“十五”国家重点电子出版物规划项目 计算机基础知识普及和软件开发系列

2002 精通热门软件工具丛书（2）

Inside XML

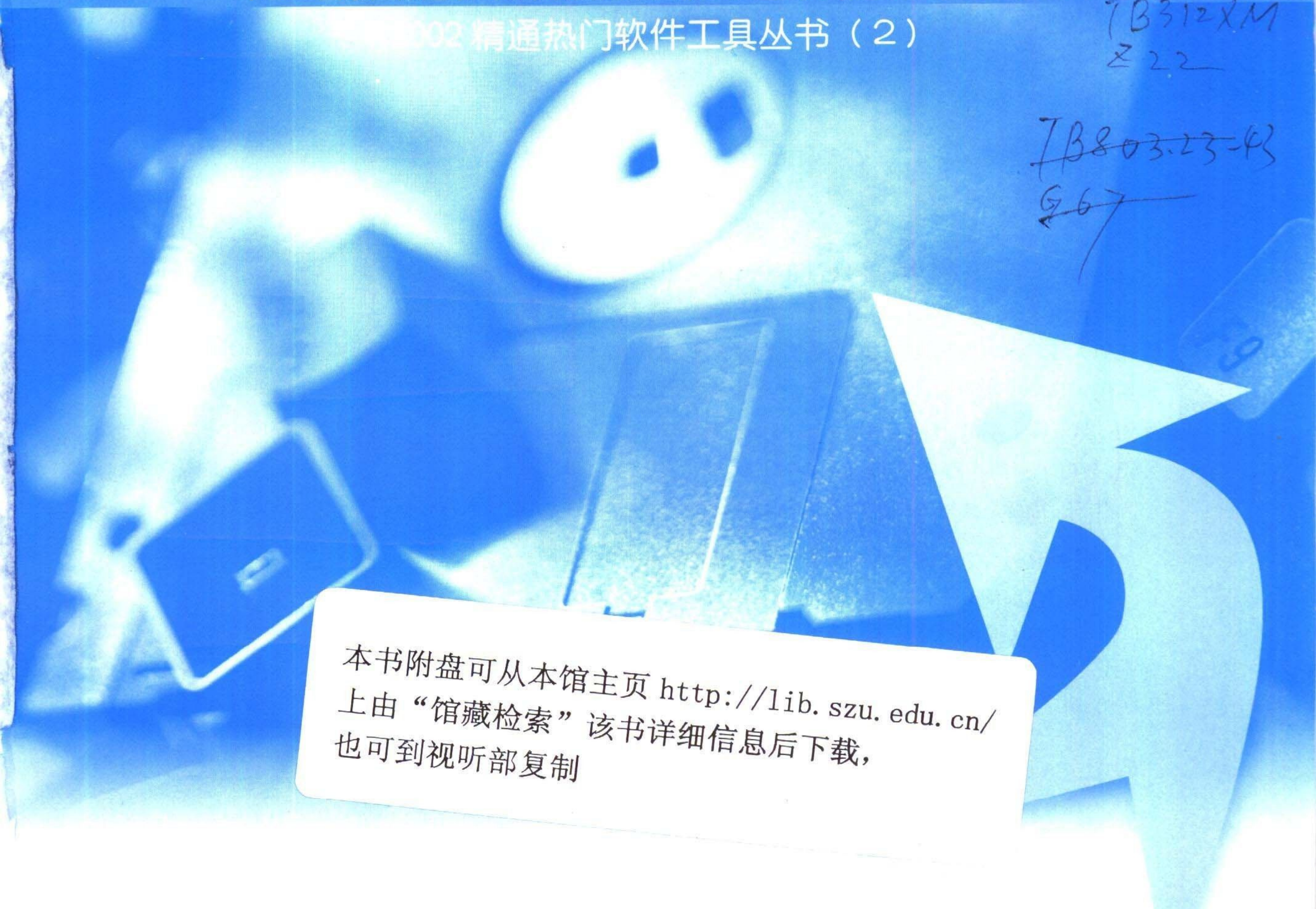
XML编程

从入门到精通

北京希望电子出版社　总策划
曾春平　王超　张鹏　编　写

中国科学出版集团
北京希望电子出版社

“十五”国家重点电子出版物规划项目 计算机基础知识普及和软件开发系列
802 精通热门软件工具丛书（2）



Inside XML

XML编程从入门到精通

北京希望电子出版社 总策划
曾春平 王超 张鹏 编 写

中国科学出版集团
北京希望电子出版社

内 容 简 介

这是一本通过 12 个典型范例介绍 XML 编程从入门到精通的专著。

全书由 3 篇，12 章组成，主要内容包括：第一篇　XML 提高，介绍数据建模、DTD 模式、XML 模式和 XML DR、名域机制。本篇附有 **3** 个实例；第二篇　XML 与数据，介绍 DOM 进阶、DOM 应用实例（包括投票系统、留言本、网址及短消息管理器）、SAX 进阶、XML 与数据。并提供了 **7** 个实例；第三篇　XML 工业应用，主要内容是 WML（无线应用协议）、SMIL（同步多媒体集成语言）、XML 与电子商务、XML 扩展（共 **2** 个实例）。

本书的特点是具体应用范例和软件功能相结合，边讲边练，由浅入深，学习轻松，上手容易。

本书面向初、中级读者，对高级读者也有重要参考价值。

本版 CD 为配套书

系 列 盘 书：“十五”国家重点电子出版物规划项目　计算机知识普及和软件开发系列
2002 精通热门软件工具丛书（2）

盘 书 名：Inside XML　XML 编程从入门到精通

文本著作者：曾春平　王超　张鹏

CD 制作者：希望多媒体开发中心

CD 测试者：希望多媒体测试部

责任编辑：周　艳

出版、发行者：北京希望电子出版社

地　　址：北京中关村大街 26 号，100080
网址: www.bhp.com.cn　E-mail: lwm@hope.com.cn
电话: 010-62562329, 62541992, 62637101, 62637102, 62633308, 62633309
（图书发行）　010-62613322-215（门市）　010-62547735（编辑部）

经　　销：各地新华书店、软件连锁店

排　　版：希望图书输出中心　周宇

CD 生产者：北京中新联光盘有限责任公司

文本印刷者：北京双青印刷厂

开本 / 规格：787 毫米×1092 毫米　16 开本　21.375 印张　498 千字

版次 / 印次：2002 年 2 月第 1 版　2002 年 2 月第 1 次印刷

印　　数：0001—5000 册

本 版 号：ISBN 7-900088-49-0

定　　价：35.00 元（本版 CD）

前　言

从 1998 年 2 月 W3C 正式推出 XML 后，在短短的三年间，XML 以惊人的速度在广大的设计人员中传播开来。随着 Internet 的飞速发展，HTML 开始对更多的网络设计要求显露出疲态。XML 也就是在这种大环境下孕育而生的。

一种技术的产生不可避免的会产生连带的效应，随着 XML 技术的成熟，XML 也越来越被人们重视。我个人也在研究 XML 的内容，尤其是比较深入的，象 XML 的接口、XML 通讯等内容。当我看了许多关于 XML 的书籍之后，我总感觉到好像欠缺了点什么。因为感觉总是每本书的内容差不多，不是从 XML 的语法讲起，就是从 XML 的历史开始，而内容中随处可见的总是一些浅层的内容，对于 XML 的一些内核和接口问题，往往什么都没有说或说的非常简略，这使我很失望。

事实上，我刚开始接触网络是从研究 ASP 和 JAVA 开始的，我想可能很多的程序员和我一样，这些技术在 HTML 网页上有很好的应用，我们总能找到很多可以实现的，技术。但是当我们涉及到 XML 的时候，我们有时候有点为难了，因为在大多数的书籍里，浅显的技术完全不需要我们已经掌握的技术。在这个方面，我们都需要更加完善的内容来介绍一些更深的内容，比如平台的互连，网络之间的通信等来发挥我们应有的能力。

对于绝大多数的 XML 入门者来说，XML 方面的可能只是一知半解，也就是停留在一个平台上，无法再提高了，这主要是国内现有的 XML 方面的书籍涉及到的内容面都比较狭窄，介绍的内容也都是很基本的一些内容，无法再更加深入；虽然有些好的国外书刊，但是往往也因为翻译不好而很难理解。读者可能有这样的感触，当自己花了很多的时间和金钱在书店购买了一大堆的 XML 的书籍，拿回家里，不得不从一本厚厚的书籍中寻觅出仅仅几十页的有用资源。这不能不说是一种浪费！因此，本书从开始就避开了一些浅显易懂的 XML 基础概念和 XML 基础语法。直接从 XML 的高级应用方面对这个很有前途的语言进行介绍。在这些介绍中，我们侧重在 SAX 接口和 DOM 接口的高级使用介绍中。对于 Windows 独行天下的今天，各个软件都充斥的接口、模块、对象等等诸如此类的东西。这种编程的趋势使得我们必须对新的语言的接口问题产生高度的重视。想要使用一个单独的语言完成一项工程是几乎不可能的，或者是非常繁琐的。使用接口，使得各个语言间有交流的通道。XML 在这一点上是成功的。

XML 的应用领域是极其广阔的，在本书的第 4 章之后，我们介绍了关于 XML 的在几个典型领域的应用，例如：电子商务、无线应用协议、多媒体 SMIL 等等。对一个设计者来说，一个语言最终的用处就是在实际的应用上，XML 的实际应用远远超出了本书中所介绍的这几方面。但是，可以通过本书的介绍对 XML 的应用有“质”的理解。对各个领域的应用的介绍是没有止境的，但是只有真正懂得了 XML 的应用的方法，才能在不断推出的 XML 扩展中，第一时间掌握它。

在写这本书的时候，我们假设了读者是有一定的 XML 的编程经验的设计人员，因此，一些 XML 的基础语法和 XML 的一些简单的概念将不会在本书中被解释。如果读者在阅读时感到有理解困难，请自行参照一些关于 XML 的基础读物。

本书着重于将 XML 作为一种开放技术的应用工具来介绍。提供了 XML 模式、数据

交换、可视化风格等技术的研究。本书面向有一定基础的 XML 设计人员。其内容大多数涉及的是一些XML应用方面的内容，比较技术性和专业性。本书的重点放在XML在DOM、SAX 以及数据库方面的应用和 XML 的一些实际扩展。尤其是 XML 的扩展，本书介绍了最新的WML语言和电子商务包括象BIZTALK方面的研究。相信读完本书后读者对于XML的各种专业技术会有一个比较全面的了解。

本书由曾春平、王超和张鹏执笔编写。此外，张东、李晓、范智育、王宏生、李光龙、王瑾、吴浩、李炎、刘伟、刘华刚、朱峰、赵晓燕、李晓、马苍、郝春容、韦勇、成美华、萧峰、李菊、张浩然、李欣、张浩、李想、朱大成、周卫、赵中伟、杨竞锐、王贵新、张诚华、朱丽云、程松、李毅、赵郡超、孙名等同志在整理材料方面给予了作者很大的帮助。

由于时间仓促，加之编者的水平有限，缺点和错误在所难免，恳请专家和广大读者不吝赐教，批评指正。

编者

目　录

第一篇 XML 提高

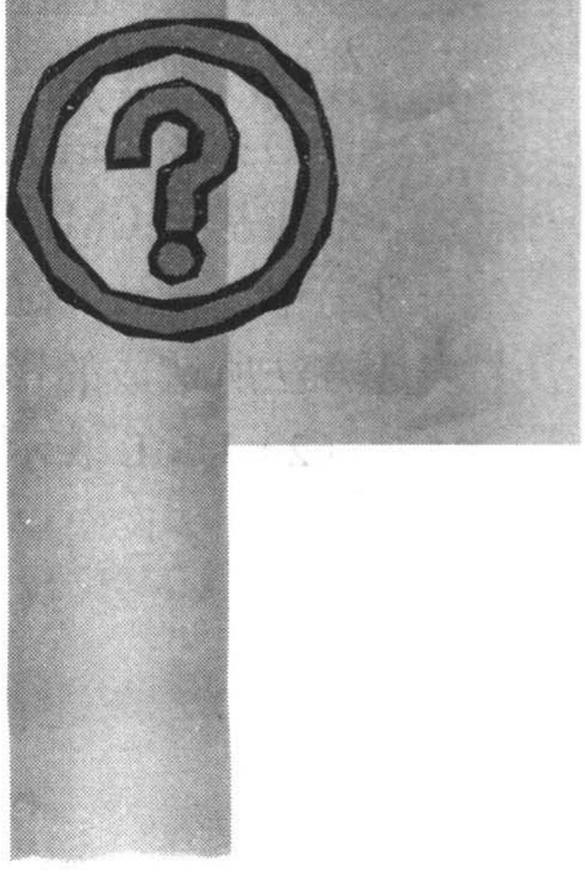
- 第 1 章　数据建模
- 第 2 章　DTD 模式
- 第 3 章　XML 模式和 XML DR
- 第 4 章　名域机制

作为本书的第一篇，该篇内容是比较简单的。

首先第 1 章介绍了一些基本的信息建模原则。在介绍信息建模的时候，对静态模型和动态模型也分别作了必要的介绍。这两种模型都与 XML 的设计有关，而静态永久性数据和暂时性消息这两种类型也是很有意义的。

第 2 章和第 3 章对 XML 的模式做了一个深入的探讨，包括 DTD 和 XML Schema 机制，以及 XML DR 等相关知识。通过该章内容的学习，读者可以选择适合自己的验证方式，也可以对这几种方式进行不同程度的改进，总之力图满足使用的需求。

第 4 章则深入介绍了名域机制。可以毫不夸张的说，正是由于名域机制的引入，XML 才得到了广泛的应用。



第 1 章　数据建模

本章导读

任何文档的设计都需要考虑两点：一是要适合现在的需要；二是要能灵活扩展，适应未来的需要。XML 文档也不例外，因此需要一种标准的方法来设计 XML 文档，这就是本章要讨论的内容。本章将主要介绍 XML 文档设计时需考虑的若干因素，它们包括：

- ✉ 数据建模
- ✉ UML 方法
- ✉ 信息建模
- ✉ 设计 XML 文档

1.1　数据建模

1.1.1　UML 方法

UML（Unified Modeling Language）是一种现今最流行的建模方法。本章的实例将会依照 UML 方法建立，下面先来介绍 UML 方法。

UML 是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它不仅支持面向对象的分析与设计，还支持从需求分析开始的软件开发的全过程。

由于这方面优势，1997 年 11 月 17 日，OMG 采纳了 UML 1.1 作为基于面向对象技术的标准建模语言。UML 代表了面向对象方法的软件开发技术的发展方向，具有巨大的市场前景，也具有重大的经济价值和国防价值。

标准建模语言 UML 的内容

首先，UML 融合了 Booch、OMT 和 OOSE 方法中的基本概念，而且这些基本概念与其他面向对象技术中的基本概念基本一致，所以 UML 必然成为商业界乐于采用的一种简单且一致的建模语言；其次，UML 不仅仅是上述方法的简单汇合，还是在这些方法的基础上广泛征求意见，集众家之长，几经修改而完成的，所以 UML 扩展了现有方法的应用范围;第三，UML 是标准的建模语言，而不是标准的开发过程。尽管 UML 的应用以系统的开发过程为背景，但由于不同的组织和不同的应用领域，所以开发过程也不尽相同。

作为一种建模语言，UML 的定义包括 UML 语义和 UML 表示法两个部分。

（1）UML 语义

描述基于 UML 的精确元模型定义。元模型为 UML 的所有元素在语法和语义上提供了一种简单、一致、通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的表达方法所造成的影响。此外 UML 还支持对元模型的扩展定义。

（2）UML 表示法

UML 符号表示法，为开发者或开发工具使用图形符号和文本语法来进行系统建模提供了标准。这些图形符号和文字表达的是应用级的模型，在语义上它是 UML 元模型的实例。

标准建模语言 UML 的重要内容可以由下列五类图（共 9 种图形）来定义：

• 第一类是用例图，从用户角度描述系统功能，并指出各功能的操作者。

• 第二类是静态图（Static diagram），包括类图、对象图和包图。其中类图描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系如关联、依赖、聚合等，还包括类的内部结构（类的属性和操作）。类图描述的是一种静态关系，在系统的整个生命周期都是有效的。对象图是类图的实例，几乎使用与类图完全相同的标识。他们的不同点在于对象图显示类的多个对象实例，而不是实际的类。一个对象图是类图的一个实例。由于对象存在生命周期，因此对象图只能在系统某一时间段存在。包由包或类组成，表示包与包之间的关系。包图用于描述系统的分层结构。

• 第三类是行为图（Behavior diagram），描述系统的动态模型和组成对象间的交互关系。其中状态图描述类的对象所有可能的状态以及事件发生时状态的转移条件。通常，状态图是对类图的补充。在实用上并不需要为所有的类画状态图，仅为那些有多个状态，其行为受外界环境的影响并且发生改变的类画状态图。而活动图描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并行活动。

• 第四类是交互图（Interactive diagram），描述对象间的交互关系。其中顺序图显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互;合作图描述对象间的协作关系，合作图跟顺序图相似，显示对象间的动态合作关系。除显示信息交换外，合作图还显示对象以及它们之间的关系。如果强调时间和顺序，则使用顺序图;如果强调上下级关系，则选择合作图。这两种图合称为交互图。

• 第五类是实现图（Implementation diagram）。其中构件图描述代码部件的物理结构及各部件之间的依赖关系。一个部件可能是一个资源代码部件、一个二进制部件或一个可执行部件。它包含逻辑类或实现类的有关信息。部件图有助于分析和理解部件之间的相互影响程度。

配置图定义了系统中软硬件的物理体系结构。它可以显示实际的计算机和设备（用节点表示）以及它们之间的连接关系，也可显示连接的类型及部件之间的依赖性。在节点内部，放置可执行部件和对象以显示节点与可执行软件单元的对应关系。

建模语言的应用

从应用的角度看,当采用面向对象技术设计系统时,首先需要描述需求;其次根据需求建立系统的静态模型,以构造系统的结构;第三步是描述系统的行为。其中在第一步与第二步中所建立的模型都是静态的,包括用例图、类图(包含包)、对象图、组件图和配置图等五个图形,是标准建模语言 UML 的静态建模机制。而第三步中所建立的模型是可以执行的,或者表示执行时的时序状态或交互关系。它包括状态图、活动图、顺序图和合作图等四个图形,是标准建模语言 UML 的动态建模机制。因此,标准建模语言 UML 的主要内容也可以归纳为静态建模机制和动态建模机制两大类。

下面总结一下标准建模语言 UML 的主要特点:

✉ UML 统一了 Booch、OMT 和 OOSE 等方法中的基本概念。

- ✉ UML 吸取了面向对象技术领域中其他流派的长处,其中也包括非 OO 方法的优点。UML 符号表示考虑了各种方法的图形表示，删除了大量易引起混乱的、多余的和极少使用的符号,也添加了一些新符号。因此,在 UML 中融入了面向对象领域的很多思想。这些思想并不是 UML 的开发者们发明的,而是他们依据最优秀的 OO 方法和丰富的计算机科学实践经验综合提炼而成的。
- ✉ UML 在演变过程中还提出了一些新的概念。在 UML 标准中新添加了模板(Stereotypes)，职责(Responsibilities)，扩展机制(Extensibility mechanisms)，线程(Threads)，过程(Processes)，分布式(Distribution)，并发(Concurrency)，模式(Patterns)，合作(Collaborations)，活动图(Activity diagram)等新概念,并清晰地区分类型(Type)、类(Class)和实例(Instance)，细化(Refinement)，接口(Interfaces)和组件(Components)等概念。

UML 的目标是以面向对象图的方式来描述任何类型的系统，有广泛的应用领域。其中最常用的是建立软件系统的模型，还可以用于描述非软件领域的系统（如机械系统、企业机构或业务过程），也可以处理包含复杂数据的信息系统、具有实时要求的工业系统或工业过程等。总之，UML 是一种通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行建模。

此外，UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。在需求分析阶段，可以用用例来捕获用户需求。通过用例建模，可以描述对系统感兴趣的外部角色及其对系统（用例）的功能要求。分析阶段主要考虑问题领域中的主要概念（如抽象、类和对象等）和机制，需要识别这些类以及它们相互间的关系，并用 UML 类图来描述。

为实现用例，类之间需要协作，这可以用 UML 动态模型来描述。在分析阶段，只对问题领域的对象（现实世界的概念）建模，而不考虑定义软件系统中技术细节的类（如处理用户接口、数据库、通讯和并行性等问题的类）。这些技术细节将在设计阶段引入，因此设计阶段为构造阶段提供了更详细的规格说明。

编程（构造）是一个独立的阶段，其任务是用面向对象编程语言将设计阶段描述的类转换成实际的代码。需要指出的是，在用 UML 建立分析和设计模型时，应尽量避免考虑把模型转换成某种特定的编程语言。因为在早期，模型仅仅是理解和分析系统结构的工具，过早考虑编码问题十分不利于建立简单正确的模型。

UML 模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的 UML 图作为测试依据：单元测试使用类图和类规格说明；集成测试使用部件图和合作图；系统测试使用例图来验证系统的行为；验收测试则由用户进行，以验证系统测试的结果是否满足分析阶段所确定的需求。

总之，标准建模语言 UML 适用于以面向对象技术来描述的任何类型的系统，而且适用于从需求规格描述直至系统完成后的测试和维护整个系统开发的不同阶段。

1.1.2 动态模型和静态模型

静态模型侧重于描述系统的状态。比如下面的语句：“一个客户有一个或多个账号”，“一本书有一个书名”，“每张音乐 CD 都有一个出厂公司”。这些语句主要是描述系统中对

象的类型、特性以及对象之间的关系。静态模型也可以定义词汇表，这些名称往往是比较简洁明了的，能得到大家的公认。

动态模型侧重于描述对信息的处理，例如：处理模型和工作流图表、数据流模型、以及对象生存周期历史。如下面的类型语句，“销售部门把产品报告发送给负责为顾客提供咨询的顾问”，“财务部提取公司某一部门的月盈余总结”。动态模型描述了信息的交换，即出于特定的目的将数据从一个地方发送到另一个地方。

一般的，静态模型与数据库的设计直接相关，信息被长期保存，并用于多种用途；而动态模型直接与信息的设计相关，信息存在的时间很短，而且用途常常随时间转移。但是在实际的设计中，我们并非单纯的把静态模型和动态模型分开来，而是同时考虑这两种模型。

静态信息模型

通常使用用例来建立静态模型。用例模型描述的是外部执行者（Actor）所理解的系统功能。用例模型用于需求分析阶段，它的建立是系统开发者和用户反复讨论的结果，表明了开发者和用户对需求规格达成的共识。首先，它描述了待开发系统的功能需求；其次，它将系统看作黑盒，从外部执行者的角度来理解系统；第三，它驱动了需求分析之后各阶段的开发工作，不仅在开发过程中保证了系统所有功能的实现，而且被用于验证和检测所开发的系统。

几乎在任何情况下都会使用用例来获取需求、规划和控制项目等信息。用例的获取是需求分析阶段的主要任务之一，而且是首先要做的工作。大部分用例将在项目的需求分析阶段产生，并且随着工作的深入会发现更多的用例，这些都应及时增添到已有的用例集中。用例集中的每个用例都是一个潜在的需求。

1．获取执行者

获取用例首先要找出系统的执行者。可以通过让用户回答一些问题来识别执行者。以下问题可供参考：

- 谁使用系统的主要功能（主要使用者）。
- 谁需要系统支持他们的日常工作。
- 谁来维护、管理，使系统正常工作（辅助使用者）。
- 系统需要操纵哪些硬件。
- 系统需要与其它哪些系统交互，包括其它计算机系统和其它应用程序。
- 对系统产生的结果感兴趣的人或事物。

2．获取用例

一旦获取的执行者，就可以对每个执行者提出问题以获取用例。以下问题可供参考：

- 执行者要求系统提供哪些功能（执行者需要做什么）。
- 执行者需要读、产生、删除、修改或存储的信息有哪些类型。
- 必须提醒执行者注意的系统事件有哪些，或者执行者必须提醒系统的事件有哪些。怎样把这些事件表示成用例中的功能。
- 为了完整地描述用例，还需要知道执行者的某些典型功能能否被系统自动实现？

还有一些不针对具体执行者的问题（即针对整个系统的问题）：

• 系统需要何种输入输出？输入从何处来？输出到何处？

• 当前运行系统（也许是一些手工操作而不是计算机系统）的主要问题？

需要注意,最后两个问题并不是指没有执行者也可以有用例,只是获取用例时尚不知道执行者是什么。一个用例必须至少与一个执行者关联。还需要注意:不同的设计者对用例的利用程度也不同。例如，Ivar Jacobson 说，对一个十来人的项目,他需要二十个用例。而在一个相同规模的项目中，Martin Fowler 则用了一百多个用例。我们认为:任何合适的用例都可使用,确定用例的过程是对获取的用例进行提炼和归纳的过程,对一个十来人的项目来说,二十个用例似乎太少，一百多个用例则嫌太多,需要保持二者间的相对均衡。

动态建模机制

动态建模机制主要包括下面几个概念，先介绍一下。

1．消息

在面向对象技术中，对象间的交互是通过对象间消息的传递来完成的。在 UML 的四个动态模型中均用到消息这个概念。通常，当一个对象在调用另一个对象中时，即完成了一次消息传递。当操作执行后，控制便返回到调用者。对象通过相互间的通信（消息传递）进行合作，并在其生命周期中根据通信的结果不断改变自身的状态。

UML 定义的消息类型有三种:

- ✉ 简单消息(Simple Message) 表示简单的控制流。用于描述控制如何在对象间进行传递,而不考虑通信的细节。
- ✉ 同步消息(Synchronous Message) 表示嵌套的控制流。操作的调用是一种典型的同步消息。调用者发出消息后必须等待消息返回,只有当处理消息的操作执行完毕后,调用者才可继续执行自己的操作。
- ✉ 异步消息(Asynchronous Message) 表示异步控制流。当调用者发出消息后不用等待消息的返回即可继续执行自己的操作。异步消息主要用于描述实时系统中的并发行为。

2．状态图

状态图(State Diagram)用来描述一个特定对象的所有可能状态及其引起状态转移的事件。大多数面向对象技术都用状态图表示单个对象在其生命周期中的行为。一个状态图包括一系列的状态以及状态之间的转移。

3．顺序图

顺序图(Sequence Diagram)用来描述对象之间动态的交互关系,着重体现对象间消息传递的时间顺序。顺序图存在两个轴:水平轴表示不同的对象,垂直轴表示时间。顺序图中的对象用一个带有垂直虚线的矩形框表示，并且标有对象的名称和类名。垂直虚线是对象的生命线,用于表示在某段时间内对象是存在的。对象间的通信通过在对象的生命线间画消息来表示。消息的箭头指明消息的类型。

顺序图中的消息可以是信号(Signal)、操作调用或类似于 C++中的 RPC(RemoteProcedure Calls)和 Java 中的 RMI(Remote Method Invocation)。当收到消息时,接收对象立即开始执行活动,即对象被激活了。通过在对象生命线上显示一个细长矩形框来表示激活。

消息可以用消息名及参数来标识。消息也可带有顺序号,但较少使用。消息还可带有条

件表达式，表示分支或决定是否发送消息。如果用于表示分支，则每个分支是相互排斥的，即在某一时刻仅可发送分支中的一个消息。

4．合作图

合作图(Collaboration Diagram)用于描述相互合作的对象间的交互关系和链接关系。虽然顺序图和合作图都用来描述对象间的交互关系,但侧重点不一样。顺序图着重体现交互的时间顺序,合作图则着重体现交互对象间的静态链接关系。

合作图中对象的外观与顺序图中的一样。如果一个对象在消息的交互中被创建,则可在对象名称之后标以{new}。类似地,如果一个对象在交互期间被删除,则可在对象名称之后标以{destroy}。对象间的链接关系类似于类图中的联系(但无多重性标志)。通过在对象间的链接上标志带有消息串的消息(简单、异步或同步消息)来表达对象间的消息传递。

5．活动图(Activity Diagram)

活动图的应用非常广泛,它既可用来描述操作(类的方法)的行为,也可以描述用例和对象内部的工作过程。活动图是由状态图变化而来的,它们各自用于不同的目的。活动图依据对象状态的变化来捕获动作(将要执行的工作或活动)与动作的结果。活动图中一个活动结束后将立即进入下一个活动(在状态图中状态的变迁可能需要事件的触发)。

6．四种图的运用

上面对 UML 中用于描述系统动态行为的四个图(状态图、顺序图、合作图和活动图)做了简单的介绍。这四个图都可用于系统的动态建模，但它们各自的侧重点有所不同，分别用于不同的目的。下面总结一下如何正确使用这几个图，在实际的建模过程中要根据具体情况灵活运用这些建议。

首先，不要对系统中的每个类都画状态图。尽管这样做很完美,但太浪费精力,而且可能只关心某些类的行为。正确的做法是：为帮助理解类而画它的状态图。状态图描述跨越多个用例的单个对象的行为,而不适合描述多个对象间的行为合作。为此，常将状态图与其它技术(如顺序图、合作图和活动图)组合使用。

顺序图和合作图适合描述单个用例中几个对象的行为。其中顺序图突出对象间交互的顺序,而合作图的布局方法能更清楚地表示出对象之间静态的连接关系。当行为较为简单时,顺序图和合作图是最好的选择。但当行为比较复杂时,这两个图将失去其清晰度。因此,如果想显示跨越多用例或多线程的复杂行为,可考虑使用活动图。另外,顺序图和合作图仅适合描述对象之间的合作关系,而不适合对行为进行精确定义,如果想描述跨越多个用例的单个对象的行为,应当使用状态图。

1.2　信息建模

1.2.1　静态建模

下面开始建立模型了，事实上面对一大堆的数据，也许不知道该从何处下手。这就需要对一些问题进行透彻的了解，这些问题包括：存在哪些信息；他们为什么存在；商业目标是如何实现的；是否有更富创造性和想象力的方法等。这些问题对你的建模是非常有用的。理解了这些问题之后，就可以开始真正的建模了。开始建模的最好方法就是为系统中

的事物设置名称。不必顾虑第一步如何开始，只要从自己所处的位置开始就可以了。

命名事物

事物通常是指那些实体、对象、类或数据元素，我们将这些命名的事物称为对象类型。因此可以把与系统相关的所有事物列出来。下面以一个营业性质的网上音乐书店来说明静态建模和动态建模的建立，这些事物包括客户、预订、仓库、出品公司等。当然这个实例本身是比较简单的，对于比较复杂的信息模型，比如大型商业系统、化学系统等，就需要对系统进行文本描述，然后从中选出所有名词。

得到这些名词后，下一步的工作就是产生对象类型的定义。这可能比较麻烦，通常你经常会为某个名词的确定而思索很长的时间，比如说对于一个网上的预订，每个人对于“预订”的现实意义的理解是不尽相同的。看到预订两字，有人也许认为这是简单的电话预约，还有人认为是因为货还没到而作的预订，这些理解的偏差往往会使人们的交流产生误解。对于单个的网上预订，究竟怎么样才算是一个真正的预订呢？譬如一个人订购一件物品算是预订的话，那么一个人订购两件物品算是一个预订还是两个？同一件商品被预订了多次算是一个预订还是多次预订。预订的客户调整了预订对象仍算是一种预订吗？

事实上这些问题都需要在建立词汇表的时候明确的定义，否则就会引起麻烦，因此就一个网上音乐店来说，一个预订的含义就应该讲明白，即一个客户订购一件音乐商品。

经过以上的对象类型定义，就得到了一长串的对象类型列表，也许可能包含了一些较长的名字，这并没关系，我们所需要努力达到的目标是如何对事物进行明确的标记，以此来使业界人士正确的理解和解释。而对于如何组织各种标记包括标记的包含关系，将在下面的几步中讨论。

分类

经过上面的步骤，我们已经列出了系统中所有的对象类型，并且给它们命名了，接下来就要为这些对象类型分类。分类是一个必要的过程，我们必须要明确哪些对象应该是超类，它们的子类包含哪些内容，它们又是如何存在的，子类中应该包含哪些信息。比如音乐又可以分为三类，即 CD、MP3 和 TYPE，而从另一个角度来说，音乐也可以按照歌手来分类以方便客户进行查询。对于客户来说，也应该分为个人客户和公司客户。按照刚才所讲的例子，可以把各种分类用下表来表示：

表 1-1　各种分类信息

超类	子类
音乐	CD
	MP3
	TYPE（磁带）
客户	公司客户
	个人客户
歌手	大陆歌手
	港台歌手
	国外歌手

以上列出了本系统中的部分对象的分类，通常我们称上述方法为标志子类，标志子类非常有用，但更重要的事，它有助于理解对象的类型定义。通过子类的包含通常可以很明确的理解超类的真正含义而不致引起错误和混乱。

在 UML 方法中，常用图表来表示子类与超类之间的关系，利用箭头从子类指向超类表示它们之间的关系。

寻找关系

有了分类的对象类型定义，接下来做的就是确定对象类型之间的关系。下面举几个例子来说明这些关系，比如说一个客户可以预订一个或多个音乐 CD，每个预订可以包含一张或多张相同的音乐 CD，每张出品的音乐专辑是对应于一个出品公司的，每张音乐专辑的进货渠道可以有一个或多个等。

这种关系在 UML 中称之为关联，可以用 UML 表示法的示意图来表示如下：

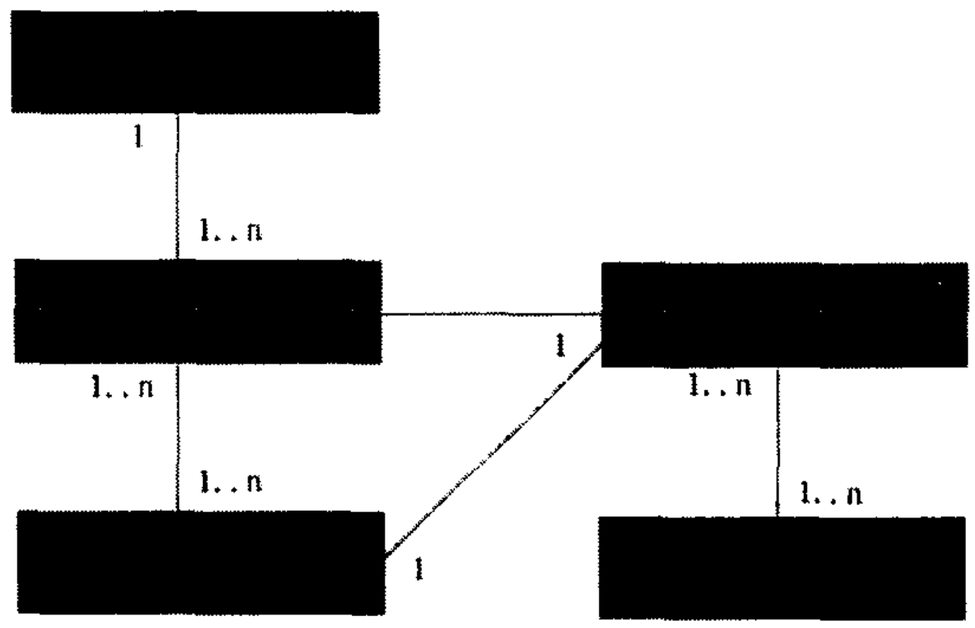


图 1-1　关联关系示意图

通常关系包含好多种，学过数据库或软件工程的读者应该很容易理解。在几类关系中，一对多关系在 XML 中最普遍：一章可以有多个段落，一个人可以预订多个房间，一本书可以有很多种版本等等。上图中我们将一端记为一个对象，另一端记为 1 到 n 个对象（1 意味至少有一个，至多有一个；1..n 表示至少一个，至多为 n 个）。有时候我们也会使用 0 到 n 个来代替 1..n 的关系。

另外还有多对多关系，例如一个读者可以读很多本书，同时一本书可以有很多个读者。在上面的示意图中也有这样的关系：一个预订可以包含很多张专辑，同时一张专辑可以承担好多个预订。对于多对多关系，通常我们要将每一对作为一个对象来命名：我们将有一个预订和一张音乐专辑组成的对称为一个有效的定购。

一对一关系并不常见，它标志一种唯一的关系，比如某个事物对应某个学名。但是在上面的图中也有这样的关系，即音乐专辑和库存的关系。我们之所以把它们的关系定位一对一的关系，是因为我们销售的音乐专辑一定要在库存中找得到，同时每张库存中的音乐专辑由一一对应于在网站上公布的音乐专辑。

事实上在 UML 表示信息的过程中，有一种关系是最重要的，前面已经提到过，那就是包含关系，它总是一对多或一对一的。比如客户包含地址、收款单位（人）等。音乐专辑包含价格、种类、编号等等。UML 定义了两种形式的包含：聚合和构成。我们把系统中