对 com 有很好的支持，com 是组件对象模型的缩写，它是 OLE 和 ActiveX 技术的基础，com 定义了一组 API 和一个二进制标准，让不同的编程语言、不同平台的彼此独立的对象相互进行通信。com 是 Microsoft 制订的行业标准。Visual C++实现 com 编程有一种特殊的方法就是使用 ATL。ATL 使用 Visual C++特有的多重继承来实现 com 接口。虽然不见得使实现 com 服务和控制更容易，但是 ATL 和最新 com 技术的接口，基于模板的构造都比其他平台强。ATL 更有利于建立小巧、快捷的 com 组件程序。

按照目前通用的观点，Visual C++应用到 com 服务程序更有优势。

### 3.3.2　Visual Basic

Visual Basic 6.0 在原有 BASIC 语言的基础上进一步发展，至今包含了数百条语句、函数及关键字，其中很多和 Windows GUI 有直接的关系。

Visual Basic 主要有以下一些特点：

(1) 数据访问特性允许访问包括 Microsoft SQL Server 和其他企业数据库在内的大部分数据库格式，可建立与数据库有关的应用程序。

(2) 有了 ActiveX 技术就可以使用其他应用程序提供的功能，例如 Microsoft Word 字处理器、Microsoft Excel 电子数据表及其他 Windows 应用程序提供的功能。ActiveX 控件是 Visual Basic 工具箱的扩充部分，它特有的方法和属性大大增加了 VB 程序员的编程能力和灵活性。

(3) 针对 Internet 的功能，在 VB 的专业版和企业版中增加了大量开发 Internet 应用的新内容。例如 VB 提供了从开发环境中访问 Web、Internet 传输控件、Winsock 控件、超链

接定位、异步下载、Internet 部件下载等有关 Internet 特征。

(4) 已完成的应用程序是真正的.exe 文件,共用可自由发布的 DLL。由于 VB 是解释执行的,编译后生成的不是计算机的机器代码,而是伪代码。在运行的时候,VB 将每一句伪代码转换成机器代码,这样才能执行。虽然伪代码的兼容性较好,可在不同的操作平台上执行,但由于它毕竟不是机器代码,执行时有一个转换的过程,所以执行速度将会减慢。但是,在 VB 专业版或企业版中,既可以将代码编译成标准的 VB 伪代码格式,也可以编译成机器代码的格式。如果将程序直接编译成机器代码的格式,就取消了伪代码这一中间步骤,可大大提高程序的执行速度。

(5) VB 是采用了事件驱动模型来运行的。在这种模型的程序中,代码不是按照预定的路径进行——而是在响应不同的事件时执行不同的代码段。事件可以由用户的操作触发,也可以由来自操作系统或者其他应用程序的消息触发,甚至可以由用户操作触发。这些事件的顺序决定了代码执行的顺序,因此应用程序每次运行时所经过的代码路径都是不同的。

### 3.3.3 Delphi

Delphi 开发环境是基于 Object Pascal 语言的,Delphi 的编译和连接速度很快。由于 Object Pascal 没有模板、预处理和宏,这本来是缺点,但带来的一个好处就是编译速度极快。至于编译完的二进制代码,在打开相同的优化选项的情况下,Delphi 和 VC 执行速度并没有太大的差别。

Delphi 的 Object Pascal 因为没有“标准负担”,语言引入了组件、事件处理、属性等新特性。由于功夫做在编译器级,生成的源代码就显得十分简洁,简而言之,就是“让语言迁就框架”。VCL 对异常处理的支持很好,而一大缺点是对多线程支持差。VCL 的大部分都不是针对多线程优化的。

在调试方面,因为尽管 VCL 比 MFC 的抽象程度高,封装较为高层,另外,Delphi 的 IDE 太占资源、启动速度太慢、与某些显卡驱动程序冲突、VCL 中有 bug、调试器不够健壮、对不稳定的第三方控件没有防护措施等问题。

### 3.3.4 PowerBuilder

PowerBuilder 是 4GL 开发工具中的佼佼者,在多年的美国客户机/服务器应用开发工具市场上一直保持者领先地位。PowerBuilder 早期版本刚推出时,就引起了 IT 行业的关注。1995 年,PowerSoft 公司推出了一个具有里程碑意义的版本——PowerBuilder 5,一举确立了它在同类产品中的领先地位。

随着 Internet 以及万维网技术的迅速发展,WWW 与 Internet 上出现了无限的商机,传统的客户机/服务器已经不能满足客户对大量信息的需求,成为人们获取最新消息的瓶颈,因此人们更加关注多层应用实现模式。而 Web 技术通过提供快速的应用开发能力以及高效的用户访问能力,能够帮助企业建立起高效的分布环境。

在 SyBase 公司收购了 PowerBuilder 之后,便把重点放在了 Web 应用开发和分布式应用的开发上,形成了一套完整的 Internet 的开发平台——Enterprise Application Studio。

Enterprise Application Studio 是 SyBase 公司为 Internet 应用开发人员特别设计的一套企业应用开发和提交的开发工具包，可以实现包括 Web 应用、分布式应用和 Client/Server 应用的开发。

Enterprise Application Studio 将强大的 4GL RAD 开发工具 PowerBuilder 和获得众多奖项的 Java 开发工具 PowerJ 以及高性能提交环境 Enterprise Application Server 集成在一起。无论是开发各种复杂的 4GL 应用程序或者 Java 程序，还是建立多层应用中适应各种提交需要的中间层结构，Enterprise Application Studio 都可以满足用户的需求。

Enterprise Application Studio 包括以下三个产品：

(1) SyBase Enterprise Application Server。

(2) PowerBuilder。

(3) PowerJ。

### 1. Enterprise Application Server

Enterprise Application Server（EAServer）可以支持多种客户类型。客户端可以用 HTML、ActiveX、Java、JavaScript 工具进行开发或者是它们中的几种工具组合开发。它是为充分发挥基于组件和多层体系结构的应用特点而专门设计的，可以同时支持 C++、Java、ActiveX 和 CORBA，使用户可以利用多种计算环境获得灵活的开发能力。

Enterprise Application Server 集成了操作方便且功能强大的 Web 应用服务器 PowerDynamo 和高性能的组件事务处理服务器 Jaguar CTS。使用它可以建立以内容为导向的应用系统，前端通过浏览器展示数据，后端不仅可以提供动态页面而且具有强大的事务处理能力。企业可以完全在一个完整开发环境中实现基于 Web 的事务处理应用的一切功能。

使用 Power Dynamo，企业可以用 HTML 和 JavaScript 创建简单的客户应用，当被浏览器调用时，Power Dynamo 可以有效地处理包含展示逻辑和数据库连接信息的模板，快速组合 HTML 页面。

Jaguar CTS 是一个先进的组件式的事务处理服务器，它体现了多层的、分布式的企业计算环境和 Web 应用的优越性。Jaguar CTS 减少了分布式应用开发的复杂性，消除了开发者对线程、锁定、事务处理和内存管理的顾虑。开发者可以方便地通过点击访问由 Jaguar CTS 提供的企业级服务进行开发和实现业务逻辑。

PowerDynamo 和 Jaguar CTS 在 Enterprise Application Server 中的结合提供了一个全面的。灵活的分布式 Web 应用的基础，可以在一个集成的环境中提供从最简单的动态 Web 站点到复杂的、数据密集型的事务处理应用。

### 2. PowerJ

PowerJ 是业界领先的标准的企业分布式应用开发工具，是一个基于组件的、高效率开发环境，它支持各种 JDK 版本、支持 JavaBeans、ActiveX 和 CORBA 等组件标准，体现了真正的开放性。

PowerJ 包含了 PowerSite——用来建立、管理和提交 Web 应用的完整的 RAD 环境，增强了 Web 页面的开发能力、提交能力和版本控制能力。PowerSite 包括一系列的新向导

(Wizards)帮助你快速方便地建立 Web 应用,通过向导设置工作空间、设置项目以及保存开发和提交的 Web 应用文件,在项目增加文件、文件夹和应用特性。

PowerJ 包含了开发版的 SyBase Jconnect for JDBC,它是由纯 Java 写成,完全符合 JDBC 标准,可以使开发者在多层的、异购的环境中植入数据库访问功能,并且可以高速地访问 SyBase 数据库,并且通过 DirectConnect 和 OmniConnect 访问其他 25 种以上的数据源。Jconnect 可以为客户的 Java 应用提供占用资源很小的数据库连接方案。

PowerJ 可以用拖曳的编程方式进行 Java 应用开发,通过强大的参考卡和参数化的编程向导可以充分发挥出 Java 的优点。

PowerJ 对 JavaBeans、ActiveX 和 CORBA 组件具有很强的支持能力,在所有 JavaBeans 组件和包中都可以充分发挥出 Java 的优点。

PowerJ 的组件重用性可以方便地在客户端、服务器端和中间层服务器上创建、测试和提交新的 JavaBeans 组件,并且第三方的 ActiveX 和 JavaBeans 组件无缝地集成到 PowerJ 的环境中。

### 3. 全新的开发和提交

研制——在生成中间层逻辑的过程中,可以很方便地使用向导来加快开发人员的开发速度。通过组件生成向导轻易地在 PowerBuilder IDE 内实现 EAServer 组件的属性描述,如实例缓冲和事务处理属性,并且生成完整的功能组件。由向导自动生成的 To-do-list 可以引导开发人员成功地完成组件开发过程所需要的所有步骤。

测试——中间层逻辑的实时编辑功能可以让开发人员不需要脱离 PowerBuilder 开发环境而对更新了的代码进行编辑和测试。EAServer 可以自动参考更新对象代码,进行及时、有效的测试,而不需要重新提交组件。

调试——当运行分布式应用程序的时候,开发人员可以同时调用 PowerBuilder 调试程序,对客户端或者服务器端的代码进行单步跟踪,并能够进行远程调试。

提交——单击一次按钮,就可以对 PowerBuilder 组件进行打包和性能优化,然后自动地安装到网络中任意一个 EAServer 库或局域 MTS 服务器中。

### 4. 全新的数据库连接

1) 增强的数据库接口

提供了三个新的数据库接口并增强了一些现有的数据库接口,使你的应用可以访问更多的数据库。

(1) SyBase SYJ 数据库接口。提供比 PowerBuilder 开发的 Jaguar 组件与 ASE11.5 更强的连接能力。

(2) JDS 和 JDM JDBC 数据库接口。提供符合 JDBC API 标准的数据库连接能力。

(3) OLE DB 数据库接口。提供符合 OLE DB API 标准的数据库连接能力。OLE DB 提供访问所有数据类型的体系结构,包括 SQL 和非 SQL 的数据库,并具有关系型数据处理功能。

2) 输入和输出数据库预定义文件

每一个数据库接口都包含输入和输出数据库预定义文件的选项,使得预定义文件可以