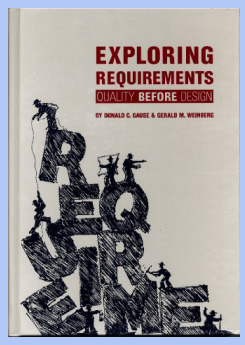
第 34 期（2004 年 2 月）



探索需求

X 非程序员
X-Programmer
软 件 以 用 为 本

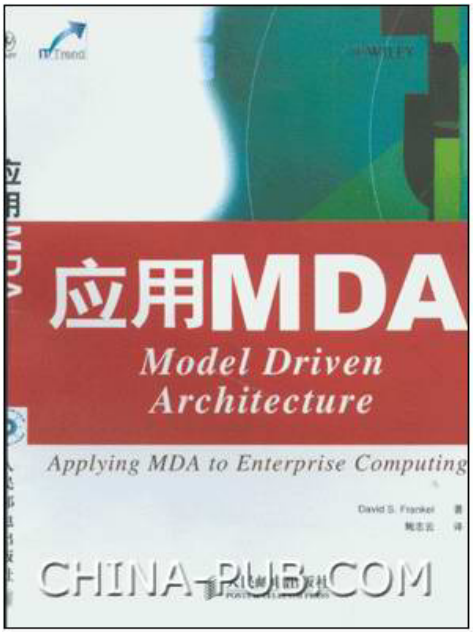
投稿：editor@umlchina.com
反馈：think@umlchina.com

http://www.umlchina.com/

David S. Frankel“应用 MDA”讲座举行

[2004/2/23]

2004 年 2 月 23 日（周一）上午 10:00-12:00，数十名 UMLChina 听众聆听了 David S. Frankel 的“应用 MDA”讲座。



David S. Frankel，构造大规模企业级系统的专家，OMG 架构委员会成员，是 MDA 的主要发起者之一。他的《应用 MDA》是国内引进的第一本 MDA 书籍。

类似的讲座将持续举行，感兴趣者请留意 UMLChina 首页的公告。

OMG Anaheim 会议消息：四个新的小组成立；架构委员会选举

[2004/2/17]

NEEDHAM, Mass.--国际对象管理集团(OMG(TM)) 成员 12 月 2 日到 5 日在 Anaheim, CA 召开会议，会议由 Compuware 公司赞助(www.compuware.com)。

在金融支付网关标准化方面的努力

包括主要的银行和金融协会的众多 OMG 成员组成了一个工作组，致力于标准化支付和业务系统之间的互操作和接口的模型。新的标准将在银行和金融的各个领域推广电子支付。所有有兴趣的公司都可以加入 OMG，共同指定这些新标准，有关信息可以垂询 info@omg.org。

新的小组

在 Anaheim，随着组织关注领域的扩大，OMG 成立了四个新的小组。新的架构驱动现代化（Architecture-Driven Modernization，ADM）平台小组(Platform Task Force，PTF)将通过定义知识发现（knowledge-discovery）的元模型，源和目标平台的模型，以及这些模型之间的转换来标准化现有应用。开发的协作服务规约专门兴趣小组（The Open Collaboration Services Initiative Special Interest Group）将基于 OMG 的 MDA(TM)开发企业间和企业内部的协作标准。信息和安全保证平台 SIG（特别兴趣小组）将调整当前和未来规约中的安全架构。最后，基于软件的交流（the Software Based Communications）领域小组（Domain Task Force，DTF）将标准化在军事、电信和其它领域实用的软件无线电的架构。

架构委员会选举

在 OMG 享有声誉的架构委员会负责审批每一个新的标准，以确保这些新标准符合 OMG 的架构并且和已有的标准保持一致。在 Anaheim，通过成员的选举，2AB 的 Carol Burt、88Solutions 的 Manfred Koethe、Raytheon 的 Robert Kukura、Hewlett-Packard 的 Jishnu Mukerji、Fujitsu 的 Thomas Rutt 被选为委员，任职二年。

新的底层架构（Infrastructure）标准

OMG 发布了三个新标准的 RFP（Requests for Proposals），其中介绍了要求以及日程表。其中两个是实时、嵌入和特殊系统 PTF 方面的：一个是基于 OMG 的模型驱动架构 MDA 的能源保存（power conservation）标准，其中定义了一个平台无关的能源保存系统。RFP 中要求这个标准必须包括为基于 CORBA 的系统提供的一致接口，另外，还必须要有为其它系统提供的接口。一个是通过可保证的高可靠（high-assurance）的实时操作系统涵盖

CORBA 应用的高可靠的对象请求代理标准。来自电信行业的第三个 RFP 则将提供一个应用于基于语音的交互对话的建模的符合 MOF 的元模型以及相应的 UML（TM）profile。这将保证开发人员可以在不用考虑硬件和运行的软件平台的情况下定义对话，使得这些应用在更先进的平台上可以得到重用。所有公司都可以加入 OMG 并提交这些 RFP，相关信息请垂询 info@omg.org。

新标准投票

成员们结束了对两个新标准的评估和最后是否采用的投票：其中一个定义了在生命科学研究、在多个层次上推进生物工艺学的各种应用之间的交互性的一系列量化标识符。另一个标准则定义了为基于 MOF 的 UML 模型定义的基于 CORBA 的仓库，对 MOF 和 OMG IDL 的绑定作了标准化的工作。

教程、信息日、赞助商演讲以及软件演示

周二，12 月 5 日，嵌入式实时系统模型集成计算的信息日上包括了 5 个演讲。会议赞助商 Compuware 公司的 Wim Bast 和 Bruce Epstein 详细介绍了“MDA 及 OptimalJ”，OMG 作了“OMG 规约总览”的教程。八个成员公司演示了他们基于 OMG 的各种规约所做的一些实现。

OMG 委员会签署新标准

在成员评估和认可之后，由 OMG 委员会投票通过，宣布了一个新的官方标准。在 Anaheim 会议接下来的工作中，委员会批准了应用于资源受限的嵌入式 CORBA 系统的一系列轻量级的服务-命名/目录、事件和时间。

下一次会议，点击链接察看更多：

下一次会议将于 2004 年 4 月 26～30 在美国的 St. Louis, MO 召开。非会员可以作为观察员出席。通过 www.omg.org/news/meetings/tc/guest.htm 可以得到邀请；OMG 主页是 www.omg.org。关于 MDA 的信息都在 www.omg.org/mda。所有 OMG 规约都可以从 www.omg.org/technology/documents/specifications.htm 免费下载得到。

（自 businesswire，袁峰 摘译，不得转载用于商业用途）

china-pub2003 总结：UMLChina 参与图书继续火爆

[2004/2/13]

继 2002 年《人月神话》之后，过去的 2003 年，UMLChina 参与的中译本图书继续火爆。根据 china-pub 的统计，2003 年总销量排行是：

1 《最后期限》 清华大学出版社（UMLChina 熊节、马姗姗 译）

2 《Java 与模式》 电子工业出版社

3 《人件（第 2 版）》 清华大学出版社（UMLChina 方春旭、叶向群 译）

4 《人月神话》 清华大学出版社 （UMLChina 汪颖 译）

5 《Java 编程思想（第 2 版）》 机械工业出版社

6 《Effective Java 中文版》 机械工业出版社

7 《重构：改善既有代码的设计（中文版）》 中国电力出版社

8 《敏捷软件开发：原则、模式与实践》 清华大学出版社

9 《设计模式：可复用面向对象软件的基础》 机械工业出版社

10 《Borland 传奇》 电子工业出版社

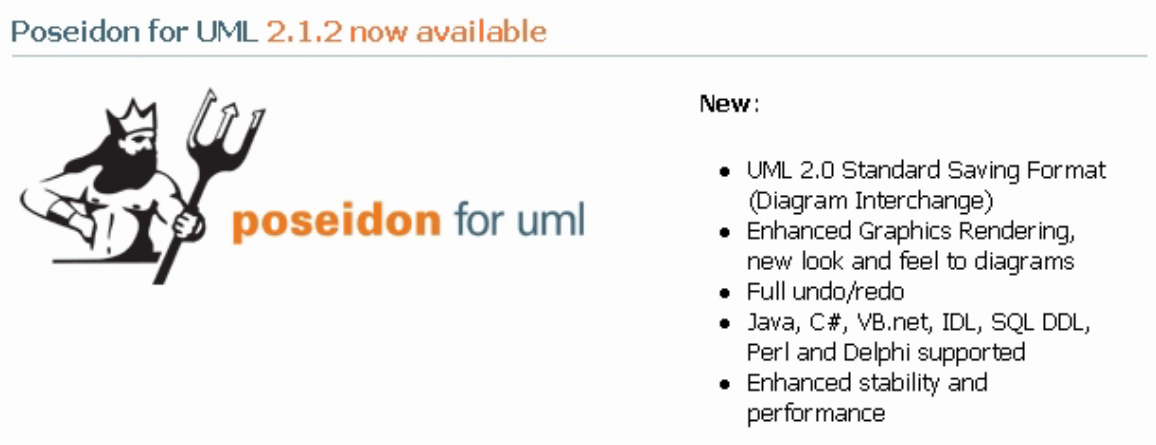
UMLChina 的下一本参与书籍，Martin Fowler 的《PEAA》即将出版。

Gentleware 在 OOP 展会上展示其首个集成 UML2.0 的工具

[2004/2/10]

Poseidon for UML 企业版使得较大开发团队的实时项目协作成为可能。

在慕尼黑的 OOP 展览会上，UML CASE 工具的国际性供应商 Gentleware AG 展出了其最新的建模工具 Poseidon for UML 企业版的 Beta 版。这是其第一个支持项目团队开发的工具，同时也是第一次集成了 UML2.0 的新标准。而除了 Gentleware 之外，其它的 UML 工具厂商还在犹疑是否要支持以及何时集成 UML2.0。在此次 OOP 展会上的技术讲座上，Gentleware 对 UML 2.0 的支持受到了积极的评价。



在技术讲座休息时，对 UML2.0 以及对这个 Poseidon 企业版充满兴趣的观众挤满了 Gentleware 的展位。展位上有一个演示的 Poseidon 企业版的演示版，开发人员和 IT 企业的领导们在这里看到了新版本的各种功能。通过两个网络工作站，新版本在团队开发方面的功能给参观者留下了深刻的印象。UML 专家以及 OOP 顾问 Mario Jeckle 教授出现在 Gentleware 展位，并回答了关于 UML2.0 以及他的关于新标准的讲座的各种问题，并现场为其新书“UML 2 glasklar”签名售书。

Gentleware AG 的执行官 Marko Boger 博士认为“ 来自产业的参观者对我们新开发的团队功能非常感兴趣，这也证明我们的工具有助于解决较大的开发环境中的问题，在这一点，我们已经比诸如 IBM Rational 和 Borland 这些竞争对手领先了不少”。企业版第一次使得多个开发人员同时在一个共同的项目中做改动成为可能，这样，物理上分散的开发团队就可以参与一个项目的开发，而不需要等待他们的同事完成子项目的工作。来自航空、汽车和金融行业的业务代表们已经表示了对使用 Gentleware AG 的新的 UML 工具的初步兴趣。

（自 PRNewswire，袁峰 摘译，不得转载用于商业用途）

Compuware 引导 MDA 主流

[2004/2/9]

模型驱动的基于模式的（Model-Driven Pattern-Based，MDPB）的开发方法填平了 J2EE 开发所需要的经验这道鸿沟，并帮助企业提高了企业应用开发的生产力。

Compuware 公司(Nasdaq: CPWR) 今天描述了应用 MDPB 支持软件开发者成功构建面向服务的企业级应用的远景。为支持其实现，Compuware 推出了其受欢迎的 Compuware OptimalJ 开发平台的 3.1 版本。该工具具有业界领先的对 Web Services 安全机制的支持，并有更多在 Compuware OptimalJ 的集成测试环境方面的支持--包括 for BEA WebLogic Server 和 IBM WebSphere 应用服务器的测试支持--以及在集成遗留代码方面的加强功能。

Gartner 的副总裁和研究主任 Michael Blechar 认为，“随着 J2EE 应用的日益成熟，基于架构的代码生成的 MDA 将会非常流行，就象 90 年代的第四代语言一样，当开发组织在寻找更好的，更高生产率的支持新旧业务集成的开发方法时，他们中的大部分都会使用 MDA”。

MDPB 方法通过模式来将业务模型自动转换成为应用程序，以使得更高质量的应用可以被更快地开发出来。在寻找用更少的资源开发更复杂的企业级应用的软件组织中，越来越多的正在转向应用 Compuware OptimalJ 的 MDPB 方法。

“通过各种最佳实践，Compuware OptimalJ 帮助我们将成本和应用的复杂度最小化，这对于我们进入 J2EE 开发领域非常重要”，Richard Seegmiller，Micro-News 网络公司的 CIO 认为，“Compuware OptimalJ 帮助我们开发开发了我们的旗舰应用，并使我们对这一点充满信心：我们在为未来的工作打下坚实的基础，市场上没有其它的工具提供了如此的价值和生产力，Compuware 的高级开发环境是 MDPB 工具市场上的领袖”。

Compuware OptimalJ 3.1 的加强功能

Compuware OptimalJ 3.1 帮助开发团队通过更少的努力得到更多的收获，并通过以下几点帮助将业务轻松地转移到 MDPB 的开发模式上来：

* 领先的 Web Services 安全机制-- 提供了支持 OASIS（the Advancement of Structured Information Standards）定义的新的 Web Services 安全规约的第一批的开发工具。保证企业可以通过 web services 安全的构造业务应用。

* 提供灵活的建模支持 -- Compuware OptimalJ 中不但提供了对 UML 建模的支持，还扩展了和各种世界级建模工具的集成，包括 IBM Rational Rose、Borland 的 Together Control Center、SparxSystems Enterprise Architect 以及 Objecteering。

* Driving Legacy Modernization -- 扩展了对 IBM 架构软件的支持，集成了 WebSphere MQ，因此保护了 Compuware OptimalJ 的顾客在已有的底层架构（infrastructure）上的投资。

* Ensuring Platform Flexibility -- Compuware OptimalJ 增加了集成测试环境的部署选择，包括业界领先的应用服务器 BEA WebLogic Server 和 IBM 的 WebSphere，这将加速 IT 组织创建、测试和调试应用的进程。

对生产率更高的企业应用开发模式的展望

Compuware 正在努力创造一种围绕 Compuware OptimalJ 的开发环境，加速 MDPB 过程的推广应用。通过引入互补的、集成的工具，并和互补的工具进行集成，并依助在行业标准组织中的领导地位，Compuware 正在依助其企业应用开发方面的技术领导新的软件开发范式。

在其展望中，Compuware 最近宣布和 BEA 达成伙伴关系，集成 Compuware OptimalJ 与 BEA 的 WebLogic Workshop。另外，Compuware 还和 BEA 以及其它的业界领袖例如 Sun 和 Oracle 等一起组成了 JAVA 工具社团（Java Tools Community，JTC），这是一个致力于提高 Java 标准的工具化以及工具之间的互操作性的组织。2004 年，Compuware 将继续打造联盟，并和其它产品集成以支持日益壮大的 MDPB 开发社团。

“MDPB 开发使得公司们可以克服最大的 J2EE 的挑战：在有限资源下的生产率问题，随着越来越多的顾客认识到他们需要某种形式的自动化来封装 J2EE 的经验，我们将会看到 MDPB 在早期推广之后走向成熟”，Dan Schoenbaum，Compuware 主管策略的副总裁认为，“在架构模型自动生成代码方面，Compuware 是最早的，也是最有经验的开发商中的一员。通过帮助组织快速地响应业务变更、提高开发效率以及大幅度削减维护费用，我们正在逐步实现企业化 Java 的承诺。”

（自 prnewswire，袁峰 摘译，不得转载用于商业用途）

LogicLibrary 书写 Logidex 的新篇章

[2004/2/6]

软件资产管理工具的开发商 LogicLibrary 发布了 Logidex 3.0 for J2EE 版本。据 LogicLibrary 宣称，该产品实现了与 Eclipse、WebSphere Studio Application Developer (WSAD)以及 IBM Rational XDE 的无缝集成，使得几个主要的开发工具的快速协作成为可能，也大大地提高了开发和集成等项目的开发效率。

Logic Library asserts Logisex 是目前在开发、集成和面向服务架构（SOA）等项目方面业内首屈一指的软件开发资产 SDA（software development asset）的管理方案。for IBM Rational XDE 的 Logidex RAS 插件使得开发人员可以快速地查找及使用恰当的 Reuseable Asset Specification (RAS) 资产，并导入到 IBM Rational XDE 中进行基于 UML 的各种开发活动。通过为诸如 WSAD, XDE, Eclipse 和 Microsoft Visual Studio .NET 等领先的开发环境提供插件支持，Logidex 用户可以通过这些开发环境搜索并使用合适的资产来满足应用开发和集成的需求。

今天，大多数业务都涉及到各种不同的环境，我们将帮助程序员解决这些不同开发环境之间的一些问题，Logidex 3.0 for J2EE 就是我们最新的演示。LogicLibrary 的 CEO Greg Coticchia 说，作为 Eclipse 开源运动的积极一员，我们认为，如果可以通过单一的接口来使用最好的工具，那么开发人员和架构设计师的生产效率将得到大幅度的提高。在和 Eclipse、IBM WebSphere Studio、IBM Rational XDE 以及 Microsoft Visual Studio .NET 的紧密集成上，没有其它哪个软件资产管理工具可以和 LogicLibrary 相匹敌。用 LOGIC LIBRARY 的话来说，Logidex 3.0 for J2EE 还包括：

---增强的安全性---Logidex 的 LDAP (Lightweight Directory Access Protocol) 接口提供了权限信息的统一管理。Logidex 的用户可以通过本地 LDAP，也可以通过业界领先的 single-sign-on 产品进行权限验证。

---工作流加强---新的资产获取功能为 Logidex 的用户提供了在管理 SDA 选择和获取上相关规则的空前的灵活性。作为获取过程的一部分，IT 组织可以为预期的 SDA 用户创建评审检查单（review checklist），这些文档的覆盖面非常广泛，可以是开源的组件 license、功能用法评审，也可以第三方组件的购买理由。这些检查单将帮助公司正确地获取 SDA 并使用。

---集成的库（library）支持---通过支持远程的库（library-to-library）连接，Logidex 使得企业可以更好地管理物理分布的 SDA。配置这些连接，可以通过一个本地的库共享任意一个或者是所有的远程库上的 SDA。这样，Logidex 中增强的 SDA 搜索能力保证用户可以搜索或者近在眼前，或者远在天边的各种软件资产。

（自 ebizq，袁峰 摘译，不得转载用于商业用途）

Martin Fowler 关于 MDA 的见解

[2004/2/2]

以下摘自 Martin Fowler 的 Blog：

一些人认为 MDA 将是软件开发上最大的变革：从对象装配到最高级的语言。其它人认为它不过是现在这些 CASE 工具的暮年而已。我站在后者的阵营中，但我还想多说几句。

现在 MDA 所说的很多东西 CASE 工具社区在 80 年代就讨论过了。我认为 CASE 工具失败的原因有很多，但最根本的是它们没有能够提供一个一致的编程环境，允许人们以比其它方法更有效的方式构建通用的企业应用。

当然，在某些工作上 CASE 工具是很有帮助的。例如，我更愿意使用图形化的工具来设计数据库而不是在记事本上敲 SQL 命令。但是，很多工作根本就没有可能或者说很难在 CASE 环境上实现。

因此，我所怀疑的就是， MDA 是否改变了这一点。UML 最早是很合理的，目的就是要帮助人们表达设计的思路，我偏向于这么使用 UML，UmlAsSketch（http://martinfowler.com/bliki/UmlAsSketch.html）。但 MDA 需要从 UML 得到最终的产品，它对 UML 的形式化和一致性的要求要高得多。当然 UML2 是很多人在这方面的努力，试图保证其计算完整性（computationally complete）。但很多工作只是停留在纸面上，并没有清晰的例子或者将 UML 应用于哪个平台上的实际经验。即使 UML 具有计算完整性，也需要有效的环境来应用它，以取代其它的开发方法。作为一个同时了解双方的人，我不得不说，这样一个环境，我还没有看到。

举例说吧，以行为逻辑（behavioral logic）为例，我看不出使用序列图或者活动图比使用现代语言直接写代码有什么长处。我发现，即使我必须要以更细节的方式来写代码，我也愿意，而不是选择画这些图，因为我可以执行代码并且测试它们。

即使 UML 提供了一个高效的编程环境，也有一个推广的过程。作为一个过去的 smalltalker，我明白一点：即使是最好的语言，也不一定能够成为主流。

支持 MDA 的观点还有一些，但也都难以令人信服。

1)拥有一系列 OMG 标准的支持，固然这是 80 年代的 CASE 工具所缺乏的，但这得看人们是不是都遵循这些标准。我很想知道到底有多少 MDA 的 fans 把 UML 当做了不想要的语言 UnwantedModelingLanguage。

2)MDA 的支持者说到平台无关性，我在 PlatformIndependentMalapropism.中已经谈到这个问题，（译者注：Martin 在那篇文章中认为这是一个可笑的说法）。

3)我听说 MDA 将如何通过模式的自动生成来简化开发。但我并没有看到用 UML 来实现这一点和借助好的库（library）和框架来实现这一点有什么区别。

4)大部分 UML 的支持者都有一个基本的论断：图要好过文字。有时候这是成立的，但我没有发现有证据证明它总是成立的－比较过流程图和伪代码的人可以得到自己的结论。

不管怎么说，如果我发现我的判断错了，我会非常高兴。我真的愿意看到软件开发的抽象层次再提高一层（更不要说 UML 的成功对我个人而言绝对是利好啊）。但我就是没有看见 UML 如何提高了抽象层次，而这，正是 MDA 成功的必要条件。

有趣地是现在有一种现象：越来越多的人希望不遵循 OMG 的标准来实现 MDA。我听的更多是借助工具实现模型驱动开发，而不是借助 OMG 的 MDA 标准族。

这里还有一些其它关于 MDA 的富有思想的批评声音：

*Steve Cook 关于 Microsoft's views on MDA 和更多的关于 MDA 的谈话。Steve 是 UML 的主要贡献者之一，也是英国 OO 早期的领袖之一。

*在 OOPSLA 2003 会议上，"Bedarra" Dave Thomas 对 MDA 的方方面面提出了怀疑。遗憾的是我没有当时他演讲的视频，但 jot column 上摘录了他的一些观点。

（自 http://martinfowler.com/bliki/ModelDrivenArchitecture.html，袁峰 摘译，不得转载用于商业用途）

Metamill 软件公司和 Innovative 及 Datraverse B.V.签订经销协议

[2004/1/28]

Metamill 软件公司骄傲地宣称他们和一家巴西公司，Innovative 软件公司，签订了经销协议，后者是为 Metamill 提供集成支持的全面需求管理系统的开发者，另外，Metamill 软件同样地宣告了和荷兰公司 Datraverse B.V 达成的类似的转售协议。

顾客现在可以从本地经销商那里购买 Metamill 的 license，并且得到本地语言的支持。这是走向国际化合作和支持的重要的里程碑。

Metamill 是一种 UML 的 CASE 工具，提供面向 C++、C、Java 和 C#的双向代码工程的支持。Metamill 有着 Linux 和 Windows 两个版本。Windows 版本下的单用户 license 价格为 125 美元，Linux 版本的为 75 美元。另外，也提供有多用户和 Site-license。用户可以通过 Metamill 的网站在线购买，也可以和本地经销商联系购买。在 www.metamill.com 上可以下载 30 天的试用版本。

（自 emediawire，袁峰 摘译，不得转载用于商业用途）

斐力庇第斯从马拉松跑回雅典报信，虽然已是满身血迹、精疲力尽，但他知道：没有出现在雅典人民面前，前面的路程都是白费。

学到的知识如果不能最终【用】于您自己的项目之中，也同样是极大的浪费。而这最后一段路最是艰难。

UMLChina 聚焦最后一公里，所提供服务全部与您自己的项目密切结合，帮您走完最艰难的一段路。

Grady Booch 访谈

UMLChina 整理

查看评论

在上周 Anaheim 举行的 EclipseCon 会议上作完主题发言之后，Grady Booch 接受了 CRN 高级编辑 Elizabeth Montalbano 的采访，Grady Booch 是 Rational 公司的创办人之一，如今是 IBM 的一员，在软件开发领域广受尊敬的权威。访谈涉及了 Booch 在 IBM 研究院和 Rational 之间的联络活动，在创办计算机历史博物馆，保留重要软件源代码方面的努力，以及 Eclipse 在软件开发行业的长远的影响。下面是这次访谈的摘录。

CRN: 在主题发言上，您强调了软件开发的复杂性以及必须做出哪些改变。那么 IBM Rational 在这一点上是如何应对的呢，尤其是 IBM 现在开发的软件看起来都是无比复杂？

Booch: Rational 在 IBM 的任务之一是进行开发的组织。对外而言，我们的使命是工具。这也是为什么 Lee [Nackman, Rational 的 CTO] 和他 400 个最亲密的朋友（从 IBM 的 WebSphere 开发组）加入我们的原因。这确实保证了 Rational 的生命周期开发工具集和过程的统一，并且和开发环境紧密集成在一起。被收购之前我们就和 IBM 在他们的工具上有过长期合作，同样，还有微软。现在我们也仍然是微软的白金拍档，我们对工具关注颇多。作为 IBM 的一部分并且承担这个使命的好处在于我们可以和 WebSphere 走得更近，这样我们可以使我们的工具更易于利用 WebSphere 平台，同时又不忽视 .Net，因为对 IBM 来说，.Net 仍然是非常重要的。

比较酷的是，如果你看了我前面在这里谈过的软件发展的趋势，Rational 所做的正是把这些都集中起来。我们在整个模式领域和面向方面开发（aspect-oriented development）领域都有投入，不用说了，这些肯定会对明天的开发人员带来冲击，我们认为这些事情会长久影响开发人员，而且我们将继续在这些领域深入研究。

我的一个角色就是要管理好我们和 IBM 研究院之间的联络。我得到了比在 Rational 更多的东西，这让我觉得就像一个孩子到了糖果店里面。IBM 原来没有地方去安置他们在工具方面的研究，现在有了。坦白地说，这对我们来说就是一个秘密武器，因为现在我们可以进行一些周期较长的研究，这些都太酷了。

CRN: 可不可以谈谈其中的一些技术，以及它们是怎样在 IBM 的软件和工具产品中得到应用的呢？

Booch: 从技术上看到的趋势上来说，我将提到其中的三个-协作、面向方面编程以及模式。在协作开发环境方面，我们在团队的参与和 virtualizations 上在做一个[名字 为 Jazz 的] Eclipse 插件，它会解决软件开发中人这方面的问题。其实 SourceForge 已经做得很好了，它帮助分布在不同地理位置或者临时分开的团队成员得到统一的管理。因此 Jazz 的项目是解决一个地方的问题，这是 IBM 在收购 Rational 之前就已经开始的一个研究。但我们在寻找将他们绑定在一起的途径，现在使用的途径就是作为一个 Eclipse 的插件。

第二个要提到的就是面向方面编程（AOP）这个领域了。Gregor Kiczales 是 AOP 之父，AspectJ（一个 Java 的面向方面扩展）的发明者。这个 AspectJ 团队已经从 Xerox 中分离出来，现在隶属于 IBM 在英国的 Hursley 实验室。

建造分布式，并发，安全的系统不是一般开发者的核心竞争力，理由是，我们可以为安全建立全局策略。所以我已经把实现这项策略的代码写在一个独立的方法里。问题是，可以在本地作一些简单的改变，它们会给全局策略带来大的变更，而一个对这些策略没有深刻认识的开发者，可能只会弄得一团糟。AOP 认为，让我们来接管这些地方——从某个特定涉众的角度引起我们顾虑的许多地方——例如一个安全官员，如果你愿意——并把它们凑到一起合并成一个关注点，然后现在应用那些真正在我们系统中把它串起来的工具。一个可恶的问题是，它是一个交错的关注点，因为那些策略四处分布在系统的各个不同部分，然而还需要把它当作整体看待。AOP 说，“让我们想办法做到怎样写出这些全局事物的语义，并找到一个途径去使用那些把它们分布到系统里的工具。”

在这个领域有三个主要的选手：施乐的 Gregor，IBM Hursley 实验室，和一个名叫 Charles Simonyi 的人。前面两个拥有某种联系因素，Gregor 不在 IBM，但 AspectJ 团队跟我们在一起。Charles Simonyi，他写了[Microsoft] Word，是微软的第 14 号员工。AspectJ 团队其他成员加入了 IBM Hursley 实验室之外的其他组织，所以 AspectJ 仍然和 IBM 在一起，尽管它已经从施乐分离出来。

如果你想对模式多一些了解，请访问 hillside.net，那就是 Hillside 组织，我是其中一名发起人。它是我们这群聚在一起，想发扬模式概念的家伙的组织。这个领域的经典书籍就是“Gang of Four”的“设计模式”－－ Erich Gamma，John Vlissides，也在 IBM，Richard Helm 和 Ralph Johnson。在我看来，这是最近十年来最重要的发展和软件理论，因为它给了我们一种语言来描述一起协作的类的集合。它也是一种对我们一次又一次看到的普遍设计进行命名的方法。用普遍的方法来解决普遍的问题，更酷的是，它就建立在编程语言之上。所以，我们现在有了一个平台－－J2，.Net，你们命名的，它们都非常复杂，每个人都承认－－甚至平台开发商们也这么说。但这里的平台和我们想要交付的平台之间有一个很大的差距，不是代码上的差距，而是语义上的差距。要弥合这个差距不能直接在中间件上建造，而应该在它们之上建立一些模式，所以，在某种意义上，对于所建造的系统，这是一种领域特定的语言。

CRN：对更多您所指的协作开发环境来说，您认为 Eclipse 将怎样有助于进化性的变化？

Booch：Eclipse 代表“大众化”的工具，就象我们看到 Linux 代表“大众化”的操作系统一样。在很多方面，因为市场经济学，这些好处是必然的。这就是为什么 Linux 存在的原因。

Eclipse 代表的是工具市场上的大众化产品，(因为)这个时刻我们大体上知道自己需要的基本工具。Eclipse 以 Java 开始，因此我们有机会建立一个每个人都能用的工具集，有效地把市场做得更大。我非常乐意看到 Eclipse 不仅仅成为一个 Java 开发环境，而是它有机会成为各种各样的工具环境。事实上，它可扩展性非常强，为后面那些（工具集）创造了基础。Eclipse 就是这样敞开，让另一代人开始建造插件，来满足不同涉众的需要。驱动这一代 IDE 的不再只是代码战士们（使用它们），所有其他的人也一样。当然也并不意味着完全重新发明。因为这些涉众需要跟我们互相交流，有一些基础，因为他们可能有一些开发经验，共享产品，共享工具，意味着现在有可能做各种特化。以前不可能做到。（当然）我们只是没有一个那么做的地方。

CRN：您怎样看待类似 Eclipse 这样的产品对应用开发和软件工具的影响？

Booch：过去几年里，工具不是 IBM 的赢利项目，它们只是跟着平台发展。Rational 已经证明确实存在可行的工具业务。对 Eclipse 来说最好的地方就是，我们可以从某个不必自己亲自创造的东西开始。所以我们从它所有的经验获益。如果你想想 Rational 在工具市场试图解决的问题，不是单个开发者的需要，而是整个开发团队的需要，所以在这个意义上，我们没事－－安于在 IBM 提供的基础上建造。

CRN：在您的生活中通常一天是怎样度过的？

Booch：我是一个自由激进派。下面就说说我生活中的某一天。很难总结到一天，但是如果看看一个星期，我可能花一天半跟各种其他团体通电话，每周有一个研究电话，因为我和 IBM 的研究实验室一起工作。每周一个电话，跟我们的模型驱动开发人员。每半周一个电话，跟一个 IBM 举行的新闻节目有关，每周和 CTO Lee 通一个电话——我仍然保持跟整个策略团队在一起。那是我每天的例行之事，（就象）每天用牙线清洁。我正在进行的三个项目，就是其他我所做的事了，一个是在模式这边，跟 IBM 一位叫 Jonathan Adams 的先生一起工作，工作叫做“IBM 在 e-business 上的模式”，开发模式的业务模型。

第二个是一个回顾性的项目，跟 Computer History 博物馆[在 Mountain View, Calif.]一起，关于经典软件的保存。这是一个很酷的故事，我们的想法是，我们处在这样一个时刻，这个行业创建了这么酷的产品，如果现在不保存，随着时间的发展我们将失去它们。因此我们确实在努力保存各种经典系统的源代码，例如 Word，或 SmallTalk，因此从现在开始的 50 年，我们都拥有源代码。

然后是我耗时最多的项目，开发一个软件架构的手册。这是我的五年计划项目，如果你看看土木工程，或机械工程，在这些行业中人们的学习方法是他们考虑现存的系统，研究架构。在软件上，我们没有这样的研究，没有人把这些编成文献，所以我选择了 100 个系统，Eclipse 也在其中，我的想法是把它的架构文档化，因此我们可以描述它，可以做一些比较性的研究。我正在考虑例如 Word 的架构，纽约交易所交易系统的架构，Prism 1000 的架构，Prism 1000 系统用于对人类 DNA 进行编码。因此这个想法是去真正得到某个软件系统的光谱，并提供如何建立这些架构的知识体。

在 Grady 一周中，我可能花 10%的时间作为导师，我指导一些人。还有和大学同僚一起工作。每周在世界各地大约有 100 个人跟我一起工作，我可能会说：“嗨，让我们看看这个方向。”或者向他们学习。这就是我平时的生活，当然，偶尔睡一觉。

Agile软件开发丛书

PEARSON Addison Wesley

有效用例模式

Patterns for Effective Use Cases

Foreword by Craig Larman

[美] Steve Adolph
Paul Bramble 著

车立红 译

UMLChina 审

UMLChina 指定教材

清华大学出版社

个案研究：聊天室 UML 模型中的一致性问题

Thomas Huining Feng、Hans Vangheluwe 著，车皓阳 译

查看评论

摘要

本文从初始需求开始构建聊天室模型，以及对个案进行研究。在不同的开发阶段，分别要用到 UML 类图、时序图和状态图。这样，难免需要确定一致性问题，现在已经提出了许多仿真和方法，用来确保模型各个方面的一致性。我们关注内部一致性，即给定模型内部制品之间的一致性问题。

1 简介

软件系统的开发过程通常会被划分成一些步骤，每个步骤会用到不同的 UML 图。由于建模系统变得越来越复杂，一致性问题就越发突出起来。而在其中，有两种类型或问题更为显著。第一，内部一致性问题，涉及给定模型内部制品之间的一致性。第二，系统之间一致性问题，涉及软件开发过程中不同演化模型之间的一致性。我们主要关注于内部一致性问题。

不同的文献提出了不同的形式化方法，用来自动检查一致性，发现设计当中存在的问题。在下面几节里，我们分步研究了一个聊天室的开发过程。本文的灵感来源于 Agder 大学 Geir Melby 完成的一次项目报告（http://fag.grm.hia.no/ikt2340/year2002/）。这个模型提出了一些潜在的一致性问题。对于它们当中的部分内容，也给出了自动化的一致性检查方法。

在这份案例研究中，我们从需求开始开发了一个聊天室模型。第二节给出了客户、管理器对象以及聊天室之间的通信协议，作为初始需求。第三节研究了一个可能的类的设计，它定义了符合协议的接口。第四节中给出的时序图进一步定义和描述了组件之间的通信，并保持与类设计之间的一致性。第五节使用状态图更进一步地确定应用程序的行为。这份规程可以在我们的 SVM（状态图虚拟机）环境中实时地进行仿真或执行。在第六节中，讨论了协议规程与仿真迹的一致性。第七节进行总结。

2 通信协议

我们将要创建的聊天室程序是按照客户机/服务器范型来架构的。客户会随机连接聊天室。如果某个聊天室接收了客户，客户就会发送消息给这个聊天室。然后聊天室广播每条消息，除了发送者以外，每个与聊天室建立连接的客户都会收到一份拷贝。

下面将要描述一个特定的简化了的用例：

- 系统包括五个客户和两个聊天室。客户端最初没有连接。每隔一到三秒（非均匀分布），它们都会随机连接一个聊天室。被请求的聊天室同时收到请求（假定没有网络延时，并且通信可靠）。
- 一个聊天室至多接收三个客户。当且仅当容量没有超标时聊天室才会接收连接。
- 发出请求的客户会立刻收到接收或拒绝通知。
- 在客户可以发送聊天信息之前必须被一个聊天室接收。
- 连接建立之后，客户会每隔一到五秒（非均匀分布）给它所连接的聊天室发送随机消息。聊天室会立即接收消息，它将耗时一秒钟来处理接收到的消息，并把它广播给除发送者之外的所有连接客户。
- 客户会同时收到广播。为了简化起见，我们不讨论没有连接的情况。

3 类的设计

根据上面的规定，显然我们需要两个类：Client 和 ChatRoom。在开发过程早期阶段，对于仿真来说无需用户干预，我们明确地按照随机过程对用户行为（连接请求和聊天信息）进行建模。以后，客户模型将会被与软件打交道的真实客户所代替。仿真开始时，会初始化五个 Client 实例和两个 ChatRoom 实例。

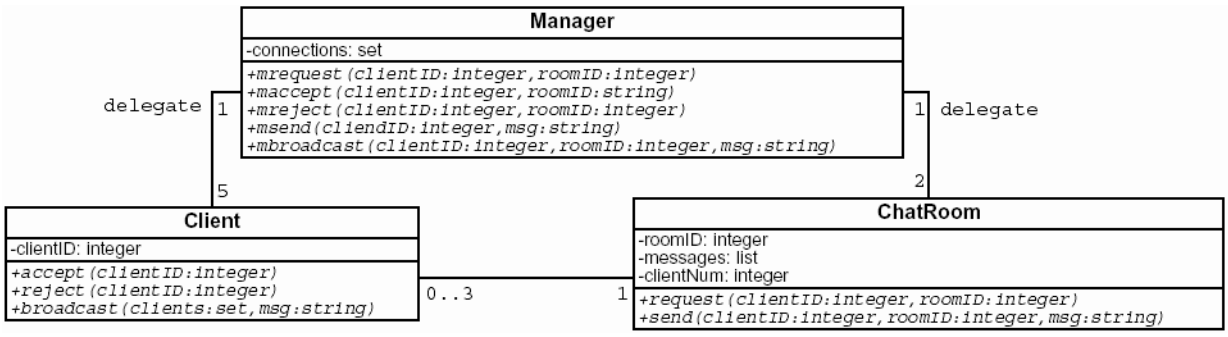
我们还加入了一个 singleton 类：Manager。Manager 以仲裁者的身份中继组件之间的所有通信。这种中央控制措施将会帮助截取系统传递的所有消息，使用它们就可以检查模型的正确性。

图一描述了由这三个类组成的 UML 类图。

✠ ChatRoom 提供两个方法处理到达的事件。request 处理来到的请求，每个请求都有参数 clientID 和 roomID。ChatRoom 把接收或拒绝通知发回拥有全局 ID 标识 clientID 的发送者。它也使用参数 roomID 来决定请求是发给它自己还是发给另外一个聊天室了[1]。send 方法接收由客户 clientID 发送的消息 msg。这条 msg 将会在一秒钟之后被广播出去。

✠ Client 的方法 accept 和 reject 处理到达的接收和拒绝通知。参数 clientID 用来确定目标客户。当一个 Client 接收到一个 broadcast 事件，它会检查自己是否在 clients 集合中。如果情况确实如此，消息 msg 就会在输出中被打印出来。

✠ Manager 中继连接请求、接收和拒绝通知、来自客户的消息以及聊天室的广播。例如，如果它收到来自聊天室的广播，有三个参数会告诉它消息最初的发送者（客户），广播者（聊天室）以及消息字符串。然后它把消息发给所有连接在聊天室中的，除了最初发送者以外的客户。



图一 类的设计

尽管这种 API 定义不是形式化的，但接口背后的行为还是很容易理解的。但是，检查协议的一致性会有些困难，甚至是不可能的，原因如下：

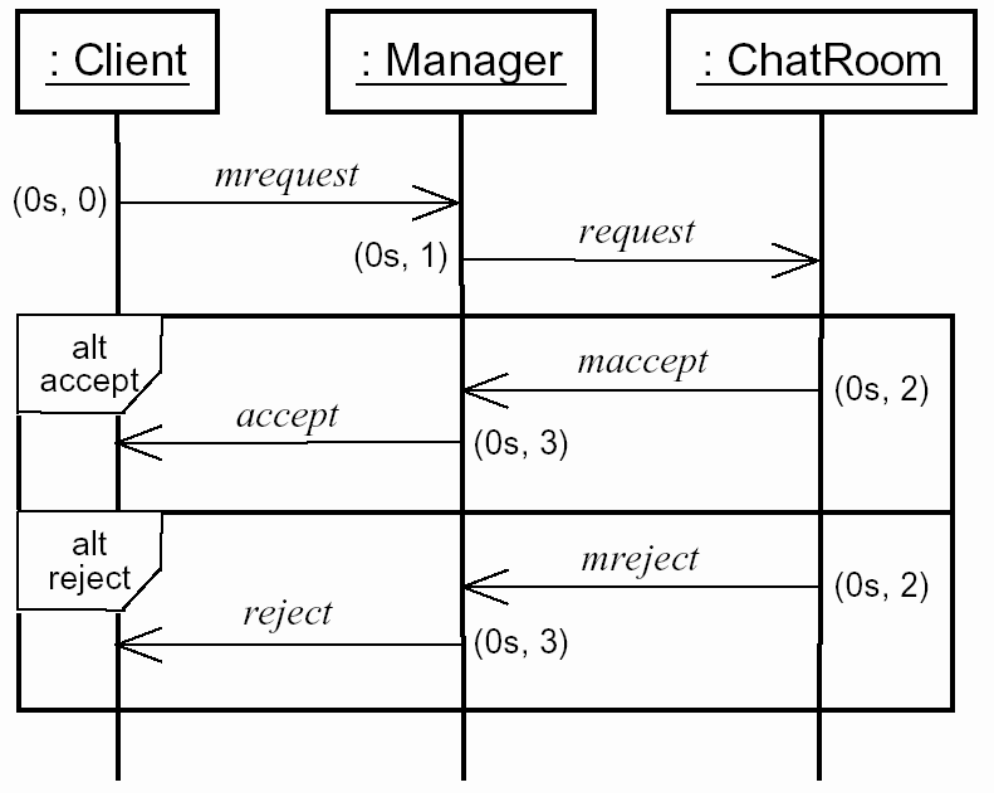
✠ 行为隐藏在接口背后，它只能在理解的基础上进行解释。

✠ 协议是用自然语言表达的，程序不可能很容易地处理。

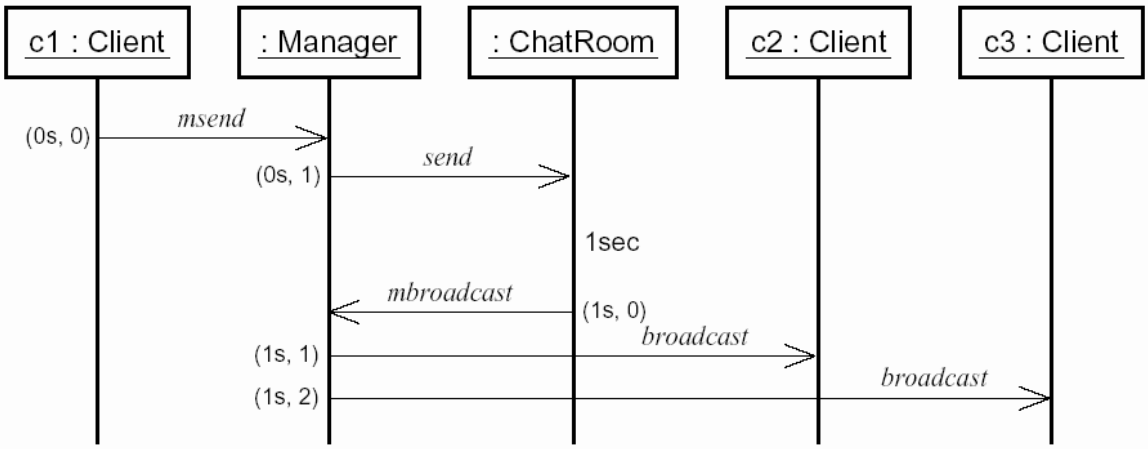
✠ 对于一个定义完好的系统来说，会有许多接口设计。它们可能有相当大的不同。例如，本设计中用 Manager 类来拦截通信。另一项设计可能会使用 RequestManager 来拦截请求、接收和拒绝，用 MessageManager 来拦截消息和发送广播。更有另外一种设计可能根本就不会使用任何管理器。

4 时序图

本节讨论的时序图把设计带入比类图更低的抽象层次（更高层次的细节）。时序图必须清晰反映组件之间的交互。



图二 请求模式的时序图



图三 消息模式的时序图

4.1 定时

定时问题使得协议转化为时序图和之后转化为状态图的过程变得困难。在协议描述中，同一时刻会发生一个或多个动作，即使这些动作在某种原因下相关也是如此。例如，聊天室不应该在收到请求之前发送接收或拒绝消息。这样的话，输出迹中出现“在时刻 1 请求；在时刻 1 接收”就是正确的，如果出现“在时刻 1 接收；在时刻 1 请请求”，那就是不正确的。

一种可能的解决方案是使用 tuple(*t*, *s*)表示时间，*t* 是以秒计数的浮点时间，*s* 是整数序号。以这种方式，正确的输出可以写成“在时刻(1.0s, 0)请求；在时刻(1.0s, 1)接收”。序列“在时刻(1.0s, 0)接收；在时刻(1.0s, 1)请求”则是不正确的。

4.2 请求和消息模式

请求模式如图二所示。Client 首先会调用 Manager 的方法 mrequest。然后 Manager 通过调用 ChatRoom 的方法 request 中继这个调用。ChatRoom 立刻响应，回调 Manager 的方法 maccept 或 mreject。然后请求 Client 会接收到由 Manager 中继的响应。

图三是消息模式，在系统中随机产生消息，进行传递。注意 ChatRoom 在收到一次请求后故意延时一秒钟。在这两个时序图中，没有其它延时存在。

4.3 类图和协议之间的一致性问题

与类图之间的一致性是容易检查的，可以通过收集组件接收到的所有方法调用来达成。例如，在请求模式中，Manager 接收 mrequest、maccept 和 mreject。在消息模式中，它接收 msend 和 mbroadcast。这五个方法在类图中都有定义，但没有定义其它的公有方法。由于时序图中并没有给出参数，那样就没有检查参数的必要。检查过程可以自动化。可以部分检查协议的一致性。从请求时序图中很容易就能看出，聊天室在时刻 0 收到一个请求，在时刻 0 接收或拒绝这个客户。这两个时刻的绝对值并不重要。重要的是，回复确实会在同一时间发回。这样设计者就可以手工检查“某个时间应该会发生什么”。如果用到后面讨论的基于规则的方法，有限的自动检查也是可以做到的。

注意基本的时序图无法表达某个时间或某个阶段不应该发生的事情。比方说，协议中会含有这样的意思，聊天室在没有收到请求时不会接收或拒绝客户。时序图中不会包含这样的信息。这会在模型中带来设计错误，影响后续开发步骤的正确性。另一种可能的设计错误是时序图的语义。例如，在请求模式中，时序图描述：如果客户发送 mrequest，管理器没有时间前置地发送 request，聊天室发送 maccept 或 mreject，也没有时间前置，然后管理器发送 accept 或 reject。不幸地是，迟钝的客户不会发送任何请求，这显然是系统当中的一个问题，不能通过检查时序图检测出来。最坏的情况是根本没有客户连接，这样系统就会永远停机。为了补偿信息丢失，需要 UML 形式化体系中的其它设计，它们并不完全依赖时序图，或者就要对时序图进行扩展。一个极好的使用时序图并引入扩展的例子是 Live Sequence Charts，参见 Harel 的文章[1]。

5 状态图

状态图用来实现类定义背后的行为。它们可以在我们的SVM（状态虚拟机）[2][3]中执行，用于扩展状态图形式化的解释器是用Python编写的。

5.1 SVM 约定

如果想要轻松理解状态图设计，就必须先了解SVM的一些约定。

SVM用扩展状态图形式化体系解释模型。加入了一些新的功能，尽管表现力没有加强[2]，但易用性却大大改进了。

尽管最初的状态图并不是模块化的，但仍然可以使用SVM实现基于组件的设计。对于聊天室模型来说这是必需的，象客户和聊天室这样的组件可以独立进行设计，但最终在系统中还是要一起工作的。组件通过导入可以进行复用。较大的组件导入一些小的组件（实例），作为它的一个状态。结果是，就象被导入组件的所有状态（分层的）和转换都是直接在这个状态当中编写一样。

SVM模型用文本文件编写。宏是在SVM中引入的一个概念。宏在SVM源文件里的MACRO节中定义。一旦定义后，在整个文本文件中就可以括在括号中使用。例如，定义PREFIX=state，[PREFIX]就可以用来代替字符串“state”，这样[PREFIX]1就等于state1。

后面就会用到其中的一些预定义宏。[EVENT(event, param)]会触发一次事件。它的参数是字符串event和可选列表param。[PARAM]用来检索在处理的事件的参数。它通常在监视哨单元或转换的输出中使用。[DUMP(msg)]打印调试信息或在文本文件中记录这些信息。

组件被导入的时候宏也可以充当参数。进行导入的组件要重新定义一部分或所有最初在被导入组件中定义的宏，也包括预定义宏。继续前面的例子，如果进行导入的组件确定PREFIX=mystate作为导入参数，被导入组件中的[PREFIX]1将会译为mystate1。

很容易就可以看出，这些扩展并没有增加状态图的表现力。

5.2 扩展状态图形式体系中的聊天室模型

Client、ChatRoom和Manager组件分别在独立的状态图中设计。如图四所示，Chat模型导入了五个Client实例，两个ChatRoom实例和一个Manager实例。同一类型的每个实例都有一个惟一的ID参数。由于可接收的事件集合是不相交的，因此不同类型的实例就可以拥有相同的ID。这个模型可以在SVM环境中仿真或执行。

Client组件如图五所示。最初，它处在nochat状态。每隔一到三秒（非均匀分布）会触发一次mrequest事件，通过管理器来重复连接聊天室，直到请求被接收为止（accept事件被接收）。uniform是一个Python函数，它返回某个区间内的随机实数值，randint返回随机整数值。事件的第一个参数给出客户的惟一ID。第二个参数给出目的聊天室（随机从1或2中选取）。然后，客户转移到状态connected，开始发送消息和接收广播。事件参数是作为列表发送的，[PARAM][0]访问第一个参数，依次类推。注意，当方括号之间的内容不是宏的名字或Python下标时，那它就是监视哨，遵循最初状态图形式化体系中的定义[4]。

用户定义的宏[ID]为每个客户指定一个惟一ID。定义ID=0的意思是缺省值为0。它可以由导入组件（在这里是Chat模型）变为一个惟一数。组件的ID很重要。既然整个系统在导入后可以被看作一幅大的状态图，每一个正交组件都能接收到所有的广播事件。这样，给特定客户发送事件的惟一方法是在参数列表中给出接收者的ID。每一个客户都要检查在处理事件之前，它的ID是否匹配。

与Client组件相比，ChatRoom要更复杂。它使用列表messages[ID]把到来的消息进行排队。这就意味着每一个有着惟一ID的聊天室都会有其自己的队列（如果ID=0，messages[ID]与message0相等）。如果当它正在处理前面的消息时（要耗时一秒钟），一条消息到达，新到达的消息就会加入列表之中。收到消息时的时间也被记录下来，这样即使消息进行排队，它的处理时间自到达时开始计算仍为一秒钟。

Manager组件仅仅只是中继消息。在聊天室接收客户时，函数rec_comm(client, room)在一个列表中记录连接信息。get_clients(room, client)查询列表并返回聊天室room中除了client以外的所有客户。get_room(client)返回client的房间ID号。

聊天室消息队列和管理器连接列表是变量使用示例。它们帮助记录模型的状态。严格地说，这仍旧是对最初状态图的扩展，状态必须明确地确定下来。讨论变量已经超出了本次个案研究的范围。

5.3 与类图之间的一致性

基于组件的设计应该严格符合图一中的类设计。再者，组件可以发送事件给接收者，但接收者却不能处理它。或者，发送者提供的参数可能会比需要的要少。这可能会造成严重的运行时错误。

自动检查所有方法调用的发送者-接收者之间的一致性，这样的程序可以写出来。而不是要编写如何在代码级由类型检查器和/或链接器来检查一致性。譬如，Manager接收事件maccept。这意味着它在类定义中提供方法maccept。在处理事件的状态转换输出和监视哨中，要用到[PARAMS][0]和[PARAMS][1]，这样它就至少需要两个参数。检查器遍历整个Chat模型，找出仅由ChatRoom组件调用（异步地）的方法。调用[EVENT("maccept",[[PARAMS][0], [ID]])]提供了两个参数（[PARAMS][0][3]和[ID]）。检查这条调用是成功的。

同样地，在模型中所有方法调用的一致性可以由状态图来检查。

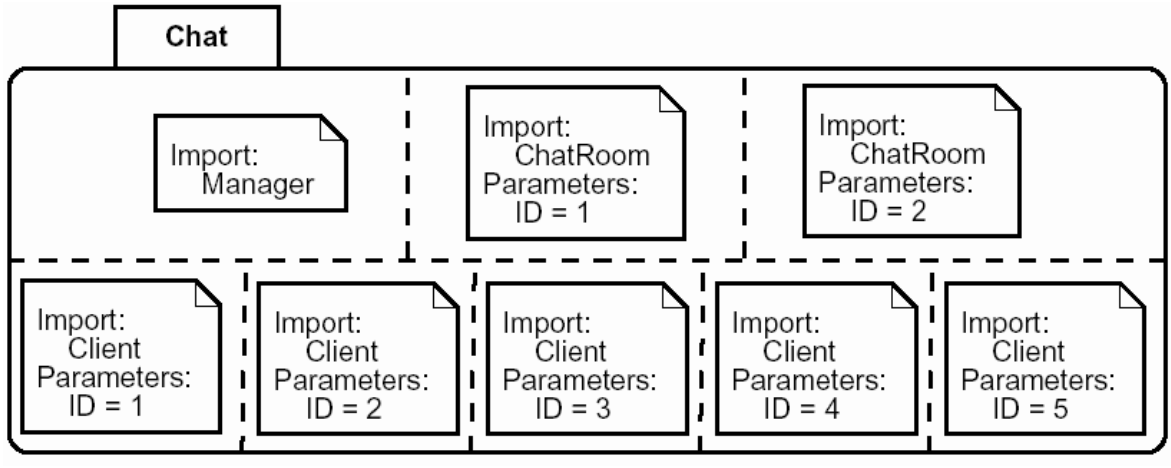
6 模型执行进行的一致性检查

SVM解释器可以仿真或实时（循环中有人参与的情况下需要）执行Chat模型。执行后的输出结果被转储显示并拷贝一份到文本文件。正如上面说的那样，如果所有的用户交互都明确建模的话，根本就不需要干预。输出迹是我们验证执行结果的惟一方法。必须检查所有设计制品与输出迹之间的一致性。

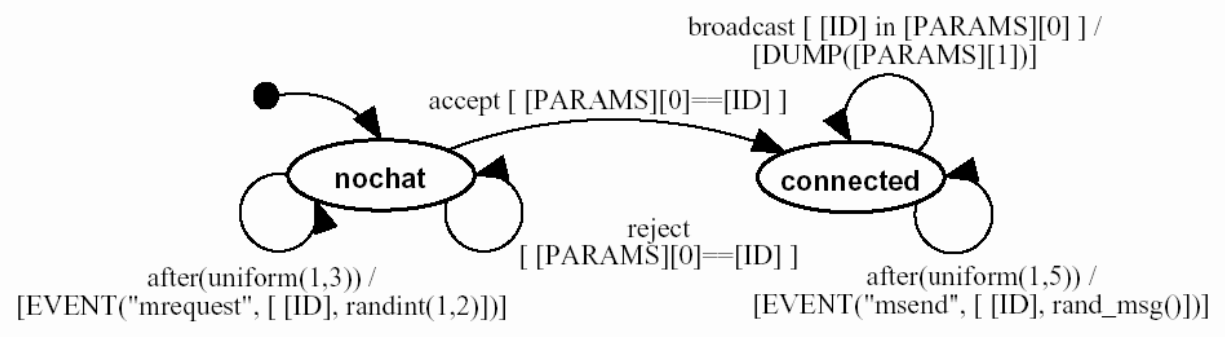
类图一致性在前面一节中已经研究过。检查器形式化地检查状态图设计。不需要模型执行。时序图一致性可以通过验证实验输出迹来检查。尽管许多情况下正确性仍然不能得到验证（搜索较大范围或者有可能无限状态空间的可能行为），但对最终产品的信心还是大大增加了。

状态图一致性含有假定SVM执行环境正确的意思。

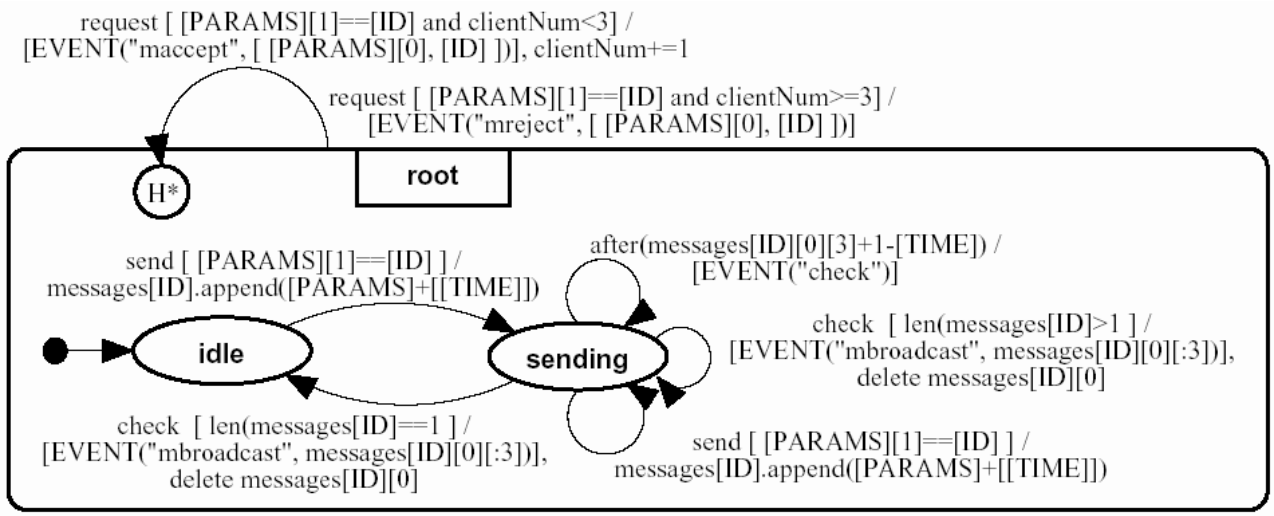
使用最初的协议验证一致性并不容易，因为它比时序图包含更多的信息。检查程序处理起来会非常困难。



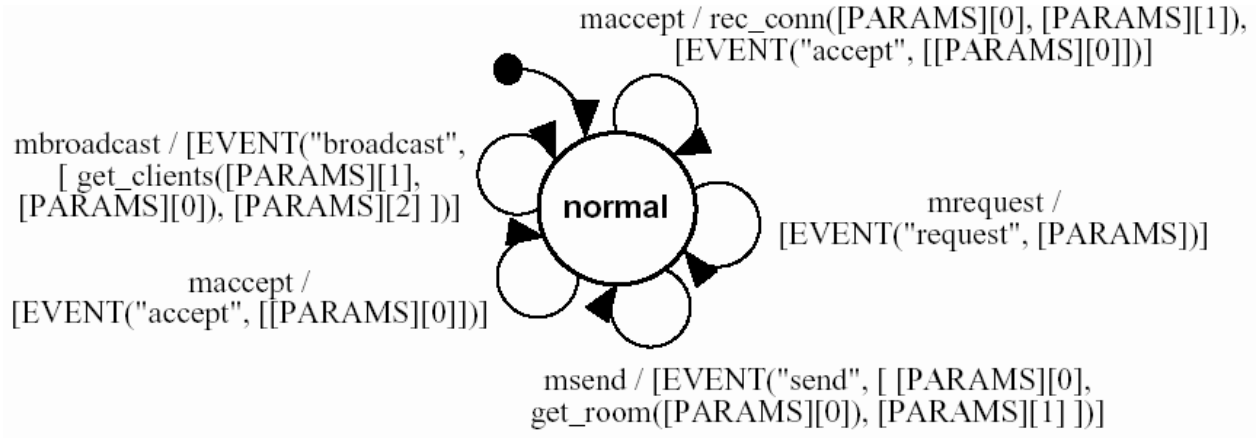
图四 Chat模型



图五 Client组件



图六 聊天室组件



图七 管理器组件

6.1 输出迹

宏[DUMP(msg)]用来在文件中记录消息msg，一直到执行结束为止（或者自动，或者由调试器手动控制）。每条消息包括三个部分：时间*tuple*(*t*, *s*)，有着惟一ID的发送者或接收者，和消息体。下面是从输出中截取的一部分内容：

. . . . . .

CLOCK: (10.5s,0)

Client 0

Says "Hello!" to ChatRoom 1

. . . . . .

CLOCK: (11.5s,0)

ChatRoom 1

Broadcasts "Hello!" to all clients except

Client 0

. . . . . .

CLOCK: (11.5s,2)

Client 1

Receives "Hello!" from Client 0

. . . . . .

管理器产生输出，它可以访问通信过程中的所有信息。ID为0的客户在时刻10.5发送了一条消息。根据协议，1号聊天室在一秒钟后把该消息广播出去。另一个连接在1号聊天室的客户1在同一时刻收到消息。这里也可以看出消息的最初发送者。

6.2 与时序图的一致性

与时序图的一致性可以通过一套基于规则的方法检查。这些规则在文本文件中定义。检查程序读取文件，检查输出迹是否可以满足所有的规则。正则表达式被扩展之后用以描述规则。规则由四部分组成：前置条件、后置条件、监视哨（任选）和计数规则属性（任选）。前置条件是正则表达式，用来匹配部分输出迹。它与监视哨（一个布尔表达式）结合，定义何时可以使用规则。当规则可用，并且计数规则属性为false时，后置条件（另一个正则表达式）必须要在输出迹中找到；如果计数规则属性为true，后置条件一定不可满足。

例如，下面的规则就说明了消息的发送者并没有在一秒钟后收到广播。

前置条件	CLOCK: \((\d+\.{0,1}\d*)s,(\d+\.{0,1}\d*)\)\n\Client (\d+)\nSays "(.*?)" to ChatRoom (\d+)\n
后置条件	CLOCK: \([(\1+1)]s,(\d+\.{0,1}\d*)\)\nClient [(\3)]\n Receives"[(\4)]" from Client [(\3)]\n
监视哨	[(\1+1)]<50
计数规则	True

在前置条件中，在括号中定义了五个表达式分组。分别编号从1到5。组1匹配浮点时间。组2匹配序号。它们组成时间元组。组3匹配以整数标记的发送者的客户ID。组4匹配消息，它可以是任意的字符串。组5匹配发送者所在的聊天室。

在后置条件中，[(…)]包含一个表达式，分组值可以在 “\”后使用索引号来引用。这样，[(\1+1)]是第一个组的值加一。[(\3)]等于组3。关于正则表达式的更多内容可以在[5]中找到。

假定执行过程停止在仿真时刻50那里，检查就不应该超过时刻50。在没有额外条件的情况下，如果在时刻49.5一条消息发送到聊天室，检查程序会希望在时刻50.5时见到相应的广播。为了实现这项功能，引入了一个监视哨[(\1+1)]<50。它会告知检查程序这条规则仅在分组1的值加一小于50时可用。

客户不应该接收到它自己的消息，这就是一条计数规则。

6.3 与协议的一致性

验证模型与协议完全一致如果说是并非不可能的话，至少也是极为困难的。协议，也可以看成需求集合，是使用自然语言描述的。准确解释它是自动检查程序开发的主要障碍。

你可能会争辩说协议可以转换为一套规则。使用上面描述的基于规则的方法，协议一致性就可以检查。但是，把协议的完整含义转化为程序易于处理的形式化表达是非常困难的。协议中含有的明显事实和常识常常会丢失。作为人与程序之间的接口，自然语言处理技术是有必要的。

在这个案例中，采用了一系列步骤来达成最终的可执行设计。在把协议转换为另一种不同的形式化表达体系时，信息就会丢失。中间步骤检查并不能保证最终产品的正确性。

另一方面，检查中间步骤并不足够有效。再者说，根据初始协议直接检查模型是极端困难的事情。在这个案例中，“如何验证最终设计的正确性”是最后的，同时也是最大的公开问题。

7 结论

在这个案例研究中，我们对一个具体的例子进行了讨论。从初始需求开始，我们开发了一个可执行程序。此间经历一些步骤，在不同的抽象层次上进行设计。我们挑选了一种基于组件的方法来把模型模块化，并使模型可以管理。类图定义了组件的接口。尽管只是部分阐述了需求，时序图形式化通信，使得自动检查变为可能。在扩展状态图形式化体系中，我们建模了基于组件的模型，这个模型由SVM执行环境直接转换。开发聊天室模型会引起一系列一致性问题。对于其中一部分内容，可以成功运用自动检查功能。

1. 时序图与类图间的一致性得到了检查。检查程序验证所有的必需方法在接口中都已正确地确定下来。

2. 状态图与类图间的一致性也可按照同样的方法检查。事件的发送者总能为接收者提供足够的参数。

3. 状态图与时序图间的一致性使用基于规则的检查程序来检查。正则表达式被扩展用来确定前置条件、后置条件、监视哨和计数规则属性。

然而，还有一些一致性问题仍然没有解决。

1. 类图与协议（初始需求）之间的一致性没有检查。在后续步骤中会发现设计缺陷，或者缺陷也可能会隐藏在最终产品里面。

2. 时序图与协议之间的一致性仅仅做了手工检查。尽管时序图只是协议的形式化方法，编写一个程序检查它的正确性也并不容易。

3. 检查扩展状态图中的最终设计和协议之间的一致性，要更为困难。这种检查是必要的，但信息在中间步骤有丢失现象。

应该把注意力放在这些开放问题上，它们大多与系统之间的一致性有关。一致性检查应该是开发过程中和软件开发工具的一个完整的组成部分。

参考文献

[1] D. Harel and R. Marelly. Specifying and executing behavioral requirements: The play-in/play-out approach. Technical Report MSC01-15, The Weizmann Institute of Science, 2001.

[2] Thomas Feng. An extended semantics for a Statechart Virtual Machine. In A. Bruzzone and hamed Itmi, editors, *Summer Computer Simulation Conference. Student Workshop*, pages S147 – S166. The Society for Computer Modelling and Simulation, July 2003. Montr´eal, Canada.

[3] Thomas Feng. Statechart Virtual Machine (SVM), 2003. MSDL, McGill University, http://moncs.cs.mcgill.ca/people/tfeng/?research=svm.

[4] David Harel and Amnon Naamad. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.

[5] Python 2.2.3 documentation, May 2003. http://www.python.org/doc/2.2.3/.

[6] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[7] Michael von der Beeck. A structured operational semantics for UML statecharts. *Software and Systems Modeling*, 1(2), 2002.

[8] Jim Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1998.

[1]UML 状态图（UML 2.0 以前的版本）在后面步骤中会用到，它不允许确定事件的接收者，即使只有一个聊天室会处理事件，所有的聊天室也同样都会收到相同的请求。

[2]为了增强表达能力，可以引入某些扩展，但这样的话检查模型的正确性将变得更加困难。

[3]这里[PARAM][0]的指的是事件请求的第一个参数，它可以由 ChatRoom 组件来处理。这个参数会进一步在事件 maccept 中传递，它是后者的第一个参数。

The Addison-Wesley Signature Series

Patterns of Enterprise Application Architecture

中译本即将上市

Martin Fowler
With Contributions by
David Rice,
Matthew Foemmel,
Edward Hieatt,
Robert Mee, and
Randy Stafford

UMLChina 负责中译本最后审校，并指定为训练用教材

使用基于组件开发和 Web 技术来支持分布式数据管理系统

M. BRIAN BLAKE、GAIL HAMILTON、JEFFREY HOYT 著，michael 译

查看评论

摘要

近年来，具有Internet访问能力的应用程序（即以WEB方式运行的程序－译者注）已经被用来支持必须具有分布式功能的领域。这一特性也与应用数据来自于分布于各地的涉众这种情形有关。在这种分布式的数据管理场景中，还存在许多重大的问题有待解决。问题之一是如何处理发生在这些领域中的变更。随着时间的过去，数据的模式（schema）和涉众的个体需要都会发生变化。任何支持这一分布式数据管理领域的架构都必须进行精心设计以支持这些特定的变更。解决这一架构设计问题的方法之一是使用具有“可插拔能力”（plug-able）的基于WEB的组件。当新的计算需要出现时，新组件可以被插接到该架构中。采用这一解决方案的架构的另一方面是希望构建出一个运行时可进化的系统以支持数据模式的变更。在MITRE公司，具有这一能力的架构已经被设计出来了，并已经被部署和测试以支持合成数据仓库的内部需要。本文描述了这一分布式数据管理系统的动机和架构，该系统被用来支持高级航空系统开发中心（CAASD）。这一基于组件的运行时可配置架构是采用基于WEB的技术来实现的，诸如XML，Java Servlets和关系型数据库管理系统（RDBMS）。

1 引言：为什么要采用分布式数据管理系统？

随着近年来计算系统的进步， 社会已经全面拥抱了信息技术，几乎每一个领域都变得依赖于信息资源，领域范围从金融领域、到工程领域、到医疗服务领域，拓展到更多需要处理当前数据并且要求高数据准确性的领域。随着分布式计算的增长，这一数据也必须可以从跨越多个企业位置和地理区域的地方访问到。此时，你可能会自问：“为什么不使用数据库管理系统呢？”。确实如此，数据库管理系统（DBMS）具有很强的对原始数据进行存储、管理、查询和并发处理的能力。数据库管理系统也允许数据从分布式的位置进行访问。不幸的是，数据库管理系统所提供的支持能力还不足以支持具有复杂的业务规则且相关处理必须在对企业范围内的涉众可用之前就进行的领域。在此情形下，我们提出了分布式数据管理系统的解决方案，该解决方案将数据库管理系统所具有的优势和基于组件的支持运行时可进化的架构所具有的弹性结合到一起。

1.1 数据库管理系统（DBMS）解决方案

当前，具备WEB访问能力的数据库已经不是什么新鲜事了，大多数数据库都具有工具和架构来支持基于WEB的图形用户界面（该界面可以访问该数据库）的开发，如Oracle公司2001年发布的的Oracle’s WebDB。图1描述了这一架构。

在图1中，DBMS将提供一个界面生成工具，该工具为用户提供了生成他们自己的GUI的选项。此工具对当前的数据库schema信息进行了描述，从而用户可以根据他（她）的需要来配置GUI。这是一个可伸缩的解决方案，因为当数据库的schema改变时，用户可以开发出新的GUI来支持这些改变。此DBMS也允许开发用于派生原始数据的数据库函数或包。这些函数在数据库服务器的内部运行。此外，该系统支持返回多种数据格式（如HTML、 Delimited text、XML等)。

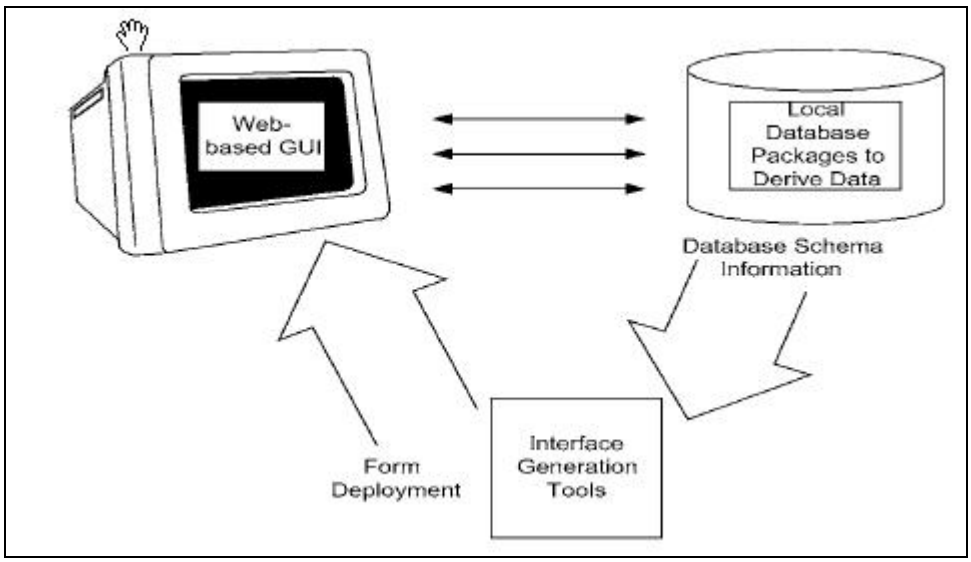


图1 支持WEB访问的DBMS架构

1.2 此 DBMS 解决方案存在的问题

在MITRE Corporation-Center的高级航空系统开发(CAASD)中，我们找到了为什么这一架构虽然可扩充，却不能满足我们的分布式数据管理需要的原因。在使用这一DBMS解决方案时，我们体验到了以下局限：

1 内部数据库包和函数不能支持苛刻的数据派生。

某些数据请求需要处理大量的信息，且必须经过一个繁复的处理才能由原始数据派生出更多有用的信息。使用包来处理这些派生过程，将会在多个用户同时请求大的数据库作业时大大降低DBMS的性能。

2 涉众需要其他“定制”的输出格式。

某些企业涉众需要以他们特定的应用程序格式来输出。虽然使用标准数据描述方法的概念相信所有人都会同意，然而当前却不很现实。我们的领域必须支持过去，现在和未来的数据格式，无论他们是标准格式（如HTML、XML、带分隔符的文本文件等）还是用户特定的定制格式。

3 某些数据请求需要动态生成多个查询（由于引入了业务规则）。

某些用户需要的查询没有外部的干涉将无法生成，因此，在此需要人的介入来提供关键信息才能完成该查询。

4. 某些用户应用程序需要直接连接到数据库。

某些应用程序需要直接将信息“流入”到它们的应用程序中。这一“流动”过程依客户的应用程序或仿真不同而不同。当前的DBMS解决方案并不支持数据“流动”的需要。此外，“流动”的功能也可能会引起额外的性能下降。

1.3 使用分布式的数据管理系统

在此，我们已经设计、实现和部署了一个分布式的数据管理系统。此系统使用当前流行的基于组件的架构来满足用户不断发展的需要。此系统也吸收了前面的DBMS解决方案的优势。图2演示了这一系统的高层次视图。界面层（Interface Layer）包含了用户用来指定GUI/query的工具（如同DBMS解决方案），但在这里用户也可以指定专门的必须在查询字符串创建之前执行的基于组件的函数。中间层（Middle Layer）包含了根据来自“web-based GUI”的指令创建工作流的专用组件，这一工作流控制组件使用创建出来的工作流来顺序地执行所需要的业务组件。如前所述，存在一序列独立的业务组件用来处理某些特殊功能，如生成动态查询、合成业务规则、生成定制输出格式、以及将数据流向某个想定制的外部应用程序等。数据存储层（Data Storage Layer）则基本上就是DBMS，可能会包含一些内部包，用来完成更多基本的格式化功能。

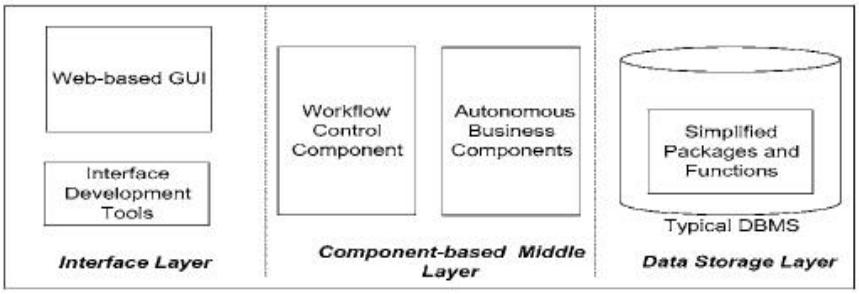


图2 数据管理系统的高层次架构

在MITRE Corporation-Center的高级航空系统的开发(CAASD)中，研究员为设计时和运行时的分析开发了仿真程序，这一研究使他们在空中交通管制领域获得了大量的知识。这些研究员被分成了多个不同的团队，他们分别研究空中交通管制领域的不同问题。虽然这些团队分别分析不同的问题，然而用来支持这些研究的数据却是同一个。同时，这些开发仿真系统的各个团队所需数据的格式是不一样的（如带限定符的专用文本文件、数据库格式、带定制模式（schema）的XML数据等）。而且，每个团队可能需要查看不同的数据子集，而这些数据可能跨越多个数据源。当前，研究员所需要的数据是由数据管理员首先从外部源头收集过来，然后进行分配。这些数据通常是以与获取它们时相同的介质和格式进行分配。MITRE的CAASD知识库系统（CAASD Repository System (CRS)）团队发现需要进行这么一项工作：从外部数据源获取需要的数据并组成一个数据仓库，以满足这些不同的用户群体的需要。 [Blake 2000; Blake and Liguori 2001a, b].本文所描述的架构就是实现用来满足此问题域的需要。

1.4 相关研究

除了有前述的关系型数据库应用所提供的初步的工具支持外，还存在几个相关的研究项目也是致力于这一目标。Zelig项目提出了一个模式(schema)可以通过使用HTML来控制不同的基于CGI的数据库可执行程序[Varela and Hayes 1994]。主要区别是Zelig项目仍然是将接口紧耦合到数据库，因为它将全部查询字符串硬编码到了HTML文档中。CRS架构通过在中间层组件中加入特殊的查询构建能力对这一方法进行了扩充。接口只具有查询组件，而中间层的组件则具有构建查询字符串的“智能”。此外，CRS架构还具有弹性结构，允许插入附加的特定业务的组件以插入更多的格式以及处理数据库结果。而且，CRS的实现使用了最新的更具弹性的技术，如XML（[Weiss 1999; Glushko et al. 1999]）和Java Servlets（[Sun Microsystems 2002]）。

另一个更接近CRS架构的项目是WebInTool（[Hu et al. 1996; WebInTool 1997]）。在 WebInTool中，Hu [1996]详细说明了一个网络到数据库的接口构建工具。Hu [1996]使用了与CRS架构类似的方法来将接口和后端的源代码分开，但他使用了与Zelig项目类似的多个基于CGI的模块，这些模块不具有智能的查询构建能力。正因为缺乏这一特性，使得WebInTool无法具备与CRS同样的能力来实现动态创建大范围的查询。此外，由于CRS架构使用了最新的技术，因而能够支持更大范围的输出格式选择。最后，由于查询字符串不在界面层中，使得理解了数据库的用户可以在不熟悉中间层组件的源代码的情况下就能构建新窗体；而WebInTool的用户则必须同时具有软件（CGI）模块和后台的数据库模式（schema）的知识。

本文将在随后的段落2到段落4对上述的各个架构层进行详细描述。在第五段落中，将描述和说明这一架构在MITRE公司的实现。在本文的最后部分，将根据非功能信息得出最后的结论。

2 界面层

界面层（Interface Layer）包含一个独立的模块，允许用户对它使用的数据查询指定参数和特别的处理。这一模块可以通过Internet进行访问，它使用基于servlet的技术来实现。当用户访问这一模块时，某个servlet将为创建GUI查询窗体提供多个选项，这个servlet有权访问数据库，因此，它所提供给用户的选项包含了数据库特定的信息。一旦用户完成指定他（她）的选项后，他(她)所作出的选择就可以用于为他（她）创建HTML文件，所创建的HTML文件中包含了一个隐藏标记，用来指定执行面向业务的组件。该隐藏标记使用了可扩充标记语言（XML）来描述指令。窗体上的其他字段也具有隐藏信息，用来将基于HTML的字段与数据库列关联起来。这一模块的主要功能就是用来收集这一信息，并将此信息与数据库的特定信息结合起来创建HTML文件。

2.1 HTML 隐藏标记与指令字段

这一GUI查询窗体必须既包含描述GUI组件的信息，也包含描述这些组件是如何与数据库相关联的信息。这一信息可以使用名/值对的形式来提供，这对于基于表格的HTML页面来说是与生俱来的。

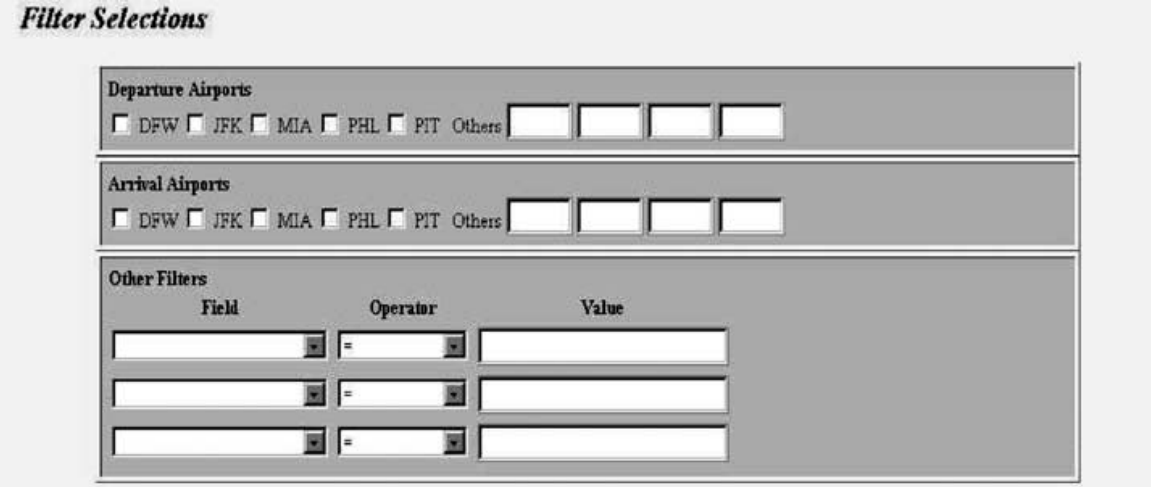


图3 界面窗体的屏幕快照

下面的HTML语句就显示了一个复选框的例子，这个复选框与某个数据库字段建立了关联。这一HTML代码实现了图3中的“PIT”复选框：

```
<input type="checkbox" name="Filter1"
  value="Table:OooiStageData~Field:deptAirport~Value:PIT">
```

这一复选框将被用作为数据的简单过滤器，因此我们需要一个过滤器（Filter）组件。字段的名称仅包含要使用的组件的类型，字段的值包含用来描述后台表信息的元数据，字符串被两个字符所隔开。名/值对之间通过“～”来分开；名与值则通过“:”隔开。这里仅仅是一个例子，这些分隔符可以被修改以符合系统的需要。“名称”包含了元数据（Table），“值”则包含了数据库的实际名称（OooiStageData）。这些“名/值对“可以被串连到一起，以包含实现将窗体上的字段和数据库关联起来所需要的所有信息。在大多数情况下，窗体上会存在多个过滤器（Filter）组件，因而在名称后添加了一个数字“1”，这样就可以让系统区分具有相同功能的组件了。此外，如果需要，它也允许系统将窗体上的不同字段进行关联，并一起处理，比如当用户需要限定大于某个字段或小于某个字段的信息时。具体到图3，举例来说，就是用户具有查询选项来指定字段、操作符和值。这些HTML字段的每一个具有不同的“名/值对”，这些“名/值对”必须被连接到一起。

当前，还存在一种不能使用“value”标记的情况，就是当用户手工输入值时。在此情况下，将具有一个隐藏标记用来包含元数据，可见字段和非可见字段将具有相同的名称（如都具有名称“Filter3”），系统能够处理这两个字段，就好像它们的信息包含在一个单一的HTML语句中一样。下面就是这种情形下的HTML代码例子，它实现了图3中的用户输入的文本框（“Other”）：

```
<input type="text" name="Filter3">"Other"
<input type="hidden" name="Filter3"
  value="Table:OooiStageData~Field:deptAirport~Operator:=>
```

限于篇幅，还有很多其他的GUI类型和数据库特定的函数无法一一展示。上述这些例子应该已经通过使用嵌入在HTML代码中的元数据说明了系统所具有的某些弹性。

2.2 面向业务的 HTML 隐藏标记指令

另一个隐藏标记用来告诉系统如何支持特定的功能。下面的HTML代码就是一个隐藏字段，用来告诉如何处理来自HTML窗体的信息：

```
<input type="hidden" name="workflow" value="Building SQL">
```

上面的HTML代码显示了一个隐藏标记，该标记指定了为了完成这一特定的窗体需要被调用的组件工作流。这些组件的设计将在第三段落中讨论。从本质上来说，将名称起为“workflow”就已经将这个隐藏标记与其他隐藏标记区分开了。每个窗体应该只有一个工作流标记，至少在系统开发的当前阶段是这样。工作流标记的值指定了要使用的特定工作流路径。当工作流控制组件（servlet）被初始化时，所有可能的工作流路径将被装载到系统中，这些工作流路径被包含在多个XML文件中，使用XML格式的目的是希望系统具有弹性，可以支持未来的外部应用程

序能够通过插入它们自己的XML指令来控制整个架构。这一特性主要用于对架构相当熟悉的有经验的用户，这些基于工作流的XML文件被放置在架构控制下的某个目录中。图4中的代码说明了与上面的HTML代码（如Building SQL）相对应的工作流。

图4中说明的工作流路径就是路径“Building SQL”，这是对于仅请求了一个标准查询的简单窗体的典型路径。第一步是解析来自发起工作流的HTML页面的信息，它由“StagedParser”对象来完成。第二步是根据从该窗体上收集到的信息构建一个一般的查询，它由“QueryBuilder” 对象来执行。最后一个对象是“OutputManager”，它将查询结果格式化为用户在该窗体中指定的格式。象上面的代码段中，工作流可以在方法级别指定指令，然而这并不是最典型的情况，因为大多数组件都带有默认的功能。不管如何，本例中的方法级别的指令说明了“QueryBuilder”对象将构建结果查询的特定语句（Select, From,Where等）。这些方法级别的指令展示了当缺省功能不适合时此架构所具有的弹性：

```
<WorkFlow>
  <WfPath>
    <name>Building SQL</name>
    <WfObject>
      <name>StagedParser</name>
      <Order>1</Order>
    </WfObject>
    <WfObject>
      <name>QueryBuilder</name>
      <Order>2</Order>
      <WfMethod Order="3">BuildFrom</WfMethod>
      <WfMethod Order="2">BuildWhere</WfMethod>
      <WfMethod Order="1">BuildSelect</WfMethod>
        <Parameter>
          <Type>Number</Type>
          <Value>10</Value>
        </Parameter>
        <Parameter>
          <Type>String</Type>
          <Value>"Select Distinct "</Value>
        </Parameter>
      </WfMethod>
    </WfObject>
    <WfObject>
      <name>OutputManager</name>
      <Order>3</Order>
    </WfObject>
  </WfPath>
</WorkFlow>
```

图4 XML文件中的工作流路径

2.3 创建新的 GUI 窗体

创建新的HTML窗体的过程是一个用户驱动的流程。用户首先进入到一个一般的窗体，在询问一些问题后，最终追溯到数据库中用户最感兴趣的表，然后用户就能够确定要过滤掉什么值、过滤掉什么时候的数据，进行排序、分组等，这些结果会在要创建的窗体上显示出来。这些被选择的与工作流相应的对象必须被执行以执行这些任务。所有这些信息都是用来创建一个HTML文件，这个文件中带有关于数据库元数据的隐藏信息，以及关于用来执行查询的特定工作流的隐藏信息。图5说明了这一过程：

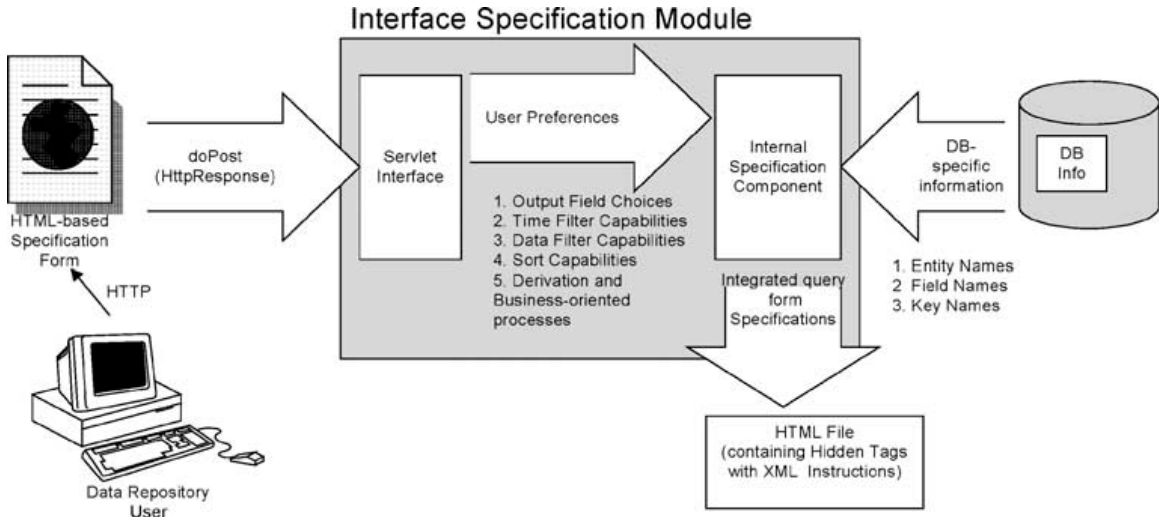


图5 构建“进行查询窗体”的流程

3 中间层

中间层是此架构的枢纽，该层遵循了严格的面向对象原则，所有设计都是使用UML（[Booch, Rumbaugh and Jacobson 1998]）来进行的。中间层包含了所有用来执行复杂的业务规则、创建用户特定的查询和执行复杂的数据格式化的对象，它可以被分成两个部分：工作流控制组件（Workflow Control component）和一组独立的业务组件。当前，这些部分通过5个组件来实现，它们分别是：MainControl、AdvanceDBManipulation、Parser_ObjectBuilder、Specialized和Statistical components.。在本架构中，组件由一组联合起来执行某个独立功能的类所描述。为清楚起见，这些组件以包的形式描述（与UML中定义的包相同）。这五个包（组件）及其所属的类在图6中描述。

3.1 工作流控制组件

工作流控制组件是此架构中用来控制在构建数据库查询和返回相关信息的过程中将要执行什么步骤的部分。工作流在过去的十年中，已经在多个上下文中定义过，在本文中，工作流可以定义为一序列步骤的说明和执行，或者是用来完成特定工作的流程。在大多数系统中存在的一个问题是工作流可以包含预定义的业务规则，由于这些功能基本上是“硬编码”方式，使得系统不好扩展，因而使得所进行的大多数修订是给系统加入许多微小的改进。然而，我们这里所定义的工作流控制组件则是一个可扩充的组件，允许工作流可以如段落2.2说明的那样在运行时设计，这个组件然后再去执行在工作流中所指定的自治业务组件。

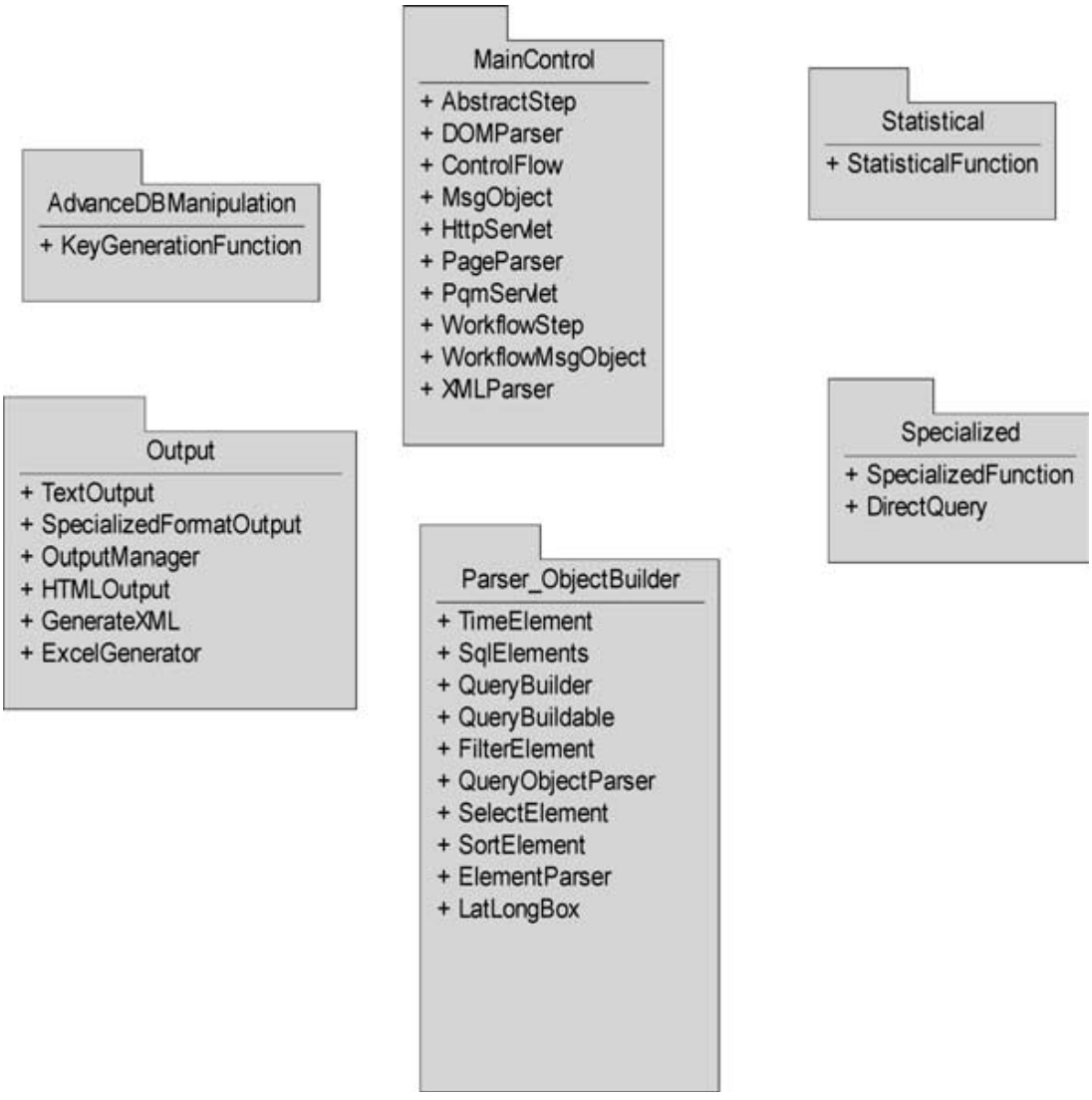


图6 中间层组件（包）的高层次视图

（此图以及随后的所有面向对象图都使用Rational公司的Rose Enterprise 2000所画）

3.1.1 MainControl包

工作流控制组件在“MainControl”包中实现，这个包中包含了可以按照预定义的工作流路径（见段落2.2中的XML文档）来组织、指挥和执行工作流的独立的类。这个包的主要类是：PqmServlet、ControlFlow、XMLParser和MsgObject。“PqmServlet”类（PQM代表 Presentation Query Management)）用于处理用户从HTML窗体上提交的请求信息；“ControlFlow” 类是主要的工作流执行引擎，由它去执行自治业务组件（将在下一段落中详细讨论）；“XMLParser”类用来解析工作流路径和捕获内存中的信息；最后，“MsgObject”具有通用的数据结构，用来存储在自治业务组件间传递的信息。由于这些业务组件是自治的（互相独立的），因而这一信息是工作流级别的普通信息。图7显示了带有上述类的“MainControl”包的类图：

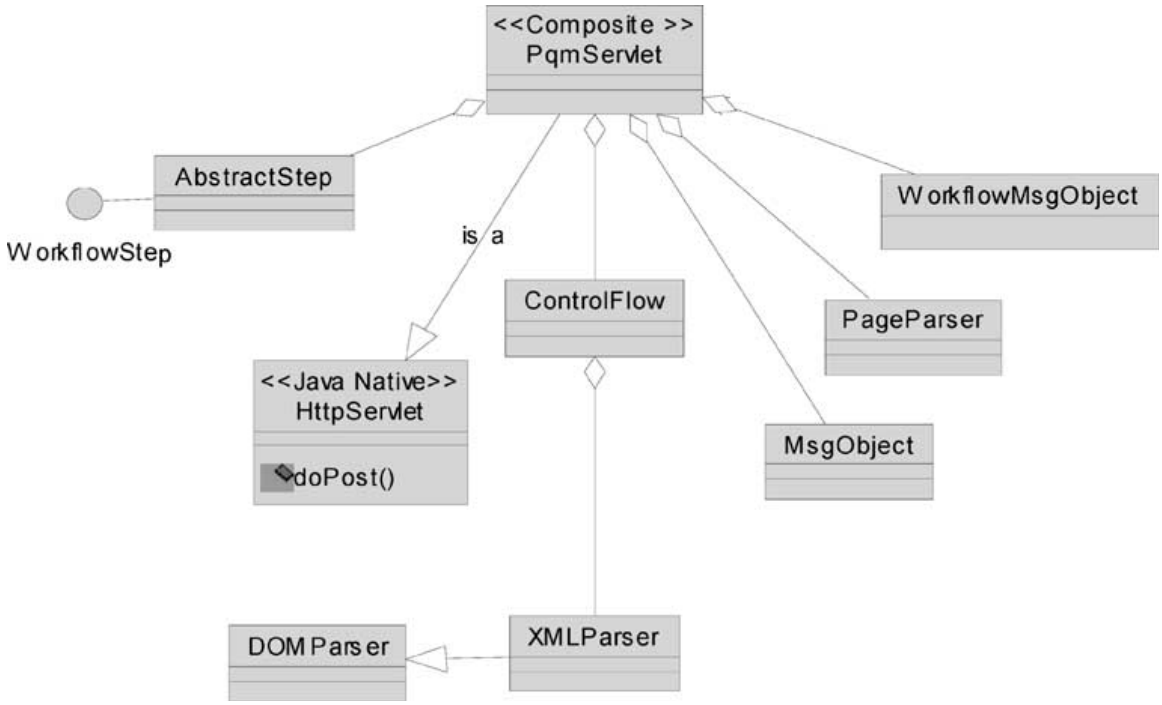


图7 “MainControl”包

工作流的执行流程可以概括为4个基本步骤：当用户在HTML页面上按下提交按钮时，PqmServlet类上的“doPost”方法被自动执行，PqmServlet使用用户的请求信息（通过JAVA的HTTPRequest）来生成“MsgObject”对象；随后，控制权移交到“ControlFlow”对象上，ControlFlow首先去解析基于XML的工作流路径；最后，ControlFlow对象使用来自XML文档的工作流信息和HTTPRequest来为每个自治业务组件创建和执行缺省或指定的入口方法。图8以协作图（在UML中定义）的方式说明了这一执行路径：

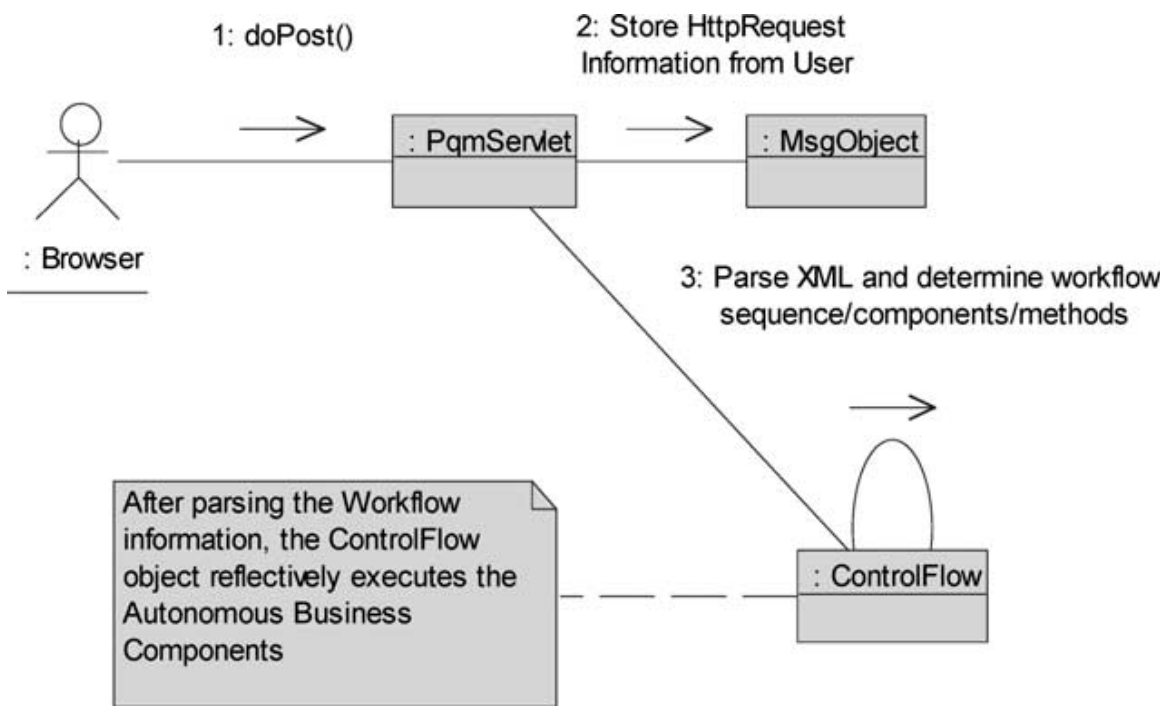


图8 MainControl包的功能概览

3.1.2 ControlFlow 类的反射能力

这一架构的一个重要方面是它能够在运行时执行新的工作流路径。反射性的架构或软件使得可以在运行时配置和执行而无须重新编译。

这一架构是一个运行时可进化的框架，它使用反射器来转换基于文本的指令为实际的软件执行方法。“ControlFlow”类中有一个简单的用Java代码写的方法用来执行被发射的方法。表1中的代码段说明了Java的反射器的使用：在表的代码中，ControlFlow对象使用MsgObject作为参数，这使得它成为了用来传递信息的自治业务组件的容器。使用MsgObject的目的是提供全局的工作流级别的信息，而无须耦合到任何特定的组件。因此，MsgObject类仅包含用来存储和检索全局信息的实现。“createObject”方法使用在基于XML的工作流路径中指定的“className”作为参数，然后反射性地创建该类的一个实例，这一实例用来多态地运行包含在所有自治业务组件中的“execute”函数。这只是这一架构利用面向对象编程好处的应用之一。

表1：发射代码语法片断

```
public class ControlFlow {

   private MsgObject messageObject = new MsgObject();
   public AbstractStep crsObject; // Parent class of all plug-able components

      /***Controller***/
      public ControlFlow(MsgObject thisMessage)
      {
       messageObject = thisMessage;
      }

      /***Method to create the Object****/
      public void createObject(String className) throws ControlFlowException
      {
      Object object = null;
      try {
            Class classDefinition = Class.forName(className);
            crsObject = classDefinition.newInstance();
            crsObject.execute(messageObject);
          }
      // (CatchExceptions. . .)
```

3.2 自治业务组件

自治业务组件在多个包中实现，这些包在图6中进行了说明，它们是：AdvanceDBManipulation、Output、Parser_ObjectBuilder、Statistical和Specialized包。这些包中的每一个针对数据提取作业实现了一个特定的自治功能。“AdvanceDBManipulation”用来从HTML文件中收集特定的表信息，并在需要用于某些数据查询的数据库表之间构建复杂的连接（join）。“Output”包用来将查询结果转换为用户指定的不同的查询输出格式。“Statistical”包具有一组内部算法，可以基于一定的算法规则派生出查询结果。“Specialized”包用来处理所有不能被一般化的功能，在这个包中实现的一个功能就是它能够使用户以SQL的形式直接输入数据库查询，并能以指定的格式返回。最后，“Parser_ObjectBuilder”包中包含了能从HTML中生成一个一般的查询的类，这个包接收请求信息，然后例行地构建一个标准的数据库查询字符串。详细说明所有这些自治业务组件已经超出了本文的范围，我们选择了对“Parser_ObjectBuilder”包中的类进行解释，因为这个包对于生成一般的SQL语句来说是必须的。

3.2.1 Parser_ObjectBuilder包

“Parser_ObjectBuilder”包是从数据库中显示数据的任何特定的工作流路径要用到的最常见处理的例子。

在此架构中，来自HTML查询窗体的独立的按钮和字段与将要构建的查询之间具有直接的关系。在这个包中，有多个类用来将HTML窗体上的字段与后台的数据库建立关联，这些类的职责是用来理解这些HTML字段并创建SQL查询，最典型的类有：SelectElement、FilterElement、SortElement、GroupElement和StatElement。这些类都是派生自SqlElement类，而“SqlElement”类则实现了Java接口“QueryBuildable”。通过实现“QueryBuildable”接口，使得“SqlElement”及其所有的子类都必须实现下面这些方法：buildSelect、buildFrom、 buildWhere、buildSort、buildGroupBy和buildOrderBy。从直观上看，这些方法将分别创建查询的各个部分，这些部分包括：Select、From、Where、Sort By和Group By块。这些类的功能可以总结如下：

- SelectElement：转换HTML字段为要显示的来自数据库的列。
- FilterElement：转换HTML字段为数据集的显示约束。
- SortElement：转换HTML字段为来自数据库的数据的显示次序。
- GroupElement：转换HTML字段为来自数据库的数据的显示分组。
- StatElement：转换HTML为统计功能，如Count()。

图9显示了“Parser_ObjectBuilder”包的类图，“QueryBuilder”类用来收集给定查询的所有“SqlElements”的列表，“QueryBuilder”类首先生成一个空的SQL字符串，当接口方法被SqlElements的每个子类所调用时就不断地填充了这一SQL字符串。这可能是面向对象的最强大的使用，它演示了信息隐藏、封装和多态的使用，因为每个SqlElement子类仅对它自身的行为负责。这些子类通过使用“build…”方法被多态地调用(这些方法都为通过“QueryBuildable”接口实现的，如图9的右上角所示)。

3.2.2 添加新的业务组件

如上所述，自治业务组件是互相独立的。每个组件都有一个接口类，包含一个“execute”方法，这个“execute”方法被“ControlFlow”类多态地调用。 使用这一设计方法，新的组件可以被无缝地添加进来，或者可以实现软件的即插即用（[Bronsard 1997]）。当一个新组件被添加进来时，它只需要从包含在“MainControl”包（这个包已经在图7中说明过）中的“AbstractStep”类继承即可，“AbstractStep”类具有接口“WorkflowStep”，而这个接口定义了“execute”方法。随后，当某个基于文本的指令在工作路径中调用了该组件时，则“execute”方法会被缺省调用，除非声明了有如段落2.2中讨论过的特定的功能。

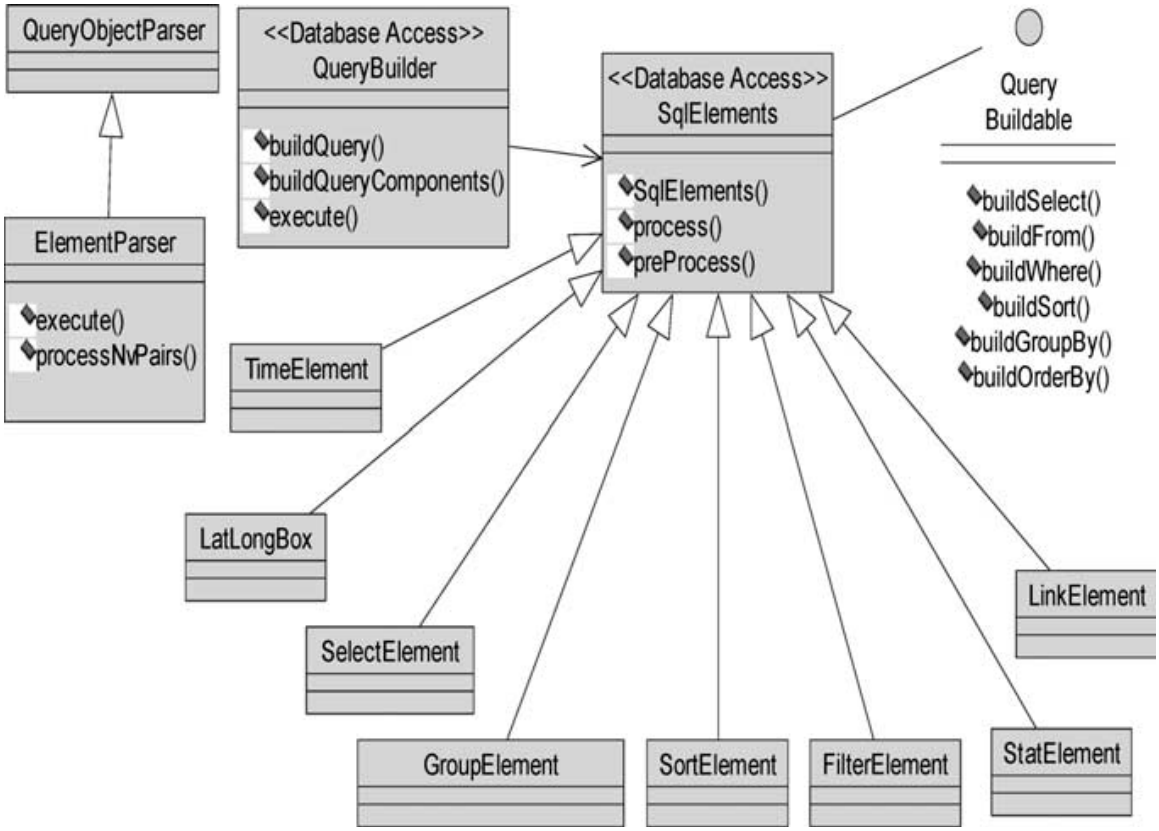


图9 Parser_ObjectBuilder 包的类图

4 数据存储层

数据层的做法就有些按照惯例了，在此架构中的数据由一个关系型数据库管理系统（RDBMS）进行管理。在此情况下，RDBMS采用的是Oracle 8i。这一架构并不需要数据被规范化，但某种程度的规范化有助于改善整个系统的性能。下面列出了我们的架构所利用到的数据库功能：

1 **存储过程，用来执行简单的格式化（日期、时间、宽度等等）。**某些格式化当使用数据库服务器时要比使用外部的基于Java的处理更容易完成。将来，可能会有更多的存储过程被添加进来以从代码中进一步删除掉某些普通的功能，比如附加的钻取（drill-down）特性。

2 **元数据，用来将数据库列名翻译为相应的英语名。**系统使用一个普通的查询流程，它直接使用列名来生成查询。在数据库中，存在的元数据表可以帮助此架构来将数据库的列名翻译为用户友好的名称。同时，此架构也对在HTML窗体中使用用户友好的名称提供了支持。

3 **为大表建立的索引。**与大多数RDBMS一样，此系统的性能也依赖于有一个正确的索引系统。索引的创建由数据库管理员执行，它通常在系统的非高峰使用时间进行创建和常规更新。

5 分布式数据管理系统在 MITRE 的实现

本节将对MITRE公司根据本文描述的架构所开发出来的应用程序进行讨论。下面的段落将对所开发的系统进行概述，描述查询构建的过程，并显示流程中用到的某些实际的窗体。

5.1 分布式数据管理系统的实现：CRS 系统

在MITRE-CAASD，CRS团队已经通过使用多方面的Internet技术实现了CRS架构。图10对这一支持上述架构图的详细实现进行了描述。CRS实现主要使用了基于Java的技术。其中的提取数据的三个基本步骤已经被分割到三个模块中：Control、Query和Output模块。本节的剩余部分将讨论这三个模块在本文描述的分布式数据管理架构中的作用。

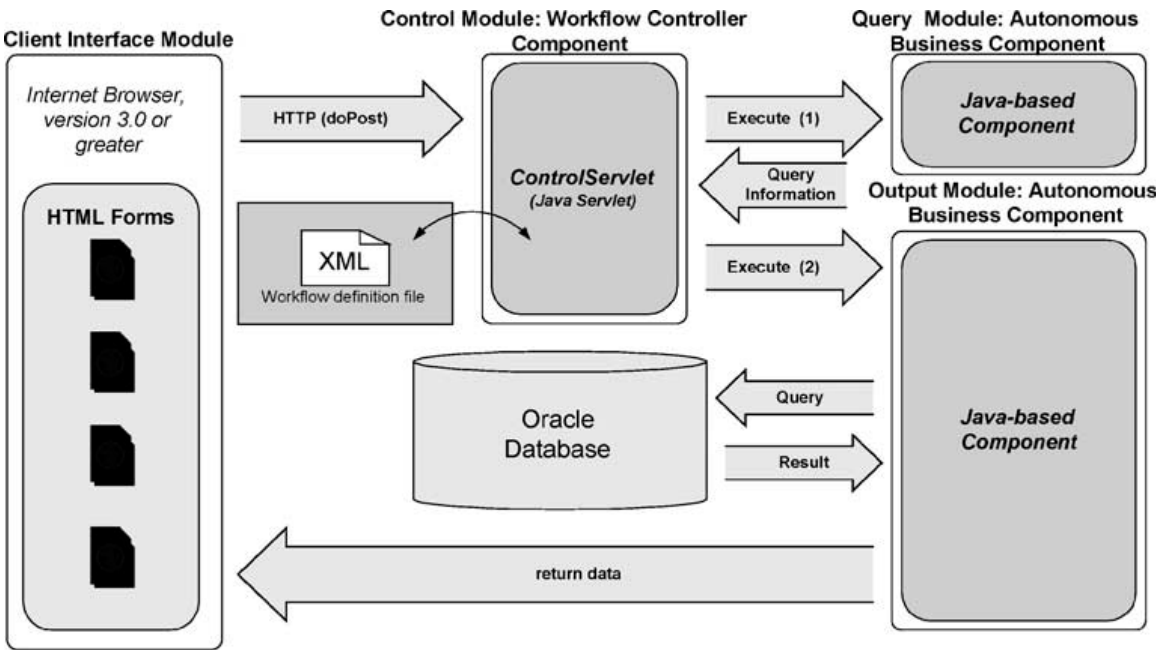


图10 CRS系统的架构

“客户端接口模块”（Client Interface Modules）中的浏览器连接到基于Java Servlet的接口（ControlServlet），ControlServlet会负责将信息传递到控制模块（这两者组成了工作流控制组件（Workflow Controller Component））。“ControlServlet”作为用户和控制模块间的主要接口，它以HttpServletRequest的形式接收来自客户端接口模块中的浏览器的信息。客户的数据和格式标准被封装在HttpServletRequest中。正如段落3.1.1中描述的那样，这一信息被解析和存在一个Java对象中—MsgObject。这个MsgObject对象基本上会依次传递到所有步骤中，每一个步骤会从MsgObject对象中提取它所需要的信息并修改它的内容（如果需要）。

下面的段落对分布式数据管理架构中与CRS实现相关的设计进行了低层次的讨论。

5.2 CRS 的模块描述

客户端接口模块（Client Interface Modules）就是试用CRS系统的涉众使用的网络浏览器。通常，涉众是基于他们感兴趣的数据集来访问标准查询窗体的。关于空中交通管制（ATM）的一个独特的数据模式（schema）是OOOI数据集，当某架飞机从终点返航、离开跑道、降落到目的地跑道、最后进入航空港时，这一数据集记录了全部的“Out/Off/On/In”时间。图11就显示了一个与CRS分布式数据管理系统的观念相一致的查询窗体。

这个窗体被分成五个部分：第一个部分为时间选择区（Time Selections），允许用户指定用来过滤输出信息的特定时间范围；第二部分为过滤选择区（Filter Selections），用户根据返回值指定过滤约束条件；输出选择区（Output Selections）允许用户来确定哪些列作为输出列表；排序选择区（Sort Selections）允许用户确定数据应该如何排序，这一选择需要被排序的列也在输出选择区中被选择；最后一个部分是输出大小和格式选择区（Output Size and Format Selections），它允许用户限制输出的数量，以及指定希望返回的格式。图中显示的情况是请求从2001年元月一日午夜开始、一个小时期间的有价值的信息，这一信息是基于航班返航的时间，你可以在时间选择区中看到这些选择。此外，此查询请求将只返回从达拉斯或匹兹堡起飞的航班信息（见过滤选择区）。

输出数据选择区显示了查询结果的每一行可以返回的有关飞机和机场的各方面信息。在排序选择区中，指定了结果将以出发机场来排序。最后，在输出大小和格式选择区中，限定了输出为5行，且将以HTML表格的形式返回。

控制模块（Control Module）是通过一个XML文件和几个Java类来实现的，它的主要用途是利用工作流控制信息来协调相应模块的执行。其中的XML文件包含了所有被选择的工作流模块的列表，并且以这些模块被执行的次序排列，然后控制模块会解析这个XML文件来得到相应的工作流。图4中就描述了“OOOI”数据集使用了这个XML文件（工作流）。

查询模块（Query Module）是一个自治业务组件，它是使用Java类实现的。如段落3.2.1中所述，SqlElement对象被创建用来创建一个SQL数据库查询字符串，随后，结果查询字符串被存储在MsgObject对象中，以便输出模块（Output Module）进行检索。上面的“OOOI”例子中的查询字符串描述如下：

```
SELECT DISTINCT OooiStageDa.carrierFlight, OooiStageDa.deptAirport, OooiStageDa.
   arrAirport, OooiStageDa.gmtdate, OooiStageDa.outtime, OooiDerivedTi.
   taxiouttime
FROM OooiStageData OooiStageDa, OooiDerivedTime OooiDerivedTi
WHERE ROWNUM < 6 AND OooiDerivedTi.OooiDataId=OooiStageDa.OooiDataId AND
   OooiDerivedTi.outtime >= to_date('2001-01-02 00:00:00', 'YYYY-MM-DD
   HH24:MI:SS') AND OooiDerivedTi.outtime <= to_date('2001-01-02 01:00:00',
   'YYYY-MM-DD HH24:MI:SS') AND (OooiStageDa. deptAirport= 'PIT' OR
   OooiStageDa.deptAirport= 'DFW')
ORDER BY OooiStageDa.deptAirport
```

输出模块（Output Module）是一个自治业务组件，它基本上是应该针对DBMS的定制接口，这并是什么新技术，因为许多供应商已经为DBMS提供了API。为了在CRS架构中使用，在此我们专门研究了其中之一。在这个研究的实现中，我们使用Sun公司的JDBC来连接到某个Oracle 数据库。所有的数据库和用户之间的交互都运行在输出模块中以改进性能。输出模块中的主要的类是“OutputManager”，它从MsgObject中提取查询，并将查询提交到DBMS，一旦返回的结果集可用，它就会以指定的格式来返回数据。这个类可以用来生成HTML、ASCII、XML、Excel以及其他任何期望的可定制的格式。图12对这一设计的思想进行了描述。输出模块是一个独立的查询构造器和数据库，因此无论查询条件怎么改变并不会影响它的功能。

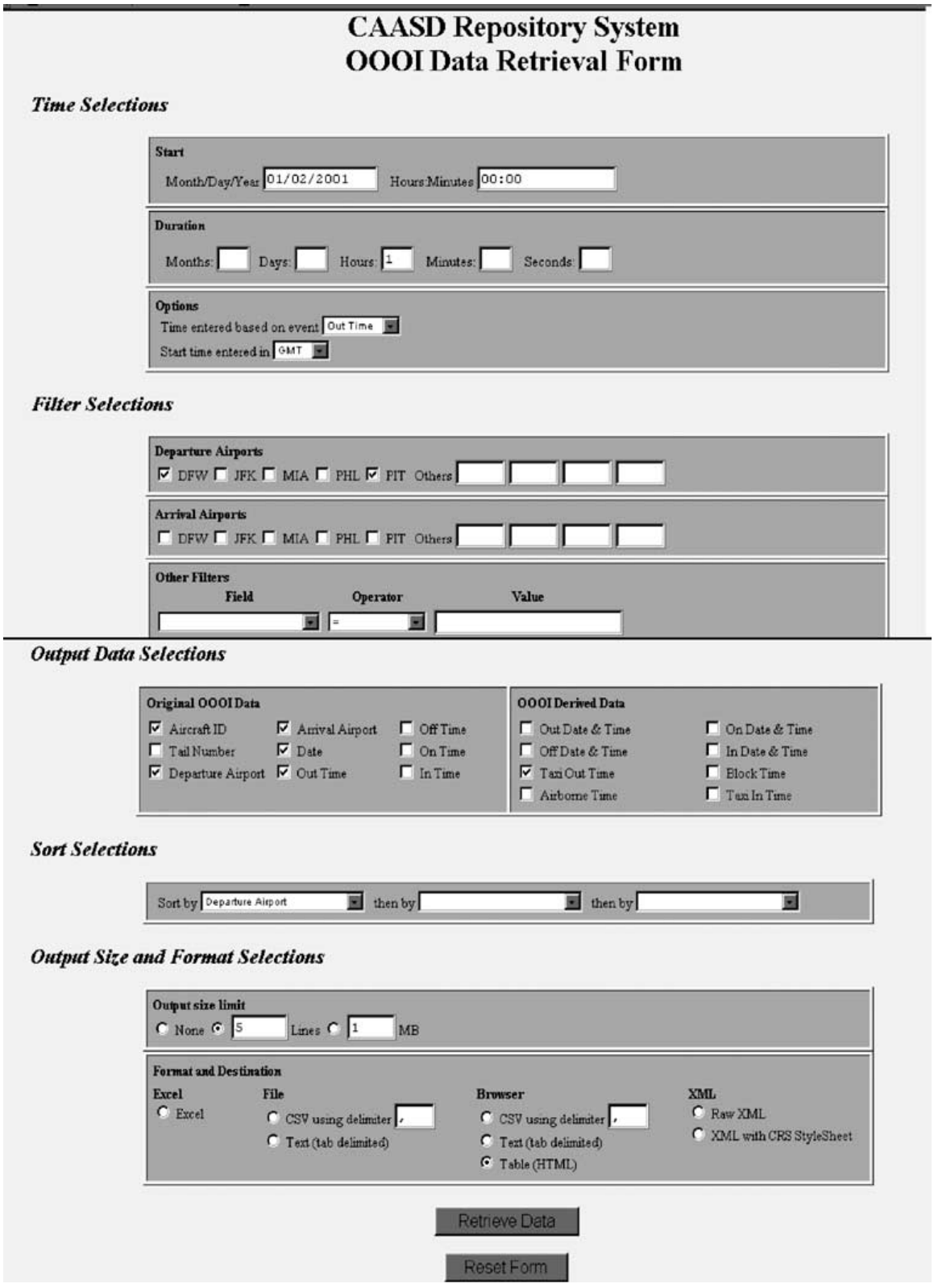


图11 OOOI数据集的范例HTML文件

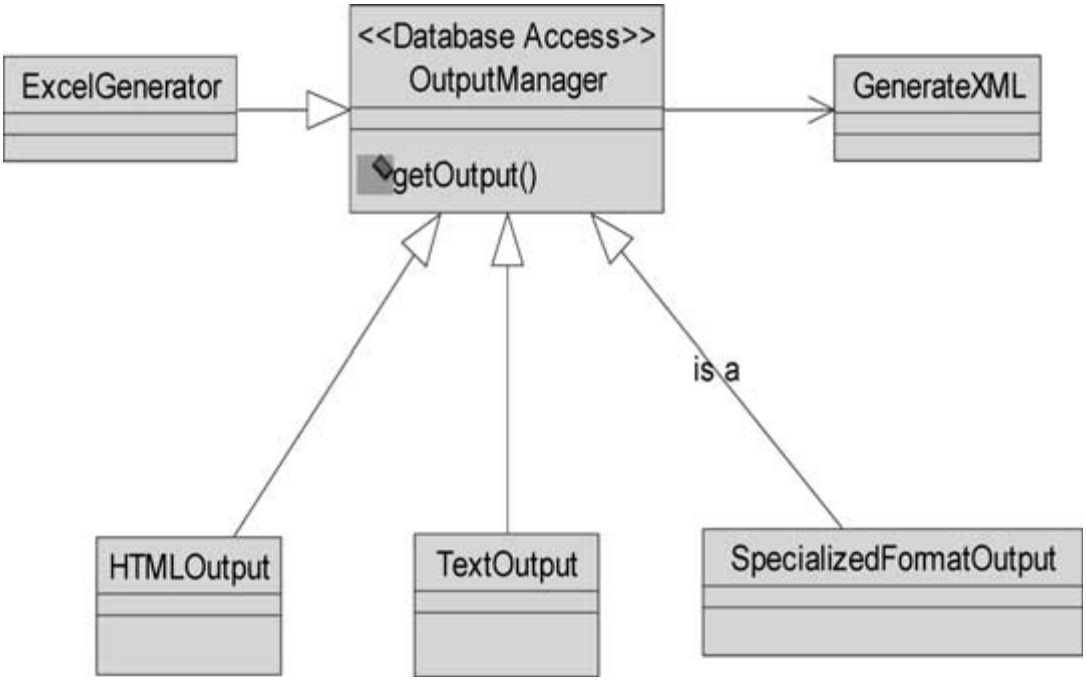


图12　输出模块的详细类图

5.3 数据提取过程

在图 13 中，我们用一个协作图来显示在创建查询字符串和返回结果时的事件序列。

下面是在图 13 的协作图中描述的软件执行事件的列表:

1 一开始，“doPost”事件被“ControlServlet”(也就是 PqmServlet)所捕获；

2 “ControlServlet”然后将“doPost”事件中传递过来的消息保存倒“MsgObject”中，以便供后续执行模块检索之用；

3 “ControlServlet“创建一个控制流（Control Flow）对象，这个对象确定了要用到哪些模块。（在本图中显示的情形中为将要用到标准查询（Query Module）和输出管理器（OutputManager）模块；

4 工作流的名称在上面的第二步就已经保存到“MsgObject”对象中了，这个名称被用来结合根据在 XML 文档中预先定义的工作流来确定要被调用的模块和类（也可以包含方法）及其执行次序；

5 在查询模块（Query Module）中，包含两个步骤：“ElementParser”和“QueryBuilder”。“ElementParser”首先被调用，它从存储在消息对象中的信息创建“SQLElements”对象，一旦所需要的每个元素都被创建了，这些信息将保存在“MsgObject”对象中以供“QueryBuilder”使用；

6 “QueryBuilder”会系统地调用每个“SQLElement”元素上的“buildSelect”，“buildFrom”，“buildWhere”和“buildSort”方法，这些方法将会真正去完成构建查询字符串的工作；

7 “OutputManager”然后使用缺省方法来检索并以正确的格式返回所请求的数据；

8 数据被返回到客户端浏览器中，或者供浏览或者保存以供后用。

5.4 OOOI 的查询结果

按照上面在查询窗体中指定的条件，结果为一个显示5行数据的HTML表格，每一行包含了从达拉斯机场或匹兹堡机场起飞的飞机的航线/机场数据，图14显示了这一结果。

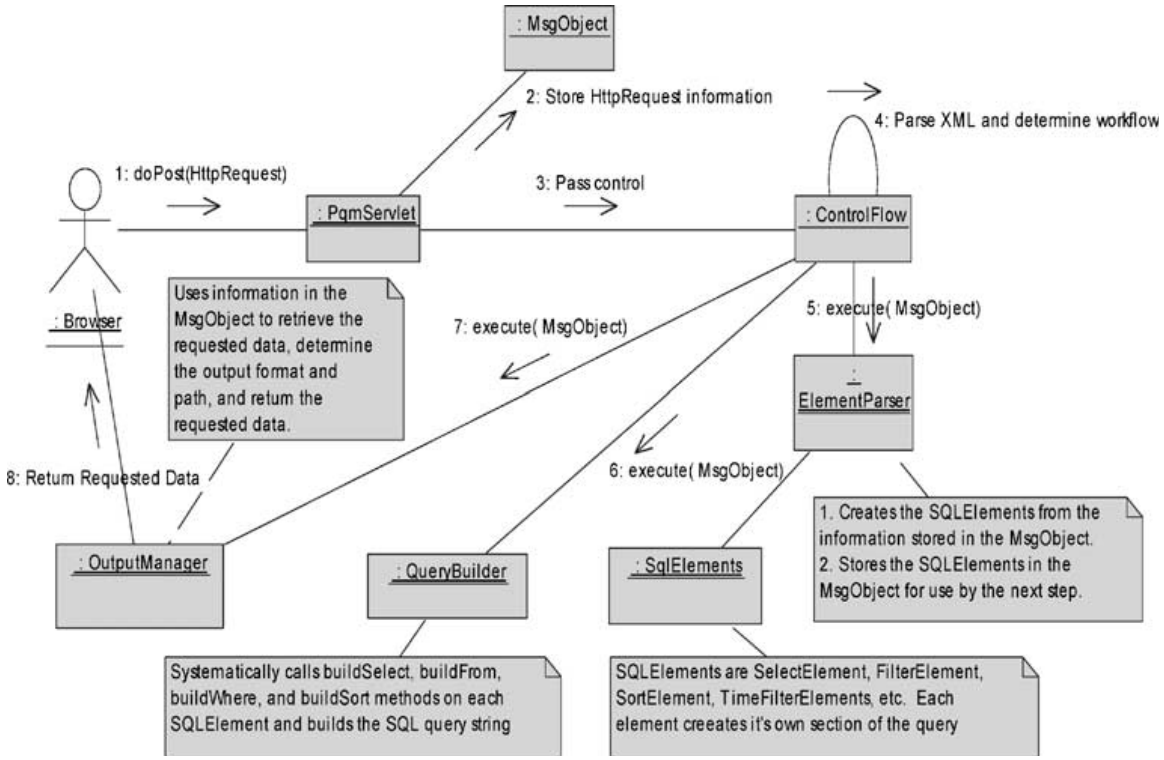


图13　数据提取过程的协作图

6 结论

在上面的过程中，我们已经展示了用来增强RDBMS的基础性而使用设计分布式数据管理系统的必要性。本文描述了一个使用面向对象的概念实现的架构。在这里采用了基于组件的架构以保证系统的运行时可进化，此外，系统充分利用了最新出现的WEB技术（(XML, Java Servlets, Java Reflection以及处于主流地位的RDBMS)。

CARRIER	ORGNLGATEDEPTTIME	ORGNLGATEARRTIME	DEPTAIRPORT	ARRAIRPORT
AAL	141	340	DFW	TUS
AAL	141	426	DFW	LGA
COA	2334	2542	DFW	CLE
UAL	0	139	DFW	DEN
USA	2318	2403	PIT	SYR

There were 5 records returned. The limit was 5 lines. There were approximately 377 bytes returned. There was no file size limit.

图14 OOOI查询结果

这一自治架构允许用户自行确定他们想要的数据以及数据返回的格式。应用程序然后会生成适当的查询，执行它，并返回结果数据。通过维护这一基于组件的模块的自治种类，这个架构可以在将来只须很少的服务功能损失就能改动。由于这些模块与数据库模式（schema）是不相关的，使得该系统只需要很少的改动就可以重用于任何数据模式。这一系统的用户只需要理解输出数据，格式，以及过滤的类型就可以进行操作。而且，当情况需要系统作出变更时，这个系统可以快速而容易地适合相应的改变。

这一架构已经在CAASD知识库系统上得以实现。CRS系统已经在MITRE-CAASD上得到了大量的成功使用，新的数据模式和功能被不断添加进来。这个架构已经非常强壮，当面对许多不同的数据模式时所需要做的修改已经比较小了。

当前，在提供特定的输出格式方面还有工作要做，以在普通的输出格式（带分隔符的文本文件、XML或HTML表格）的基础上提供更多的种类。未来的方法将允许CRS的用户能指定首选输出格式为XML文件，系统将使用这个文件来处理查询响应。

致谢

这一工作是与MITRE公司的CAASD合作的结果。我们获得了MITRE-CRS团队成员的大量支持，他们是Rob Tarakan、Fred Wieland、Tho Nguyen、Ted Cochrane、Jay Cheng、Ali Obaidi和Gretchen Jacobs。

参考文献

1 Allen, R.J., R. Douence, and D. Garlan (1998), "Specifying and Analyzing Dynamic Software Architectures," In *Proceedings of the 1998 Conference on Fundamental Approaches to Software Engineering (FASE98)*, Lisbon, Portugal, Springer, Berlin, pp. 21 - 37.

2 Blake, M.B. and P. Liguori (2001a), "An Autonomous Decentralized Architecture for Distributed Data Management and Dissemination," *IEICE/IEEE Joint Special Issue on Autonomous Decentralized Systems and Systems' Assurance, IEICE Transactions on Information and Systems*, E84-D *10*, 1394 - 1397.

3 Blake, M.B. and P. Liguori (2001b), "An Automated Client-Driven Approach to Data Extraction Using an Autonomous Decentralized Architecture," In *Proceedings of the 5th International Symposium of Autonomous Decentralized Systems (ISADS2001)*, IEEE Computer Society Press, Dallas, TX, pp. 311 - 319.

4 Blake, M.B. (2000), "SABLE: Agent Support to Consolidate Enterprise-Wide Data Oriented Simulations," In *Proceedings of the 4th International Conference on Autonomous Agents (AGENTS2000)*, *Workshop on Agents in Industry*, Barcelona, Spain.

5 Booch, G., J. Rumbaugh, and I. Jacobson (1998), *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA.

6 Bronsard, F. *et al.* (1997), "Toward Software Plug-and-Play," In *Proceedings of the 1997 Symposium on Software Reusability*, pp. 19 - 29.

7 Garlan, D. and M. Shaw (1992), *Software Architectures, Perspectives on an Emerging Discipline*, Prentice-Hall, Upper Saddlehill, NJ.

8 Glushko, R. *et al.* (1999), "An XML Framework for Agent-Based E-commerce," *Communications of the ACM 42*, 3, 106 - 107.

9 Hu, J., D. Nicholson, C. Mungall, A.L. Hillyard, and A.L. Archibald (1996), " WebInTool: A Generic Web to Database Interface Building Tool," In *Proceedings of the 7th International Conference and Workshop on Database and Expert System Applications (DEXA96)*, IEEE Computer Society Press, Zürich, Switzerland.

10 Oracle Corporation (2001), "WebDB Application 3.0," http://oradoc.photo.net/ora816/webdb.816/a77075/basics.htm.

11 Rational Corporation (2002), "Rational Rose Enterprise Edition," http://www.rational.com.

12 Shrivastava, S. and S.Wheater (1998), "Architectural Support for Dynamic Reconfiguration of Large Scale Distributed Applications," In *Proceedings of the 4th International Conference on Configurable Distributed Systems (CDS' 98)*, IEEE Computer Society Press, Annapolis, MD.

13 Sun Microsystems Inc. (2002), "Java Language Specification and the Distributed Event Model Specification," http://java.sun.com.

14 Varela, C.A. and C.C. Hayes (1994), " Zelig: Schema-Based Generation of Soft WWWDatabase Applications," In *Proceedings of the 1st International Conference of the World Wide Web (WWW94)*, Geneva,Switzerland, Elsevier Science.

15 WebInTool (1997), http://www.ri.bbsrc.ac.uk/webintool.html.

16 Weiss, A. (1999), "XML Gets down to Business," *Networker 3*, 3, 36－37.

斐力庇第斯从马拉松跑回雅典报信，虽然已是满身血迹、精疲力尽，但他知道：没有出现在雅典人民面前，前面的路程都是白费。

学到的知识如果不能最终【用】于您自己的项目之中，也同样是极大的浪费。而这最后一段路最是艰难。

UMLChina 聚焦最后一公里，所提供服务全部与您自己的项目密切结合，帮您走完最艰难的一段路。

一个课程管理分析模式

Xiaohong Yuan, Eduardo B. Fernandez 著, blackbo 译

查看评论

摘要

本文讨论一个课程管理的分析模式，该模式描述了诸如学生注册、增加和取消课程、成绩管理等事件。模式可以推广到相似的应用中。本文包括两个相似的模式:课程注册模式和成绩管理模式，这两个模式都有各自的参考价值，适用于不同的场合。

1 简介

课程管理是大学和培训机构的一个共同业务，同时也是远距离学习和基于 web 的教育项目的一个必需组成部分。本文将显示一个描述课程管理的基本方面的分析模式，该模式是我们称为语义分析模式的一个例子，因为它强调应用模型的语义方面而非灵活性。这种类型的一个模式实现了一些关键用例，也包括了一些组件模式。课程管理模式强调课程管理的基本方面，对它的扩展、例外和变化只做简要的讨论，这个模式可以作为开发课程管理应用的一个开始，需要根据具体的应用进行修改和补充。我们将在文章后面把课程注册和成绩管理整合到课程管理中。

2 课程注册

2.1 目的

该模式描述学生注册课程的过程，同时保持跟踪注册某门课程的所有学生以及某个学生已经注册了哪些课程。

2.2 上下文

提供任何类型课程的机构。

2.3 问题

学生参加一门课程的学习之前或者学习了一段时间之后，需要注册该课程。在规定的时间里，学生可以注册或者取消课程。注册课程涉及到安排教师和教学设备、收取学费。这些处理是相当复杂的，需要用一种方便和高效的方法。虽然存在各种各样的注册方式，但它们都有着许多相同的地方，我们需要发现一个可以描述这些共性的模型。

2.4 约束

• 当注册同一门课程的学生很多时，这是很难管理的，需要将每门课程分成更小的单元(美国的大学称之为Section)

• 无论学生选修课程或者教师讲授一门课程都可能存在计划冲突，或者存在住所离校园较远这样的限制。这就要求我们为它们制订最合适的计划。

• 组织成标准文档(如注册表)的数据应该可以在模型中直接表示。只有文档中的信息是重要的，它的外观或表现不重要。

• 为了提高学生在某些课程的表现，应该对学生选修该课程设置满足一定水平的前修知识的限制。

• 为便于管理和提高学生的表现，需要设置学习课程的期限。

2.5 解决方案

将一门课程分成多个小部分的课程(section)，称为课程班，每个课程班包括一小组学生，分别安排教师和教学时间。这样，一门课程可能包括多个编号不同的课程班，学生可以根据他们的需要和限制条件注册课程班，但他们所选修的课程还会有某些限制(如最大和最小注册人数)。学生以一种特定的登记表进行课程注册。选修某些课程时，可能会有前修课程的要求。以上的处理可以表示为下面的用例:

(1)课程登记

在允许的登记时间段，学生可以浏览可供选修的课程，并选择所要修的课程的课程班。同时，考虑到某门课程会因选课人数满额或因某种原因被取消，学生也可以选择一个候选的课程班。选好后，学生提交选课登记表，系统将检查是否存在满额或已被取消的课程，是否存在时间冲突，或者学生是否符合修该课程的前修知识要求。一旦学生的登记成功，系统将产生学生的课程时间表，以及所需付的学费。

(2)添加/取消课程

在指定的时间内，学生可以增加课程或者取消已登记的课程。同样，他需要以登记表的形式提交他所要增加或取消的课程，系统将产生新的课程时间表和学费。

图 1 是实现这些用例的类图。类“课程”提供了一个课程的描述，如名字、课程号、内容概要。类“课程”的前修要求反映了一门课程要求预备知识的事实，同一个学期里，一门课程可能包括几个课程班，类“课程班”和“学生”之间的关联**是否注册**描述了一个学生注册多门课程，以及多个学生注册同一门课程。关系类“计划表”描述了学生整个学期所注册的课程的时间表。每个学生提交一个或多个填写了所选课程列表的表格。类“证明书”记录了学生已修的课程，用于检查该学生是否符合他所选课程的前修要求。

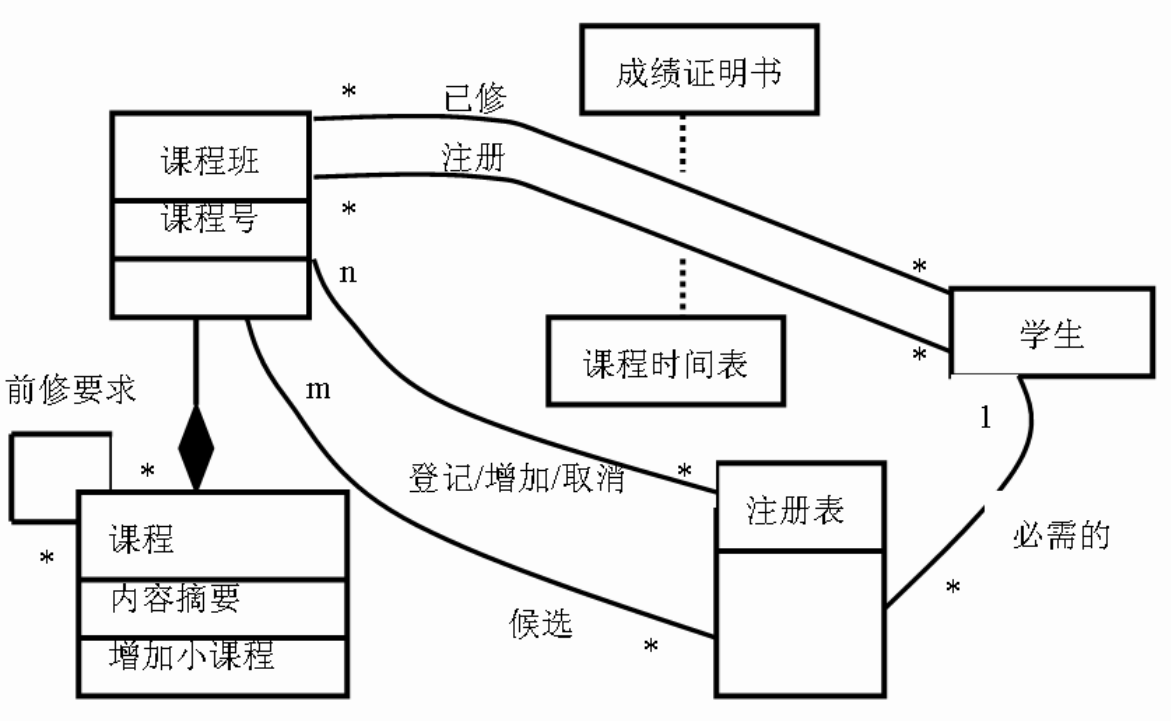


图 1　课程注册用例图

图2显示了类“课程班”的状态图。一个课程班对象可能处于状态“创建”、“开放”、“满额”、“取消”、“冻结”中的一种，当一个小课程段处于“开放”状态时，学生可以注册/增加/取消该小课程段，当注册一个小课程的人数达到最大限度时，该小课程变为“满额”状态;当注册人数小于最低限度时，该小课程将被取消;当允许注册/增加/取消课程的期限已到，该课程的状态将转为“冻结”。学期结束时，课程班对象将被加到一个历史日志中。

图3显示了注册课程的序列图。学生提交所要选修的课程的登记表，其中包含了候选的课程。系统首先检查当前是否处于注册时间，然后检查每门小课程的有效性(如课程是否已满额等)，学生是否满足前修要求，是否与他已注册的课程存在时间冲突。如果以上情况存在，则考虑他的候选课程。最后，系统将计算学费。图4则显示了取消一门小课程的序列图

2.6 结果

模型具有以下好处:

- 描述了一个抽象的课程注册过程，该过程可以应用于各种特定的情形。
- 将课程分成小的课程班带来了灵活性，允许学生根据自身的限制条件注册不同的课程班，同时也让不同的院(系)方便安排教学。
- 利用类图显示关键的文档。
- 每门课程的前修课程要求能够被清楚标明。
- 可以根据课程的生命期定义指定的注册阶段。
- 用例的每一个方面都可以在模型中表示，但不包括实现的细节。开发人员可以选择不同的实现途径。

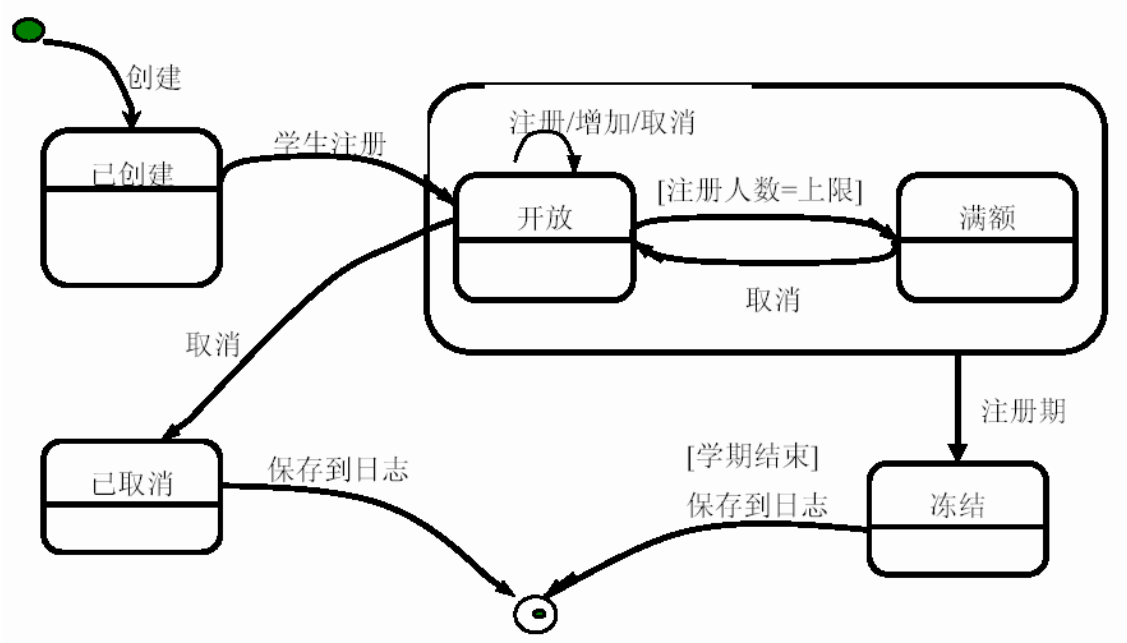


图2 课程班类状态图

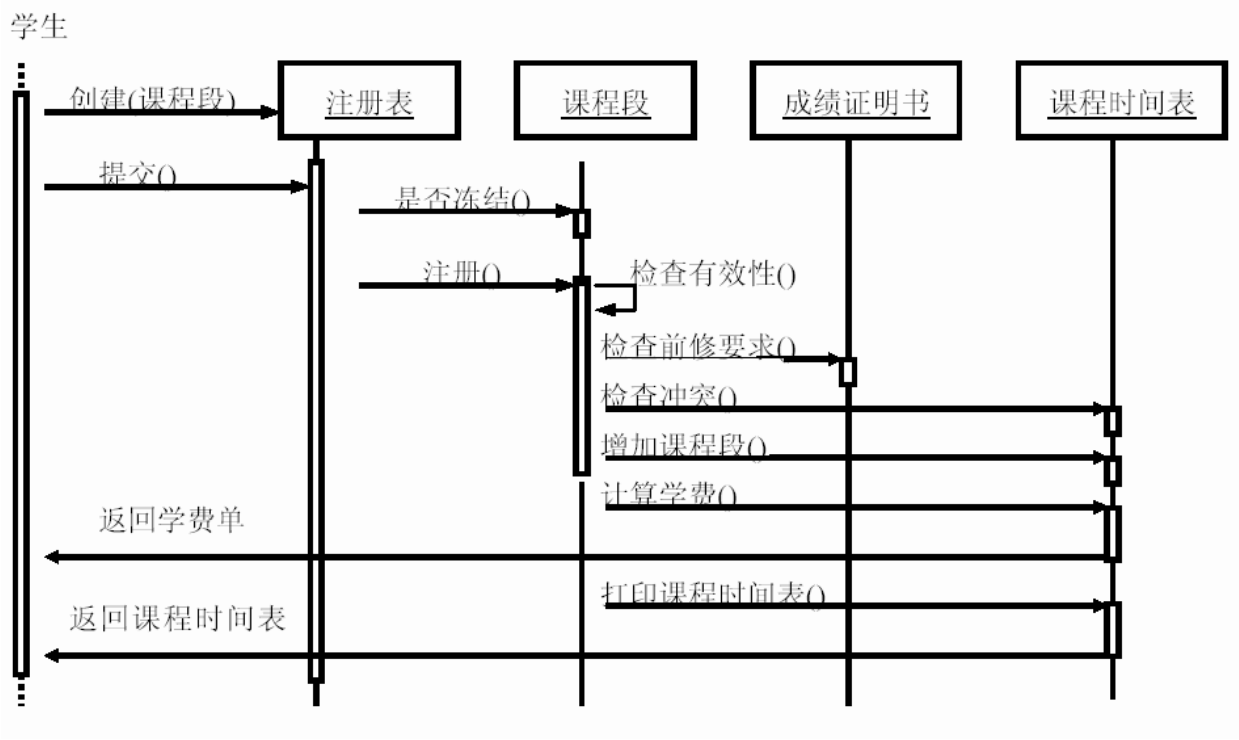


图3 注册课程序列图

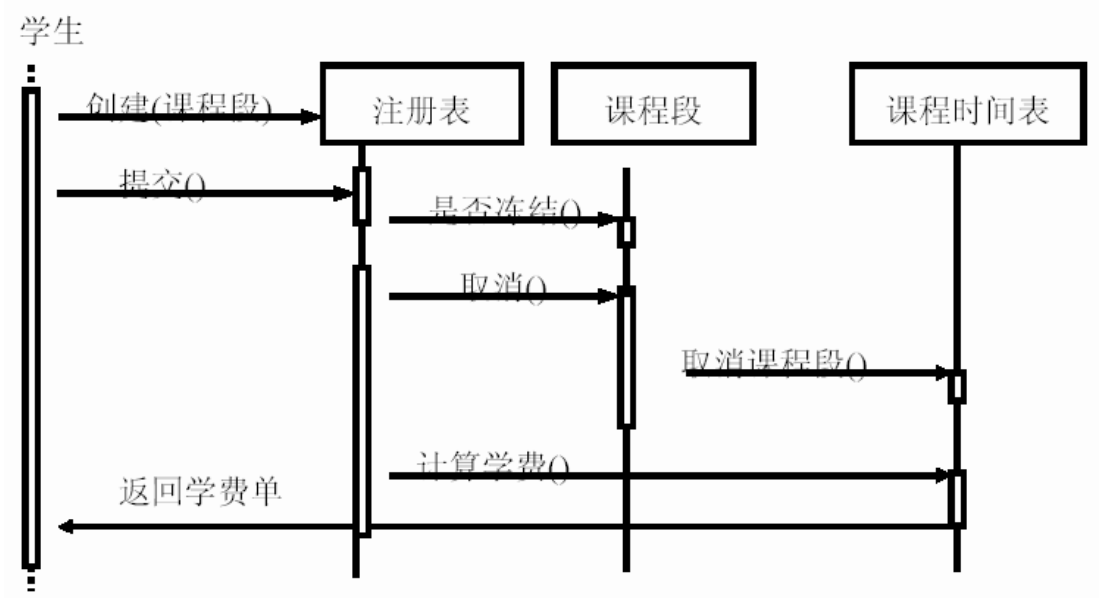


图4 取消课程序列图

当然，现实中并非所有的情况都可用以上的模型描述，这就需要根据具体的要求进行修改：

• 存在多种课程注册的方式，在部分大学里，学生在课程登记表上填写所要选修的课程和候选课程后，提交给登记员或辅导员，他们将把学生分成班级；某些学校的学生可在网上进行在线注册，在这种情况下，注册表在线创建和提交；有些大学允许学生通过电话注册，而不需要注册表；某些大学可能提供多种形式的注册方式。

• 某些大学并不将课程分为课程班。有些地方，某些课程班会有一定的注册条件，比如，特许课程班只允许特许学生注册；而某些课程班则只为特定专业的学生开设。

• 一些大学可能在学生的档案中设置约束，比如健康保险费。如果档案上的计算保险费约束没有消除，他就不能注册课程。

• 某些大学可能限制学生一个学期注册课程的最高和最低学分。

另一方面，模型中尚未包括以下的内容：

• 账单和支付方式。如何在允许的的时间里处理支付失败呢?在某些大学里，如果学生在指定的期限内没有支付所注册的课程段，则执行课程段将被取消。这样，学生需要首先付款，然后重新登记注册。

• 时间冲突。当学生由于时间冲突而无法注册时，他可以用一个候选的课程代替。有时候学生需要取消某些已注册的课程，然后重新选择其它的课程。

• 取消课程的政策。不同的机构可能存在很大的不同。

2.7 已知应用及相关模式

• 注册系统。在美国，很多大学使用相似的模型，如Florida Atlantic University、 North Carolina A. & T、StateUniversity、 U. C. L. A。

• 在线注册系统。很多美国大学提供了这个便利。

• 为基于WEB的远距离学习项目而设置的注册系统。一个例子是University of Phoenix、 AZ，它提供了大量的web课程。

• 工业培训中心的注册系统，如Lucent Training Departmen。

3 成绩管理

3.1 目的

教师对学生在课程学习的表现做出评价

3.2 上下文

开设课程的机构，学生参加这些课程的学习以后应该得到所学课程的成绩。

3.3 问题

学生学习完一门课程以后会认为自己已获得的一定程度的知识。教师给出的学习成绩有时达到了就业的学位要求，或者满足了个人的需要；有时却沦为学生参加了课程学习的证明。这表明，教师需要一定的方法对学生进行评价。

3.4 约束

- 需要有一个评价标准。
- 教师需要方便的方式记录他对学生的评价。
- 标准文档，如报告卡和证明书应该直接的显示在模型中。

3.5 解决方案

对参加课程学习的学生进行评价的最常用的方法是给学生一个成绩。教师记录两种成绩:平时成绩，如家庭作业的成绩，参加测试的成绩和参加项目的成绩等，这些成绩列在报告卡上；课程的总成绩， 这是根据报告卡计算出来的成绩。

证明书则记录了学生参加的所有课程的成绩。

图5显示了成绩管理的类图，学生类和教师类代表了两个最基本的实体，AcadPerson类是学生和教师的范化，一个教师指导多个学生，可能会教授多门课程。关联类“报告卡”描述了学生参加一门特定课程学习的成绩信息。类“证明书”则描述了学生的所有课程成绩。

图6显示了教师为学生打分的序列图。计算学生的评价成绩，并将最后的成绩增加到学生的证明书中。

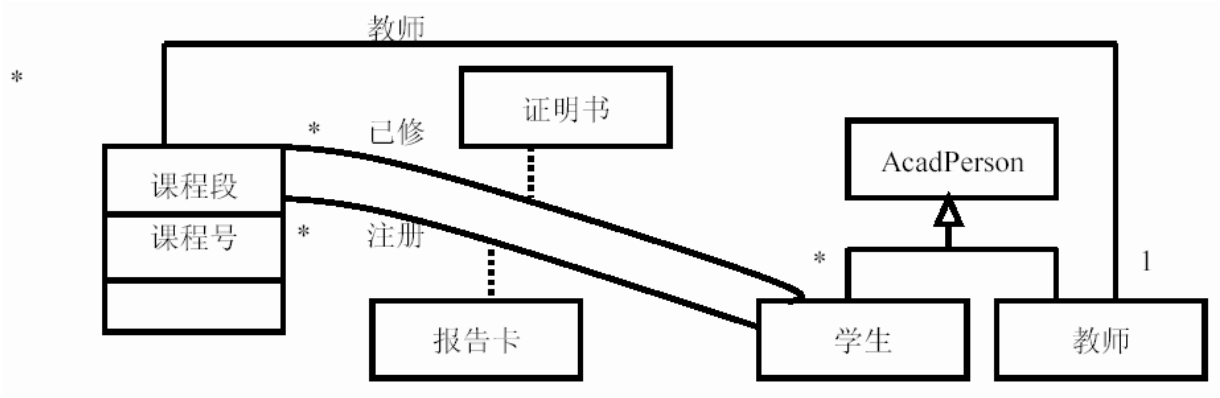


图 5 成绩管理实现类图

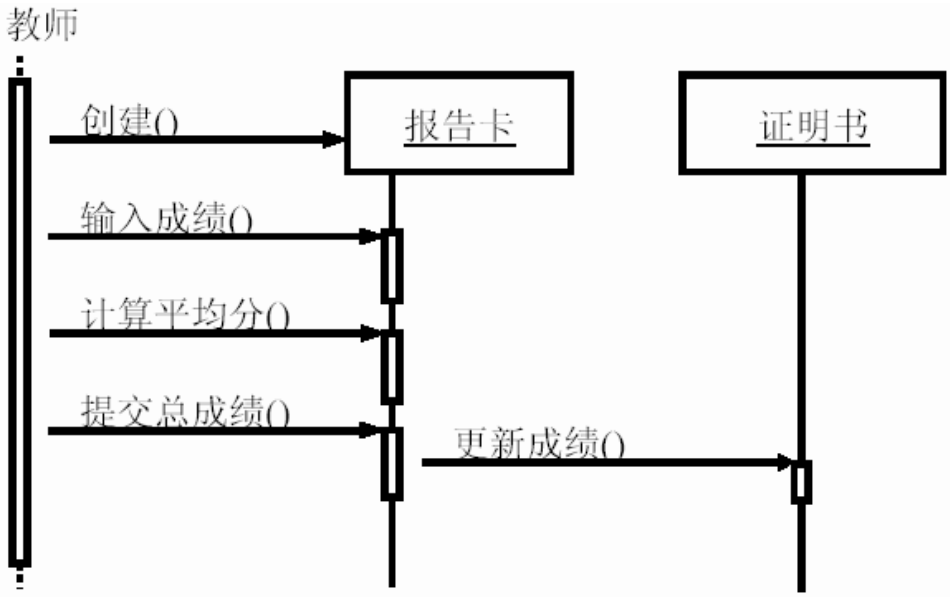


图 6 成绩评价序列图

3.6 结果

• 成绩管理模型描述了一门课程的成绩管理的基本内容。

• 成绩记录在标准的文档中，如模型中的报告卡和证明书。

• 如果只要求参加课程的证明，则报告卡换为出席记录卡。

模型中没有包括的方面：

• 某些课程可能为学生安排助教，在这种情况下，助教也会给学生一个评分，这时，学生的最后成绩可能有教师和助教共同评定。

• 存在不同的记分方式。比如，在美国大学里，用 “A”、“B”、 “C”、“D”、“F”； 有些大学则用“P”(pass)和“F”(fail)；在欧洲和南美洲，一般用数值表示分数，如0到5。

3.7 已知应用及相关模式

• 在线成绩报告系统。如Omni Middle school in Boca Raton、 FL。

• 基于WEB的远距离学习项目的成绩管理系统

4 课程管理

4.1 目的

此模式描述的是课程管理的概念模型，包括注册和课程成绩评定，同时也包括增加或删除课程，为课程班安排教师等基本操作。

4.2 上下文

提供任何课程的机构。

4.3 问题

在某些机构里，需要将课程注册和成绩评定以及与之相关的功能进行整合，如建立每个学期的课程列表、安排教师、跟踪辅导等。

4.4 约束

• 教师开设新的课程后，需要将之添加到课程列表中，而一旦某门课程不再开设，应该马上删除。

• 教师一般可以讲授多门课程，同时他们也会说明最希望讲授哪些课程。

• 学生应该了解他们的指导教师，以及这些教师的优缺点。

4.5 解决方案

整合注册管理和成绩管理模式，并增加跟踪对学生的辅导的机制，为课程段安排教师，增加或删除课程。以下是增加的用例:

- 增加、删除、更新课程。院系负责人增加、删除课程班或更新课程的内容。
- 增加、删除教师。
- 为课程班安排教师。
- 为学生指定辅导教师。

图7显示了将前面讨论的两个模式整合并加入一个新类后的类模型。

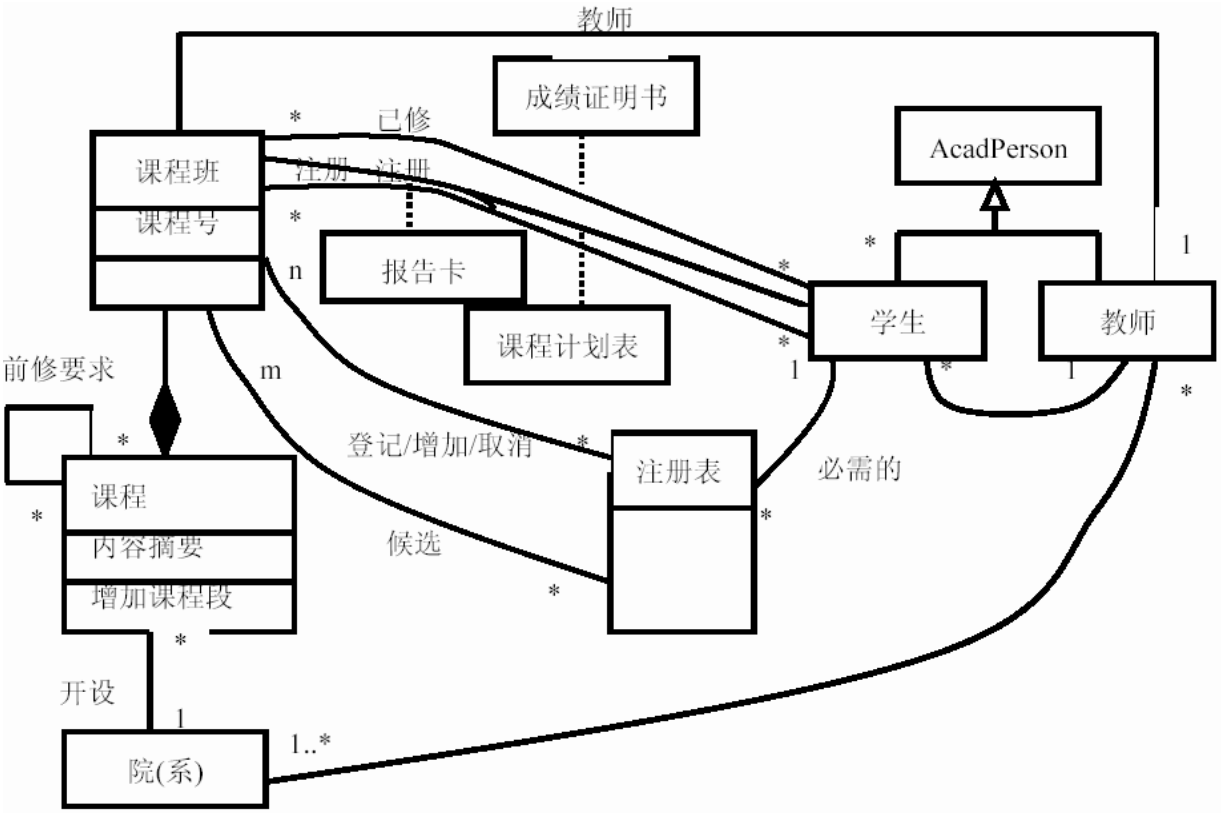


图 7 课程管理用例图

4.6 结果

整合后的模式具有以下好处：

- 增加或删除课程变得容易；
- 可以方便地为课程安排教师；
- 便于跟踪学生的辅导教师。

模式中还可以加进以下内容:

• 访问权限 注册系统只能被注册员使用。当然，如果允许学生在线注册，学生也有权访问，但教师仅仅可以录入所授课程的成绩;

• 安排教室和时间。

参考文献:

[Amb01] S.W.Ambler, *The object primer (2nd. Ed.)*, Cambridge University Press, 2001.

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal., *Pattern-oriented software architecture*, Wiley 1996.

[Bau00] D. Baumer, D. Riehle, W. Siberski, and M. Wolf, "Role Object", Chapter 2 in *Pattern Languages of Program Design 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.). Also in Procs. of PLoP' 97, http://jerry.cs.uiuc.edu/~plop/plop97

[D' S97] D. D' Souza. "Framework: Java to UML/Catalysis", *Journal of Object-Oriented Programming (JOOP)*, September 1997, 10-13.

[Fer99] E. B. Fernandez, "Good analysis as the basis for good design and implementation of object-oriented systems". http://www.cse.fau.edu/~ed/Goodanalysis.pdf

[Fer00] E. B. Fernandez and X. Yuan, "Semantic analysis patterns", *Procs. 19th Int. Conf. on Conceptual Modeling (ER2000)*, October, 2000. 183-195.

[Fer01] E. B. Fernandez, "A pattern language for security models", http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/

[Kim01] G. Kimovski, V. Trajkovic and D. Davcev. "Virtual learning system", *Procs. 2001 IRMA International Conference*, 427-430.

[Vad99] K. Vadaparty. "Mapping objects to tables", *Journal of Object-Oriented Programming (JOOP)*, July/August 1999, 45-47.

Agile软件开发丛书

PEARSON Addison Wesley

有效用例模式

Patterns for Effective Use Cases

Foreword by Craig Larman

[美] Steve Adolph
Paul Bramble 著

车立红 译

UMLChina 审

UMLChina 指定教材

清华大学出版社

UML 建模工具比较：Enterprise Architect 和 Rational Rose

Jie Zhao, Dunstan Thomas Consulting 著，ludingping 译

查看评论

自从 1997 年正式发布 UML 以后，大量商用 UML 建模 CASE 工具粉墨登场。这样为我们提供了许多的选择，同时也要求我们在选择正确的 UML 建模工具以更好地适应我们业务和软件应用程序开发需求，达到最好的投资回报率（ROI）方面做大量的调查。在这篇文章中，我们将比较两款 CASE 工具的 UML 建模能力、双向工程特性和项目生命周期支持：Sparx Systems 的 Enterprise Architect(EA)专业版 V.3.51 和 IBM Rational 的 Rational Rose 企业版 V.2002。

Enterprise Architect　　Rational Rose

为什么我们需要 UML 建模 CASE 工具

今天，系统的构建变得越来越复杂，UML 建模 CASE 工具为项目相关人员(如，项目经理，分析员，设计者，构架师，开发者等)提供了许多的好处。UML 建模 CASE 工具允许我们应用规范的面向对象分析和设计的方法与理论，远离纠缠不清的源代码，达到构建和设计变得更直观，更容易地理解与修改的层次。在大型项目中，使用 CASE 工具更重要。通过使用 CASE 工具：

* 通过用例模型，业务/系统分析可以捕获到 业务/系统需求。

* 设计者/构架师 所作的设计模型能在不同层次的同一层内清晰表达对象或子系统之间的交互（典型的 UML 图如类图和交互图）。

* 开发者能快速地将模型转变为一个可运行的应用程序，寻找类和方法的子集，以及理解它们如何交互。

模型被看作是蓝图和构建系统的最终手册。同样，建模也就是一种从高层并以适当的形式来考虑一个设计的表述和理解它怎样运行的能力。

出于这些动机，UML CASE 工具以及对应的方法论为我们提供了一种因系统太复杂而不能理解下层源代码的描述系统的方法，同时允许我们更快更便宜地开发正确的软件解决方案。

当然，要考虑 CASE 工具在 UML 建模能力，项目生命周期支持，双向工程，数据建模，性能，价格，可支持性，易使用性等方面的不同。这篇文章将探索 Rose 与 EA 在 UML 建模，项目生命周期支持以及双向工程领域的相同点和不同点，希望能帮助你在你的项目中选择正确的工具。

UML 建模特性

UML 标准由三部分组成，即：构造块（如对象，类，消息），构造块间的关系（如关联，泛化）和图（如，活动图）。UML profile 使用 UML 可扩展性机制扩展标准 UML 符号，即，构造型，标注值和约束。EA 专业版 V.3.51 和 Rational Rose V.2002.05 都支持 UML 1.4 九种图中的八种标准 UML 图 - 用例图，类图，序列图，协作图，活动图，状态图，实现图（组件）图，部署图，和几种 UML Profiles. 如果需要，对象图可以使用协作图来创建。不同点仅仅存在于创建 UML 图（表 1）和扩展 UML profiles 时所支持的一些特性。

UML 图	功能	EA	Rose
用例图	建立描述领域的边界	Yes	No.但是，一些工作使用文本或包。
序列 协作	序列图与协作图之间的相互转化	No	Yes
序列	更改消息的范围	Yes	No
序列	显示消息层次号码	Yes	No
序列 协作	在浏览器中创建对象	Yes	No
序列	管理控件的焦点	容易	困难
所有	图的属性	Yes	No

表 1. EA 和 Rose 的 UML 图建模比较

Enterprise Architect 有一个通用的 UML profile 机制用来加载和运行不同的 Profiles。Enterprise Architect 为 UML profiles 指定一个特定格式的 XML 文件。而在 Rational Rose 中却需要生成一个附加项。 表 2 展示了在 EA 和 Rose 中 UML profiles 的可用性。

UML Profiles	EA	Rose
业务流程建模	支持 Eriksson-Penker 业务流程建模扩展	使用 UML 活动图
业务建模	No	Yes
数据建模	Yes	Yes
用户体验建模	Yes	No
Web 建模	Yes	Yes
XSD Schema	Yes	No
XML DTD	No	Yes

表 2. EA 和 Rose 的 UML Profile 比较

双向工程

双向工程包括正向工程 — 从模型到代码 和反向工程 — 从代码到模型。一旦设计完成后，使用模型（设计模型和数据模型）信息能够生成特定编程语言的源代码或者数据库的 DDL 脚本。当开发人员添加/更改代码或数据库实现时，设计和数据模型能够通过双向工程同步代码或 DDL 脚本以保持一致。表 3 显示了 EA 和 Rose 双向工程的特征。

语言	EA	Rose
ANSI C++	Yes	Yes
Visual C++	No	Yes
VB6	Yes	Yes
Java	Yes	Yes
C#	Yes	No
VB.NET	Yes	No
Delphi	Yes	No. 第三方附加项.
J2EE/EJB	No	Yes
CORBA	No	Yes
Ada83, Ada95	No	Yes
Database	Yes. 从数据模型到 DDL 脚本的正向工程。ODBC 数据源的反向工程	Yes.DB2，Oracle, SQL 92, SQL Server, Sybase
COM	No	Yes. 只是反向工程
Web 应用程序	No	Yes

表 3. EA 和 Rose 的双向工程

EA 为类生成类的源代码文件放在同一个包里。Rational Rose 在 VC++或 VB 中更多的涉及到具体的项目。Rational Rose 也可以通过向导和提供代码模板来创建类，这样可以大大增加源代码生成的数量。另外，EA 和 Rose 都可以应用设计模式。当使用 EA 时，用户必须自己创建模式，而 Rose 则提供了 Java 的 20 种 GOF 设计模式。

项目生命周期的支持

CASE 工具应该为团队中的所有队员完成他们的任务提供支持。关于项目生命周期的支持，EA 将大量的功能合成一体,而 Rose 则主要是一个建模工具，它可以与其他的 Rational 或第三方工具集成，如 RequisitePro, Test Manager, Soda, MS Word, MS Project 以达到同样的目标。 表 4 比较了 EA 和 Rose 在不同科目的功能支持。

项目科目	EA	Rose
业务建模	Yes. 使用 UML Profile 为业务流程建模	Yes.使用业务用例模型
需求管理	Yes. 功能和非功能需求；需求跟踪矩阵	合并 RequisitePro
分析和设计	Yes UML 类图和交互图，如果需要可以添加一些原型，如<<层>>，<<用例实现>>	Yes UML 类图和交互图 框架向导提供了一系列的模板来构造模型
实现	参见表 3 适合 C++，VB，C#和 VB.NET 项目	参见表 3 支持大多数语言，除了.NET 外
测试	Yes	No. Quality Architect 提供了单元测试，但是它需要其他的 Rational 工具，如 Test Manager,Robot
版本控制	不直接支持。使用控制单元，为将来发布计划。	集成 SCC 相应版本控制应用程序
项目管理	风险管理 资源分配 项目预算	No
Web 发布	Yes	Yes
生成报表	Yes	No. 使用 SoDA.
多用户协作	Yes	Yes

表 4 EA 和 Rose 对项目生命周期的支持

结论

大体上，EA 和 Rose 在 UML 建模能力上有相似的功能。 EA 和 Rational Rose 都支持 UML 九种图中的八种。 从表 1 中可以看出 EA 在用户友好性的灵活性中比 Rose 更胜一筹，特别是序列图。 在双向工程中，Rose 比 EA 支持更多的语言，除 C#和 VB.NET 外(事实上，Rational 开发了另外的工具 - Rational XDE for .NET 就是针对.NET 环境的)。 表 4 阐明，在项目生命周期的支持方面，EA 相对于 Rose 来说，是更好的选择。尽管你可以购买其他的 Rational 工具来协助它，但是绝大部分公司在考虑成本问题时却不认为这是一个可接受的方案。当然，你需要工具上没有或第三方工具不支持的一些其他的特定功能时，这也是要考虑的重要因素。在这一点上，Rational Rose 得到了更广的支持。

最后，经过一系列同类型的比较，是不是费用也不同？ 是的——非常大的不同！单是 Rose 花费就是 EA 的 28 倍。 如果你要比较项目生命周期的支持，假设你是一个 Rose 用户，你将必须去购买 Rational 捆绑的一整套产品，如 Requisite Pro, SoDA, Test Manager 等。虽然附加的工具比同类的 EA 提供了更丰富的功能，但是在大部分时间里 EA 的基本功能已经够用了。在写本文时，EA 企业版(最高版本支持 SQL 后台)的费用是$179.00(£111.58)而 Rational Rose 企业版的费用是$5024(£3140)，一天整套开发包（Rational Developer Suite）的费用是$8976(£5610)。

征稿

http://www.umlchina.com/xprogrammer/xprogrammer.htm

The Addison-Wesley Signature Series

Patterns of Enterprise Application Architecture

中译本即将上市

Martin Fowler
With Contributions by
David Rice,
Matthew Foemmel,
Edward Hieatt,
Robert Mee, and
Randy Stafford

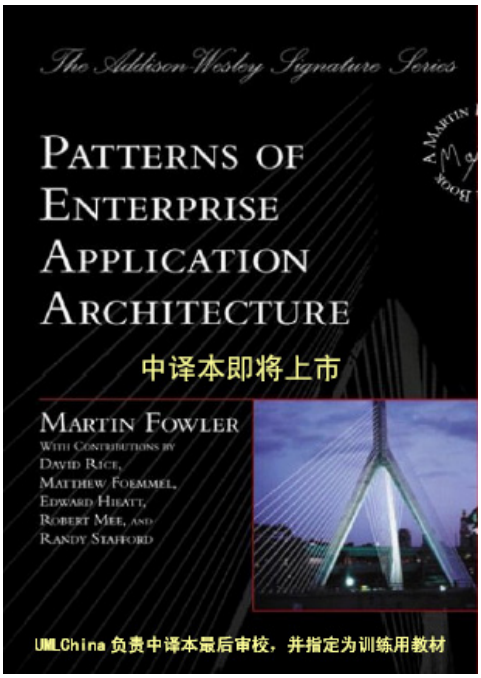
UMLChina 负责中译本最后审校，并指定为训练用教材

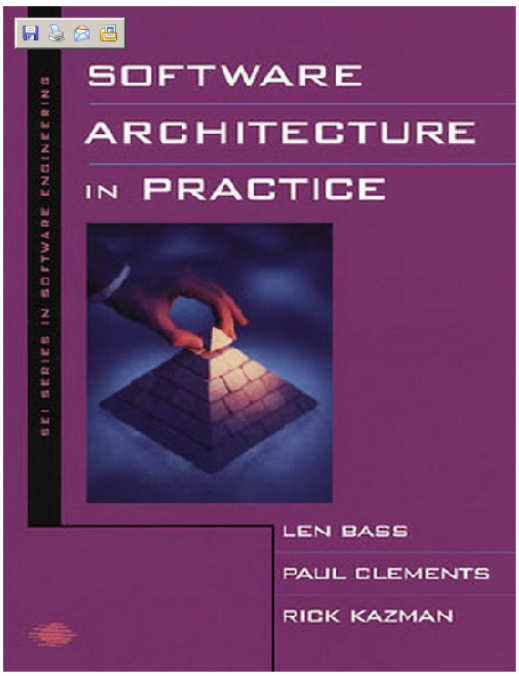
SAIP 和 PEAA 的对照与比较

Ramona Su 著，李巍 译

查看评论

一亿美金，五年的合同。所有你要做的就是推出一个新的系统，使得（美国）伊利诺伊所有的公立大学能够同步所有大学生的申请，从而促进对入学过程的评估工作。你可能不再像过去大学程序设计课程所作的那样，只是坐在那里便可以开始编写该项目的代码。对于这个项目，以及所有诸如此类的大型项目而言，你需要为系统创建一个架构（architecture）。这个架构将会作为你项目的基础。为了打下一个良好的基础，在架构创建时需要考虑多个方面。你的涉众（stakeholders）是谁？用户（users）是谁？最佳的项目分解方式是什么？能够用到哪些最佳的模式？在 Software Architecture in Practice（SAIP）和 Patterns of Enterprise Application Architecture（PEAA）中会对这许多问题做出回答。尽管对于「如何创建架构」SAIP 和 PEAA 有着不同的观点，但它们都有助于阐述架构创建的方式，以及强调其对于系统如此重要的原因。





「设计一个架构」是“与决定相关的，开发者希望在早期便能做出正确的决定，因为他们意识到这些决定难以变化”（PEAA 2）。在 SAIP 和 PEAA 中，作者都将架构设计的重要性强调为项目的主要基石。正如 Fowler 在 PEAA 中所言，这是因为你一旦确定了架构上的决定并开始实现系统，如果你发现缺陷将很难回去重新设计架构。因此，关于架构的创建需要多加思考—-无论是作为一个整体，还是各个部分需要如何交互。SAIP 给出了一个需要在创建架构时考虑的设计过程方面的示例。作者论述了「在创建架构之前理解架构驱动者（architectural driver）」的重要性。“为了确定架构的驱动者，需要识别优先级最高的业务目标（business goal）...将这些业务目标转变成质量场景（quality scenarios）或用例（use case）”（SAIP 155）。在这个例子中，架构设计师必须要理解系统的驱动者和系统所要达到的质量属性（quality attribute）。架构设计师必须要采集系统之外所预期的信息，然后为所获得的这些质量属性确定优先级，以使系统成功。于是架构设计师必须要更进一步，通过了解为获得某种质量属性所应用的策略的权衡，以及它将如何对系统产生影响（SAIP 第 5 章）。是否将会为获得涉众满意的较高安全性而具有较低的性能呢？系统较低的可用性是否将依然能够满足涉众提出的可用性需求呢？这些有代表性的问题有可能会在创建系统架构时予以涉及。这许多问题包括要理解为架构所做决定的权衡。为了能够对这些权衡进行分析，架构设计师必须要将系统架构看作是一个整体。他们不能专注于优化一个质量属性（如性能），或者系统可能无法完成的其他系统需求（如安全性），以及涉众的需求。在这个小示例中可以看到，SAIP 强调了把时间和精力放在一个架构设计过程方面和系统驱动者的重要性，以及甚至是在架构未成形之前就要从整体上来看它将会如何对架构产生影响。

同样，PEAA 也强调在架构创建初期投入大量时间的重要性。“企业应用通常需要处理复杂的数据—而且非常之多，并伴随着无法按照逻辑推理来测试的业务规则。尽管有些技术和模式与各种软件相关，但许多则只是与一个特定的分支所关联”（PEAA 2）。在整个 PAEE 中，Fowler 论述了各种在系统中使用的模式，并且他总是会为每个模式包含一个“何时使用”的部分。这样，Fowler 强调了知道所要设计系统的需求的重要性。通过理解这一点，一个架构设计师便可识别出在架构中使用一个特定模式和使用其他模式之间的权衡--某一模式将会为一个系统及其需求很好地工作，而其他的便不能。例如，在论述对象-关系映射模式（object-relational mapping pattern）时，Fowler 说“一个比较好的方式就是将 Domain Model（领域模型）与数据库完全隔离开来”，并使用 Data Mapper 模式。在这个例子中，理解不同对象-关系映射模式之间的权衡并不是唯一重要的事情，而是要理解它将如何与架构中的其他模式进行交互。通过强调这样一些交互的重要性，PEAA 希望架构设计师在选择一个架构内调用的模式之前，能够理解整个系统所涵盖的需求和质量属性。

SAIP 和 PEAA 看上去都强调了要将大量的时间和前期工作放在架构的创建上。尽管创建工作需要花费许多的时间，但这在长期的工作中是非常值得的，因为其“能够带来与质量进度和成本相关的最高投资回报”（SAIP xii）。如果在架构设计过程中出现错误和大的缺陷，则更往往是“系统的结构出现错误，并且这将无法通过一些缝缝补补、裁裁减减的工作便能够修复”（SAIP xii）。

另一点在 SAIP 和 PEAA 中关于架构创建方式的相同之处就是，两本书都给出了架构设计过程在现实生活中的经验。“我们的书面向软件专业人员－设计和实现大型软件系统的人们，软件专业人员的管理者...”（SAIP xii）。在整个 SAIP 之中，有众多的案例分析来展示「如何运用创建架构的思想」。比如，SAIP 的第六章便是一个有关航空运输控制系统的案例分析，并且设计了一个具有高度可用性的系统，Initial Sector Suite System。在这个案例学习中，系统需要完成多个需求和质量属性—-可用性，高性能，开放性，软件和硬件的可修改性等等（SAIP 133）。该案例分析展示了架构设计师在创建他们的架构和发觉需要做出权衡时，应该如何来看待需求和质量属性。这样，他们展示了如何对「刻画一个真实系统的架构设计的系统质量属性」进行优先级的排定。

同样，PEAA 在论述架构设计时也引入了来自现实世界的经验。“本书表现了这一些以某种形式书写的（构建计算机系统的）知识，我希望能够有助于你们尽快地学习这些经验，或者在我归纳这些模式之前能够与其他人更加有效地交流”（PEAA 1）。贯穿本书用到的各种模式的描述，Fowler 提到了某些他在真实世界中经常使用的模式，“在我的职业生涯中我专注于企业级应用，因此我在这里的模式全部都是有关于此的”（PEAA 2）。他还提到其不经常使用的模式，以及为何在某些系统中不怎么使用它们。通过给出这样的见解，整个书中 Fowler 都不断在提醒读者，模式的选择能够对一个系统的架构产生重大的影响，最终会影响到系统的成败。他着重强调了「吸取他在创建和设计架构中所学到的教训」的重要性，以及不要在设计和创建新的架构时犯同样的错误。

尽管这两本书以及它们对架构创建的观点有着某些相似之处，但他们却有着诸多不同。在观点上不同的主要原因是 PEAA 专注于企业级应用架构设计中用到的具体模式，而 SAIP 则更专注于架构创建的通用方面。因为 PEAA 专注的应用类型更为具体，使得 Fowler 描述的这些在系统中重复出现的架构模式具有意义。因此在该种意义上，PEAA 所创建的架构的范围更为狭窄，因为它基于一个具体类型的软件—企业级应用。另一方面，SAIP 看起来则是专注于软件架构更为通用的方面—-论述如何开发一个系统的需求和质量属性，以及评估在质量属性间所进行的权衡。

这两本书另外的不同点是 PEAA 往往更专注于架构创建的技术方面，而 SAIP 则更偏重于架构的业务方面。在 PEAA 开头，Fowler 就描述了书中所述模式的使用动机（motivation）。然后他便继续论述模式的细节—-模式如何工作，何时使用模式，以及甚至描述了如何实现模式。Fowler 论述了模式间的权衡，以及哪些可以相互良好地工作。其专注于模式是由于 Fowler 这本书面向“那些正在构建企业级应用，以及想要提高自身对架构问题的理解或相互间对此进行交流的程序员、设计者和架构设计师”（PEAA xx）。因此该书旨在帮助技术团队看出「模式会如何影响系统的架构」。而另一方面，SAIP 则更为专注架构创建的业务方面，以及它如何促进在架构创建中所做的抉择。SAIP 的作者论述了架构上的业务周期和涉众，而 PEAA 则对软件架构的业务方面鲜有提及。SAIP 更多的是在揭示「架构设计师在设计一个具有业务世界的系统时所需要的交互」。看起来 SAIP 试图要去拓宽构架设计师的视角，从而超越当前开发系统的开发者和技术人员，以及什么才是在设计架构时业务世界所出现的通用事物。

此外，关于架构创建，SAIP 论述了其他 PEAA 所不具有的方面。我想这是因为 SAIP 更直接地将焦点放于业务世界中的软件架构和通用的软件架构上面。例如，SAIP 会花费时间来谈论「架构对文档的需要」。SAIP 包含文档部分的动机是由于项目中不同的涉众将需要不同类型的文档。例如，管理者更加关注项目团队之间的工作划分，因此他们将需要一个更为广泛的和一般的构架视图。然而，开发人员在编写他们的系统部分时，则需要更为详细的文档和架构的技术需求。另外一个在 SAIP 中论述的问题是「从老的架构中重新创建一个架构」。我想这样做是为了要将软件构架设计师在业务世界中更加通用的经验包含进来。在业务世界中，软件构架设计师并非总是要创建一个新的架构。有时他们可能需要重新构造现有的架构并以此作为构建的基础，或者重新对其设计以适应涉众的需要。

“由于架构出现在一个产品生命周期的前期，其将为后续的任何工作打好基础--系统的开发，集成，测试和修改”（SAIP xii）。该引述看起来正是为何要编写这两本书的症结所在，将架构放在系统设计的核心位置。尽管这两本书对于架构创建的论述在方式上存在着许多不同，但根本主题都是为了要在创建系统架构时做出可能的最佳决定。你所做出的决定能够让你的项目成功，也能够铸成大错。我想结合这两本书，便能给出一个真实世界中软件架构的良好视图。

斐力庇第斯从马拉松跑回雅典报信，虽然已是满身血迹、精疲力尽，但他知道：没有出现在雅典人民面前，前面的路程都是白费。

学到的知识如果不能最终【用】于您自己的项目之中，也同样是极大的浪费。而这最后一段路最是艰难。

UMLChina 聚焦最后一公里，所提供服务全部与您自己的项目密切结合，帮您走完最艰难的一段路。

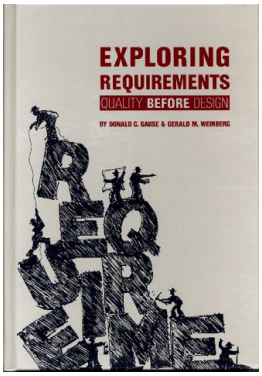
相识何必曾相逢

Think 著

查看评论

出版社的编辑请我为《探索需求》写个东西，可是等我写完了，才有机会真正看到这本书！

一年之中，我的大部分时间，都花在东奔西走(例如，写这几个字时，我正身处西南部的一家钢铁企业)为软件开发团队作指导之中，指导团队如何将用例驱动的面向对象方法真真正正应用到自己的当前项目中。贯彻的起点往往是需求技术，即用例技术。用例是开在上面的那朵鲜艳的花，它需要植根于丰厚的土壤——各种已有的有效需求实践，它们会在这本或那本需求技术书籍中提到。不，我接下去不是说“我看过的这些书籍都很推崇《探索需求》”。事实上，被推崇的经典书籍很多，《探索需求》淹没其中，又因年代较久，没有特别引起我的注意。



直到有一天，我在马桶上看Dean Leffingwell的书，这位Requisite公司的CEO、RequisitePro【1】的开发者在他的书中说，RequisitePro的思想起源于Donald Gause和Jerry Weinberg的著作，尤其是《探索需求：设计之前的质量》——这比10位专家大叫“经典！经典！”更能打动我，我开始对这本书有了浓烈的兴趣。可惜，这本书

不象其他一些书一样，那么容易弄到“电子版”，暂时只有通过“猜想”来间接阅读它。再次翻开那些我看过的书，看到Donald Gause和Jerry Weinberg的著作被引用时，感觉果然有所不同。再后来，我又发现《探索需求》出版10年之后另一本可爱的书《掌握需求过程》是由Weinberg写的序，于是再次从侧面印证了我的想法。

碟子里一个包子，一个馒头，你吃了一个下去，包子还在，为什么？答案是：你吃的是馒头。这个简单的道理，就是有人假装不懂。上面的话不是我说的，是古龙说的【2】。同样，无论如何腾挪闪躲，你最终要交给客户一个满足需求的软件。这个简单的道理也经常有人假装不懂，他们宁可躲在办公室把玩“技术”，也不愿去和涉众一起“探索需求”。经常有开发人员问我，“老师，要编写用例文档，那岂不是很烦？”我经常会表达我的愤怒，“用例文档怎么是编写【3】出来的？是通过和涉众交流提炼出来的！写几个字你都觉得烦，我敢肯定用例背后隐藏的艰苦工作你从来没有去做过。”

我猜想，《探索需求》应该就是使这些“艰苦工作”变得容易一些的一本书。现在，我终于有机会真正看一看这本书了。

它会不会象无视频时代的网恋一样，结局大多是见光死？

且听下回分解。

（写于2004年2月）

【1】RequisitePro，领先的需求管理工具。1997年被Rational收购，2002年随Rational被IBM收购。

【2】《陆小凤传奇之银钩赌坊》，古龙

【3】Cockburn的名著就被译为《编写有效用例》

Agile软件开发丛书

PEARSON Addison Wesley

有效用例模式

Patterns for Effective Use Cases

Foreword by Craig Larman

[美] Steve Adolph
Paul Bramble 著

车立红 译

UMLChina 审

UMLChina 指定教材

清华大学出版社

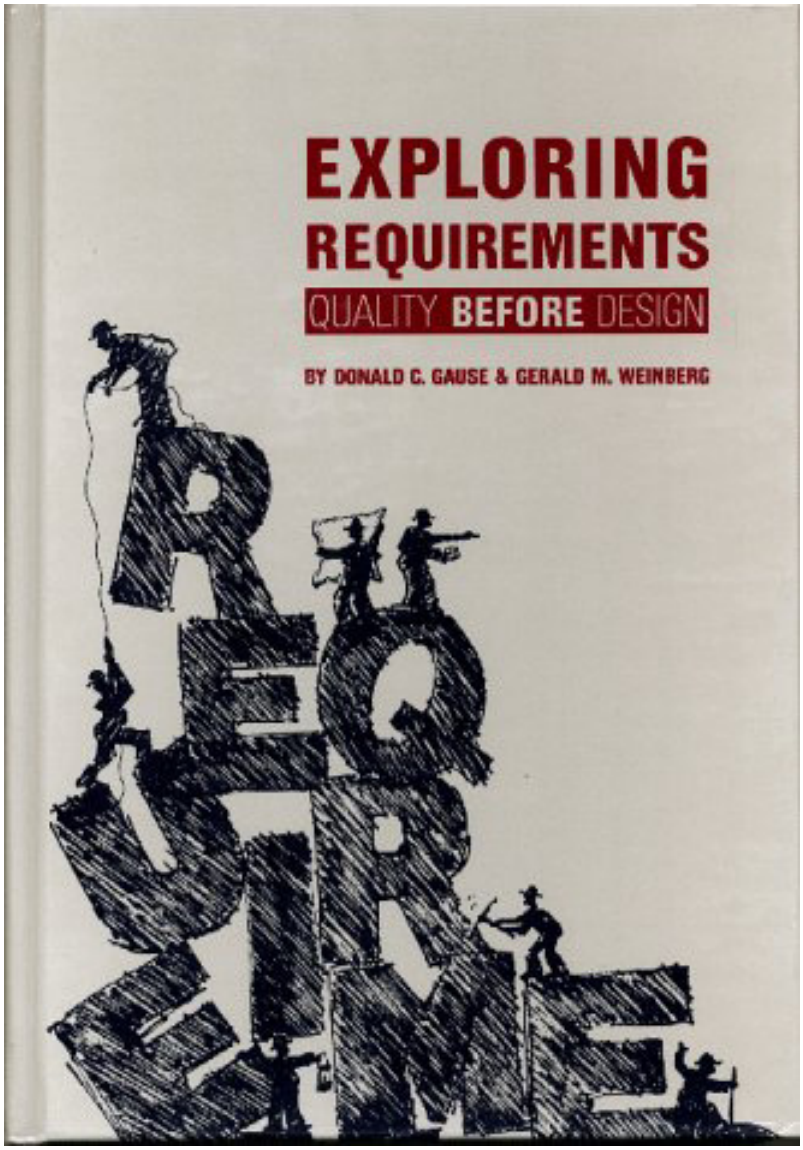
《探索需求》中译本（草稿）节选

Donald C. Gause、Gerald M. Weinberg 著，章柏幸 译

查看评论

本书中译本即将由清华大学出版社闻洁编辑室制作，闻洁编辑室曾经为中国读者引进了《人月神话》、《人件》等书。译者章柏幸的译著《你的灯亮着吗》2003 年获得大奖。

UMLChina 向您推荐本书，并将之作为 UMLChina 训练的辅助教材。



第 2 章 在陈述需求中的含混性

随便什么时候只要你使用那些忽略了人的因素的工具，你就不可能完美地描述需求，并且还会造成含混性。然后，含混性就会带来对相同需求的不同解释。

2.1 含混性的例子

例如说，图 2-1，2-2 和 2-3 展示了三种完全不同的建筑物，但它们都是按照下述相同的意义含混的问题描述而建造的：

创建一种方案来保护一小组人，使之免受其环境中的不利因素伤害。

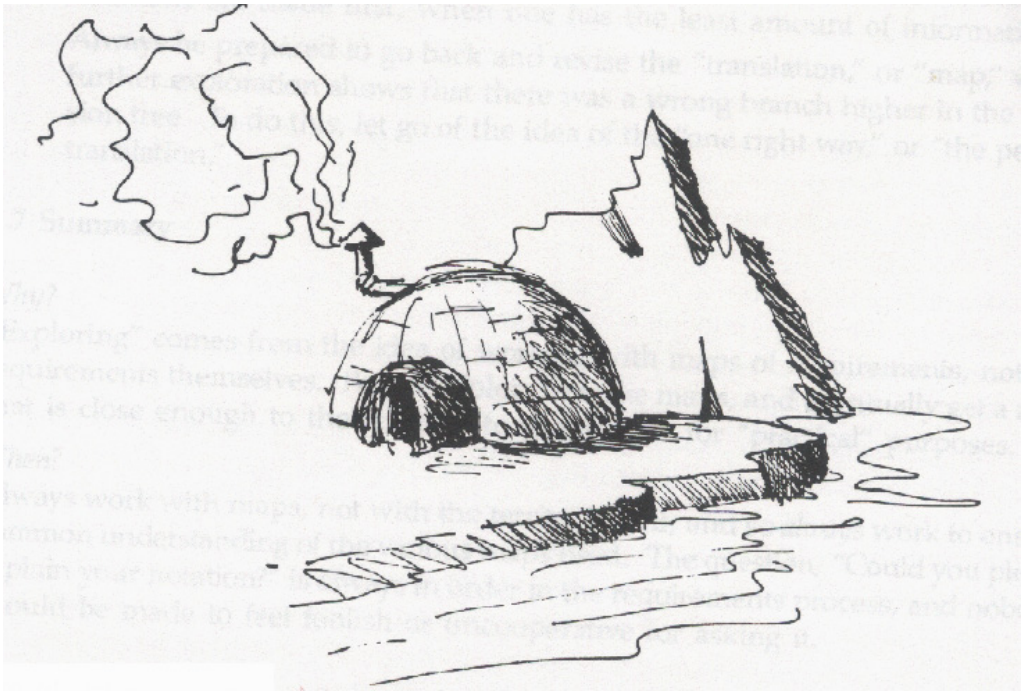


图 2-1 圆顶建筑——用本地建筑材料建造的土著房屋

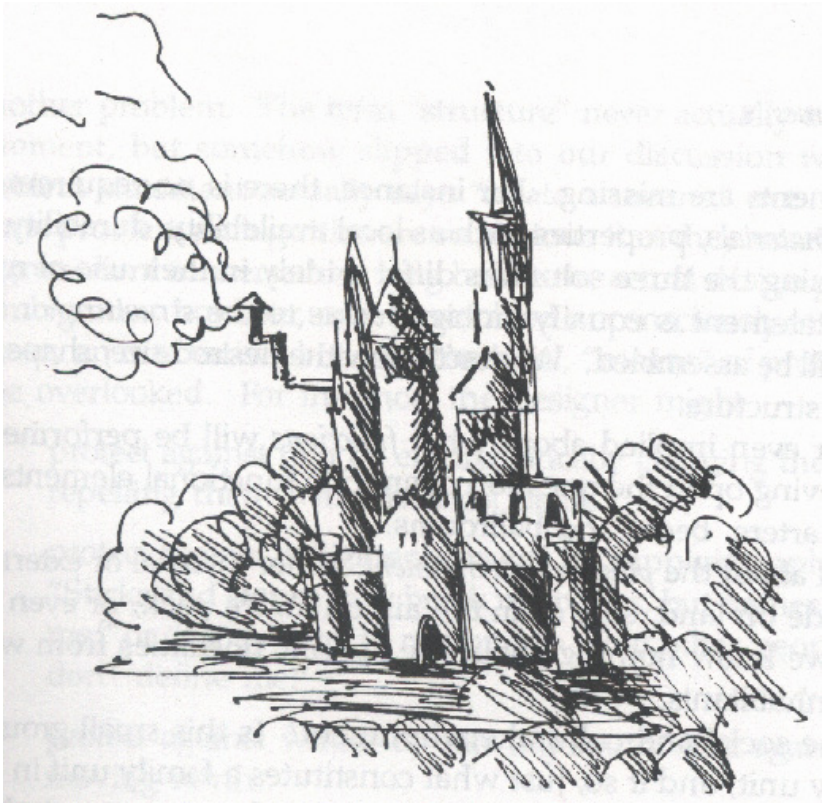


图 2-2 巴伐利亚城堡——为威慑邻邦而建造的房子

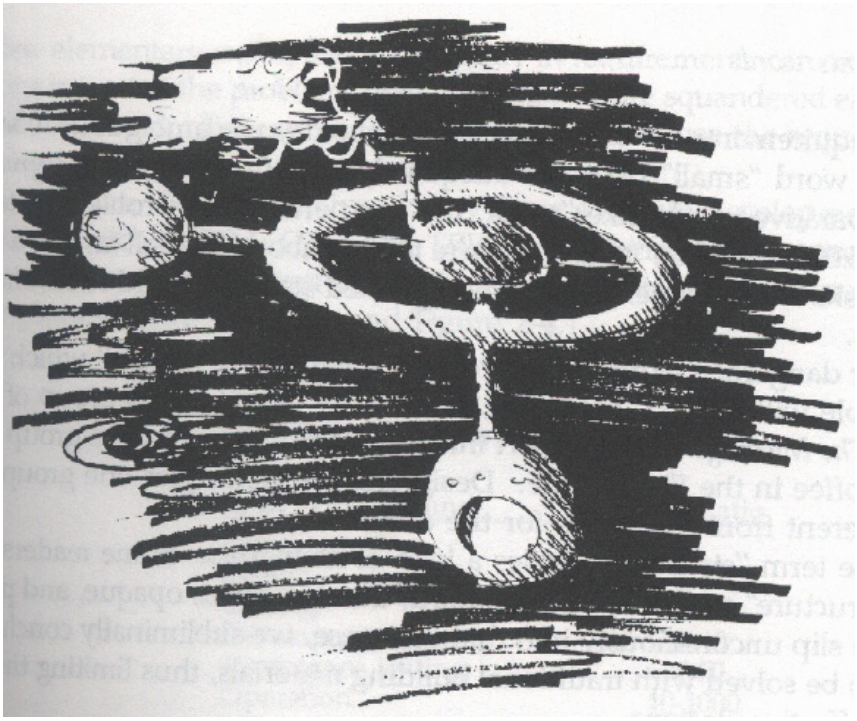


图 2-3 太空站——用于观察的可移动房屋

首先，这三种建筑确实提供了所描述问题的有效的解决方案，而且这些解决方案具有令人咂舌的差异。通过考察这些解决方案之间的差异，我们发现线索在于需求中的一些含混性。

2.1.1 缺少的需求

有时候需求是有遗缺的。例如说，没有关于*材料属性*方面的需求描述，这些属性包括当地可用性、耐用性或成本。因而三种不同的解决方案在其使用材料方面的强烈差异就不会那么令人惊讶了。

对于建筑方式或如何组织这些建筑材料的问题描述同样是含混的。我们不知道所期望的建筑物的大小、形状、重量或者寿命。

这些建筑物内部应该包含的*功能*几乎什么都没有说，甚至都没有暗示；这样，例如像火炉、仆人间、床和客厅这样的一些详细的功能要素都确定不了。

无论是内部的还是外部的*物理环境*也什么都没有提到，建筑物可能是在陆地上或大海里；在空气中、在冰层上甚或在外太空。而且，我们也不知道我们需要保护的居住者所面临的详细不利因素。

其社会和文化环境又是什么？这一小群人是一个家庭吗？如果是的话，那么在这种特定文化下的家庭是如何构成的？可能它是一个工作小组，例如猎人或石油勘探工程师；或者可能是一个娱乐小组，例如玩扑克或跳方块舞的人。

2.1.2 含混的词语

即使在需求被明确地说明时，还有可能使用了含混的词语。例如说，词语“小”不能充分地说明*人群的大小*。注意那些在问题描述中按比较估计的词语，比如说“小的”或“便宜的”。如果我们是在讨论内布拉斯加（Nebraska，美国州名）大学总部比赛的足球迷，那么 25000 人也只是一个“小的”人群，而这就需要一个新的半球形露天大型运动场才能够满足所描述的需求。

另一个在问题描述中的危险词语是“组”，它暗示这些人将会有相互影响，但不知何故，这里也不清楚其究竟。表演“*费加罗的婚礼*”的演员们之间的相互影响与在费加罗咖啡馆喝咖啡的一群人之间的关系截然不同。为某个小组设计的建筑物将会和为其他人设计的完全不同。

甚至是术语“建筑物”也带来了一大堆的含混性。有些读者可能会推断“建筑物”意味着某种坚固、耐用、可靠、不透明而且可能是巨大的东西。如果我们在无意中滑入了这种推论，我们潜意识里就断定需求可以通过传统建筑材料来获得满足，而这就限制了那些有可能有效的设计的范围。

2.1.3 无意中引入的假设

当然，我们可以通过小心探索问题描述中的每个单词的多种含义来预防那些含混的词语，但是这样我们还是避免不了另一个问题。术语“建筑物”实际上从未在问题描述中出现，但不知何故它在我们不注意的时候进入了我们的讨论。问题描述实际上说的是“创建一种方案”而不是“设计一种建筑物”。

有些问题的含混性是很明显的，它们能够在实际设计过程开始之前设计者和客户之间长时间交谈的不经意中获得解决。可是更多微妙的含混性，可能需要在设计者脑海里不知不觉地解决。既然这样，一种创新的、无需建筑却又能保护一小组人的“方案”可能被忽略了。例如说，设计者可能会：

1. 为了免受带有静电雨滴的雨的影响，可以通过电场来解决。

2. 为了免受好战分子的影响，可以通过警告语来解决，比如“枪棒无眼，和谈为贵”或“人不犯我，我不犯人”。

3. 为了躲避寒冬可以迁移到南方，为了躲避烈夏可以迁移到北方。

2.2 含混性的成本

这些需求中的含混性的最基本的例子只能让我们注意其对问题的巨大影响。每年因为生产不符合需求的产品就会浪费数十亿美元，而这些绝大多数是因为没有清楚地理解需求。

例如说，Boehm（原书底注：Barry W. Boehm 著《软件工程经济学》，Englewood Cliffs，美国新泽西州：Prentice Hall 出版社，1981）分析了诸如 IBM、GTE 和 TRW（GTE：通用电话与电子设备公司，General Telephone and Electronics Corporation；TRW：汤普森·拉莫·伍尔德里奇公司，T hompson Ramo Wooldrige Inc。）等公司的 63 个软件开发项目，对于由需求阶段的错误假设所带来的错误，为其在不同的后续阶段发现并改正所需成本确定了一个范围。（参见表 2.1 和图 2-4）

表 2.1	
与修正一个错误相关的代价	
发现错误的阶段	成本倍数
需求阶段	1
设计阶段	3-6
编码阶段	10
开发测试阶段	15-40
应用测试阶段	30-70
实际运行阶段	40-1000

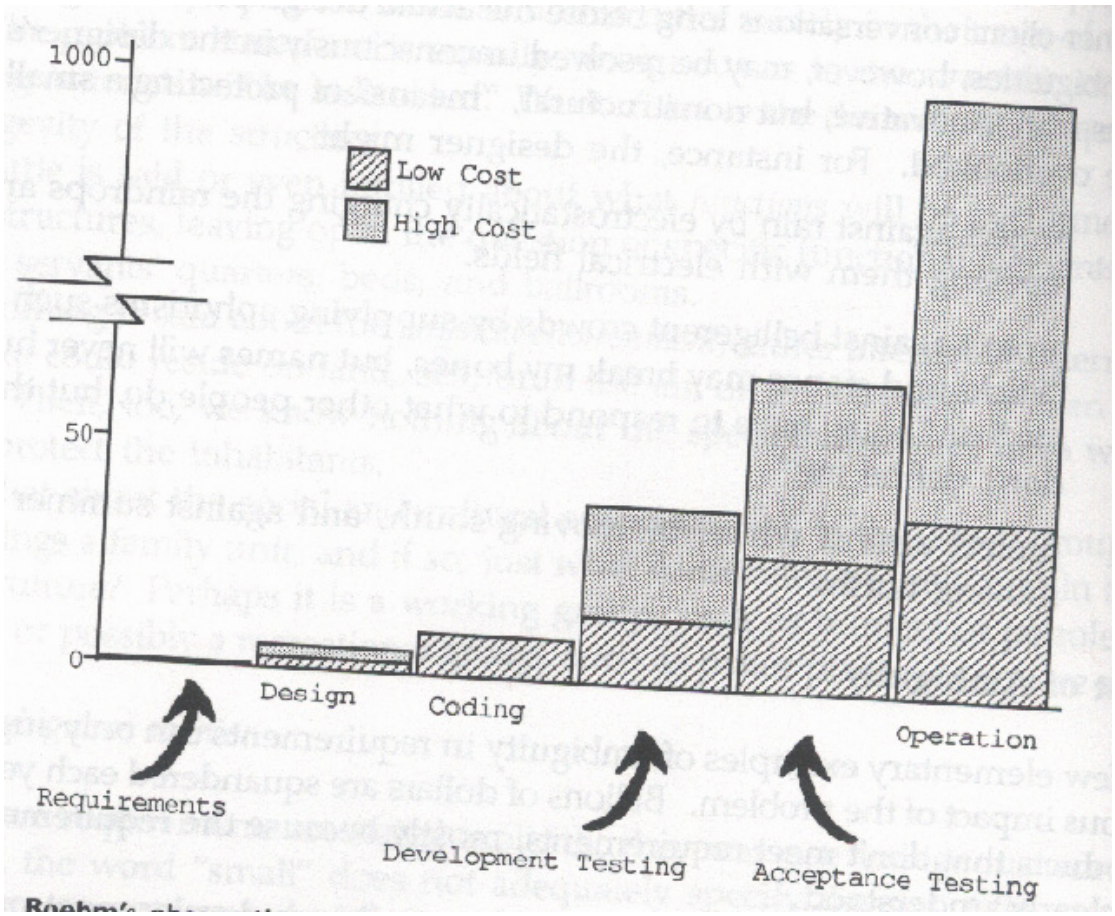


图 2-4 Boehm 在项目成本上的观察结果

尽管表 2.1 生动地表明了发现需求含混性的重要性，这些数据实际上可能还十分保守。首先，Boehm 研究的仅仅是那些完成的项目，而一些观察者估计大约有三分之一的大型软件项目从未完成。这些流产项目的巨大损失也可归结于其贫乏的需求定义。

在我们考虑基于早期错误假设的错误设计决策带来的灾难性结果，情况看起来更为糟糕。例如说，在二十世纪七十年代生产的福特斑马车（Ford Pinto），在假设没有任何尾部冲撞的条件下，燃油槽座架螺栓的位置是一个极其杰出的设计结果。这一假设被证明是错误的之后，福特汽车公司自己估计已经在诉讼和产品召回维修上花费了 1 亿美元。更何况还有那些因为这一错误而付出的生命又该怎么来估价呢？

来看另一个案例。由 Johns-Manville 公司开发、生产并销售的石棉建筑材料所基于的假设是：从环境角度看，在其产品中使用的石棉形态对所有裸露的人们是安全的。在 John-Manville 产品中的许多优良的观念都是基于这个现在大家都知道的错误假设。在麻省坎布里奇市的流行病研究公司估计这一高级决定最后带来了 5 万 2 千起诉讼事件，并给公司带来大约 20 亿美元的债务。实际上，该公司整个组织、文化和主要投资都投入到石棉材料产品中去了，于是被迫宣告破产，而现在已改组为 Manville 公司。

2.3 为消除含混性而探索

在过去的三十年里，我们两个一直从事于帮助避免代价昂贵的错误、失败项目和灾难性项目的工作，这些中的大部分都始于含混的需求。我们写作本书的目的就是为了向大家介绍一些成功用于抑制含混性的探索需求的工具。这是一些综合的方法，并可应用于几乎所有类型的设计项目：无论是在皮奥里亚市盖房子还是在巴伐利亚建城堡，或无论是在线信息系统还是生产型组织，或无论是广告竞争还是在新西兰的自行车旅行。

2.3.1 需求的图片

图 2-5 是一幅阐明了我们所谓的需求的含义的图片。古人们相信宇宙初期起源于一个巨大的蛋，所以我们使用“蛋”来表示所有可能的事物的总体。我们在蛋的中间画一条线来把我们所需要的内容从不需要的东西中分开。如果我们确实能够画出（或描述清楚）这样的一条线，就表示我们拥有了完美的需求描述。

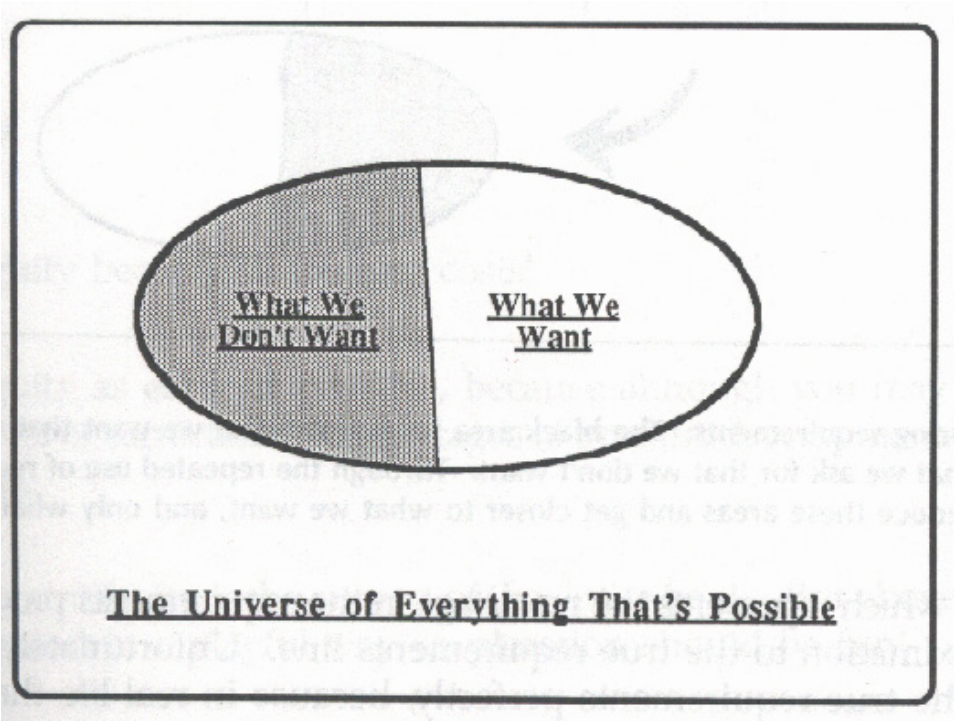


图 2-5 通过需求来表达我们的意图

The Universe of Everything That's Possible	所有可能的事物的总体
What We Don't Want	我们不想要的
What We Want	我们想要的

图 2-6 是一幅阐明了我们所谓的探索需求的含义的图片。第一个蛋上画了一条波动相当明显的分界线，它表示对需求线的第一次近似。这条线表达了对问题的第一次含糊的描述。第二个蛋展示了在一些早先的探索技巧取得的结果。这条线仍然是波浪型的，它表示仍然描述了一些我们不想要的内容，也还有一些我们想要的没有被描述在内。但是至少有一些最大的潜在错误（离需求线最远的区域）被消除了。

每个在需求过程中代表下一阶段的新蛋，都是对真实需求线的一次更好的近似。可惜，这条线将永远不可能和需求线完美重合，因为在现实生活中这几乎是不可能的任务。即便如此，通过探索，开发者应该努力在为其波动支付太昂贵的代价之前让它尽可能地接近笔直。

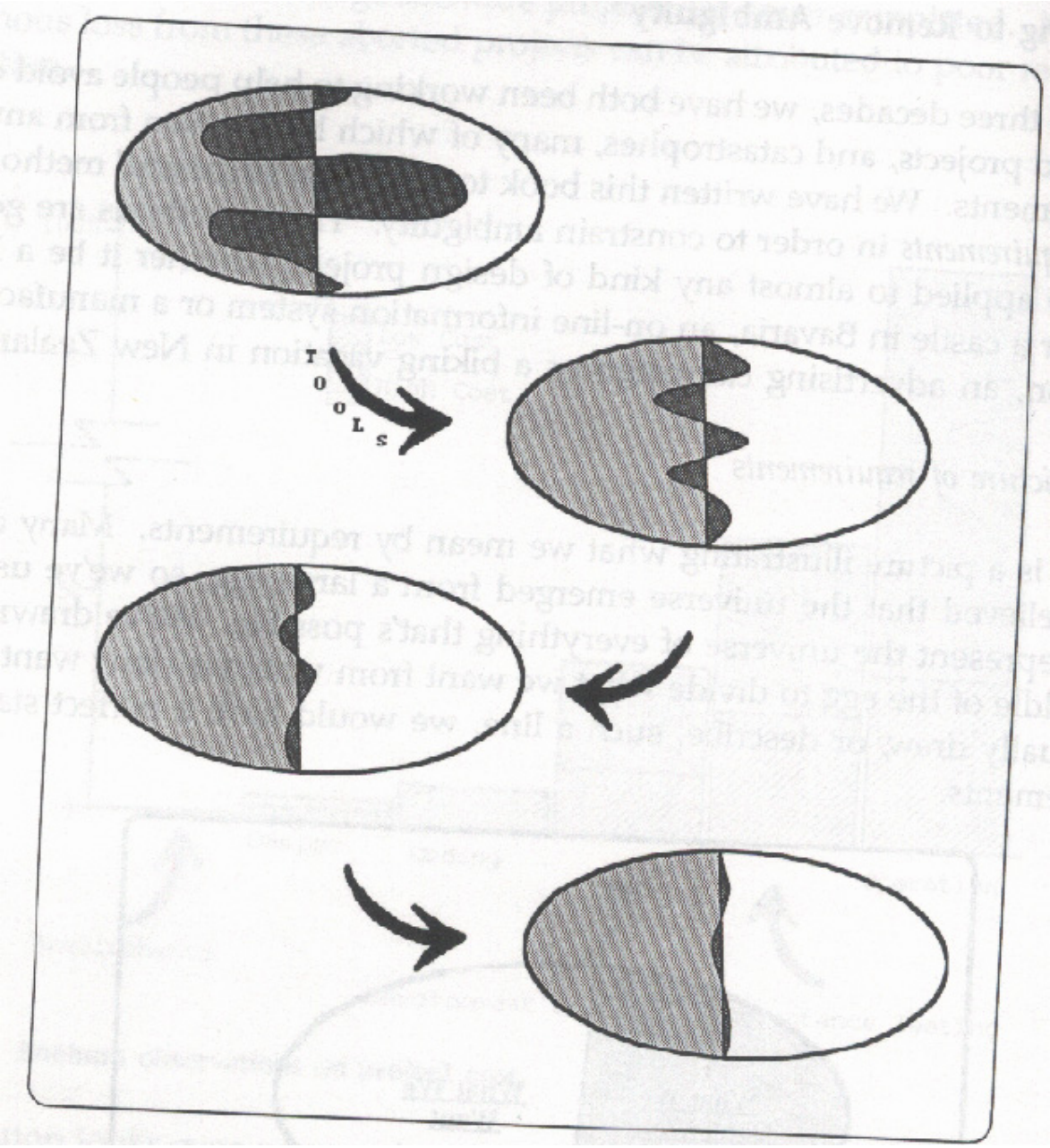


图 2-6 探索需求：黑色区域代表那些我们想要的内容却没有要求，或我们要求了并不想要的。通过再三使用需求工具，我们缩小了这部分区域，并越来越接近于我们不多不少想要的内容

2.3.2 需求的模型

拉直波浪线的过程是一种探索过程，在这方面，伟大的探险家马可波罗、哥伦布、刘易斯和克拉克（译者注：Marco Polo，马可波罗，1254-1324，意大利旅行家，商人；Columbus 哥伦布，1446?-1506，意大利航海家，据传于 1492 年发现北美洲；Lewis，刘易斯·梅里韦瑟，1774-1809，美国军人和探险家，曾率领刘易斯和克拉克探险队从圣路易斯出发横贯大陆抵达哥伦比亚河口；Clark，克拉克·威廉，1770-1838，美国探险家，曾参加刘易斯对太平洋的一次探险，他负责仔细标明每条道路）都是我们的榜样。概言之，下列步骤是探索者所要做的：

1. 向某个方向移动。
2. 看看在那里发现了什么。
3. 记录所发现的东西。
4. 有目的地分析所发现的东西。
5. 通过对所发现的东西的分析和记录来选择下一个方向。
6. 跳回到第 1 步，继续探索。

在本书的后续章节里我们将会看到，在探索需求中使用的过程是一样的。

2.4 有用的提示和变化

- 我们的同事 Ken de Lavigne 建议说，在 Deming 的著作《逃离险境》（作者底注：W. Edwards Deming 著，《逃离险境》，麻省坎布里奇市，麻省理工学院高等工程研究中心，1986。）中描述了一些关于需求含混性的结果的非常重要的例子，在“手术定义”的那个主题下。

- 除了看别人的例子之外，找一些你自己的实例会更好。下次如果你发现产品并不完全如你所想，你可以问问自己：“是什么需求导致了这样地创建一个产品？”

2.5 小结

为什么?

攻击含混性是因为含混性需要成本。

什么时候？

尽可能早地攻击含混性，因为即使你最终消除了它，在产品开发的早先阶段改正所需的成本要比以后改正的成本少很多很多。

怎么做？

如何攻击含混性是全书的主题。但首先，一定要记住用一种非常有趣的方法来使用你的智慧——探索应该是一种乐趣。

相关人是谁？

就是你自己。

（本草稿经清华大学出版社同意刊登）

❖❖❖

赞誉

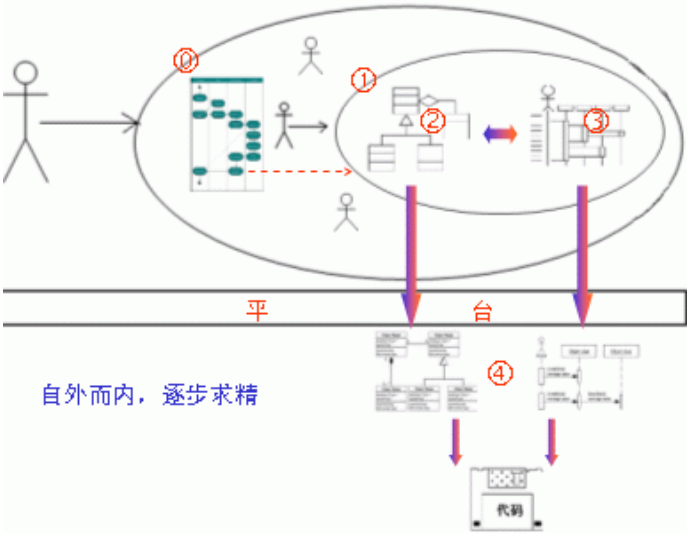
“高斯和温伯格……照亮了产品开发过程中重要而又最黑暗的部分：获得对需求的正确的理解。本书提出了一系列极好的原则，并通过贴切而又发人深省的实例对之进行充分的阐述。”——Barry Boehm, UCLA

“……一本出色的书——独特、发人深省而又有趣。这是任何从事需求过程的人员的必读书。”——Claude W. Burrill, President, Burrill-Ellsworth Associates, Inc.

“为系统分析员的觉悟提升而作。”

——Tom DeMarco, Principal, Atlantic Systems Guild

UML 应用实作细节（UML/UP 实作中阶）



训练介绍

现在，UML、RUP、Rose 等概念已经传播得越来越广，书籍、资料也越来越多，决定在开发过程中使用 UML 的开发团队也越来越多。

在开发团队应用 UML 的软件开发过程中，自然会碰到很多细节问题：“我这样识别 actor 和用例对不对？”、“用例应该多大？”、“用例文档这样写合适吗？”、“RUP 告诉我该出分析类了，可类怎么得出来啊？”、“先有类，还是先有顺序图啊？”、“类怎样才能和数据库连起来啊？”...许许多多的细节问题，而每一个细节都和背后的原理有关。

本训练秉行 UMLChina 一贯的“只关心细节”的原则，完全和受训团队当前项目结合，**【训练的过程就是指导团队用所授方法实作团队当前项目的过程】**，目标明确，效果明显。

学员对象

已经或准备在项目中使用 UML 的开发团队。

训练目标

通过大量练习和项目实践，使学员真正理解如何结合用例、类、顺序图等 UML 要素来开发软件的流程，能在项目开发中正确和灵活地应用。

知识准备

要求团队成员在接受训练之前要了解公司提供给 UMLChina 作剖析案例的项目。

团队成员如果已经有 UML 和面向对象的基本知识和 UML 工具的使用经验或事先作知识准备，会对训练效果有较大帮助，但这不是必要条件。如果因工作忙没有时间作知识准备，训练期间也有足够的练习让学员领会概念。

时间

2-3 天（12-18）学时，每天 6 学时。

训练内容安排

1. 概论（1 小时）

--软件开发方法演变

--面向对象方法

--UML 的统一和意义

--使用 UML 开发过程、工具、资料介绍

2. 使用用例组织需求（30%学时）

--识别系统边界和 actor

--讨论和练习，项目实践

--用例概念辩析和识别用例

--讨论和练习，项目实践

--正确书写用例文档

--讨论和练习，项目实践

--识别用例的关系

--讨论和练习，项目实践

--用例的排序和分包

4. 使用顺序图的动态分析（20%学时）

--顺序图精要

--讨论和练习

--用例、类图、顺序图的互动

--正确进行责任分配的原则

--讨论和练习，项目实践

5. 过渡到设计（20%学时）

--分析阶段工件的细化

--存储层的映射

--讨论和练习，项目实践

--数据服务层的映射

--讨论和练习

--架构概念及三层、N 层架构

--根据实际情况折衷

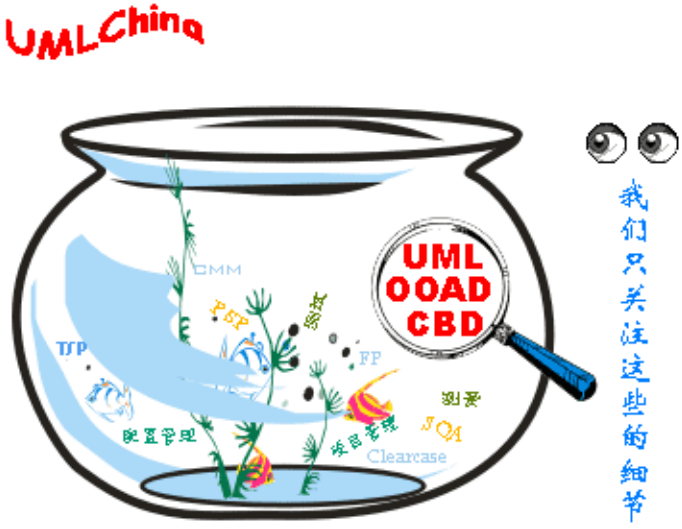
3. 使用类图的静态分析（30%学时） --从用例过渡到类图 --识别类及其属性 --讨论和练习，项目实践 --识别类之间的泛化 --讨论和练习，项目实践 --识别类之间的关联（聚合、组合、连接） --讨论和练习，项目实践	**6. 基于用例的业务建模技术** 此部分内容根据团队和项目特点在相应地方补充 **7. 状态图实践** 此部分内容根据团队和项目特点在相应地方补充 **8. 活动图实践** 此部分内容根据团队和项目特点在相应地方补充 **以上时间分配会根据项目特点和训练进程调整。**

后续训练

团队接受中阶训练后，经过一段时间的实作和思考，视情况可以选择以下服务：

1. UML/UP 实作高阶（1）：基于用例的业务建模和需求分析
2. UML/UP 实作高阶（2）：结合模式的面向对象分析设计
3. UML/UP 实作项目指导

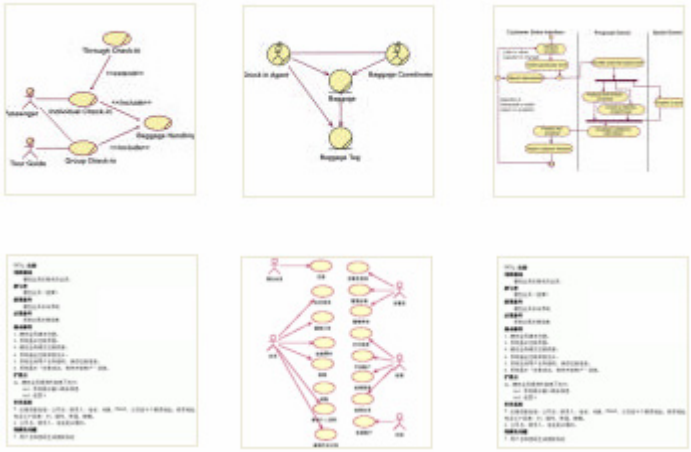
} 都是和团队项目密切结合



think@umlchina.com，umlchina1@hotmail.com

QQ: 3504847，电话: 0755-22093452/13925218052

基于用例的业务建模和需求分析（UML/UP 实作高阶）



介绍

以用例技术为核心，辅予其他 UML 元素，有效完成业务建模和需求分析。

本课程可以作为开发团队接受 UMLChina“UML 应用实作细节”训练之后的进阶课程，更进一步强调细化业务建模和需求分析实践中的相关细节和技能；专注于需求技术改进的团队，也可以单独选择本课程。

本课程秉行 UMLChina 一贯的“只关心细节”的原则，完全和受训团队当前项目结合，**【训练的过程就是指导团队用所授方法实作团队当前项目的过程】**，目标明确，效果明显。

学员对象

需要改进需求技术的开发团队。要求开发团队已经有在需求过程中使用 UML（或者已经接受过 UMLChina 的“UML 应用实作细节”训练）的经历。

训练目标

通过大量练习和项目实践，使开发团队真正理解在业务建模和需求分析中如何应用用例技术，能够在项目开发中正确和灵活地应用。

训练内容安排（2 天，12 学时）

1. 用例技术进阶（30%学时）

--什么是用例

--用例要素精解

--讨论和练习

--相关工具、资料介绍

2. 业务建模（30%学时）

--业务建模流程

--业务建模调研技术

--业务执行者、业务用例、业务工人

--讨论和练习，项目实践

--业务用例详细描述（文本，图形）

--讨论和练习，项目实践

--业务对象模型

--讨论和练习，项目实践

3. 需求分析（35%学时）

--需求分析流程

--需求分析调研技术

--从业务用例到系统用例

--讨论和练习，项目实践

--系统用例的详细描述（文本，图形）

--讨论和练习，项目实践

4. 用例与后续开发（5%学时）

--用例与分析设计

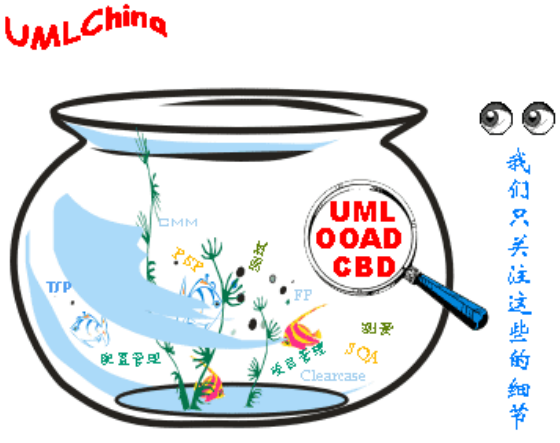
--用例与界面设计

--用例与测试

--需求管理

--讨论和练习，项目实践

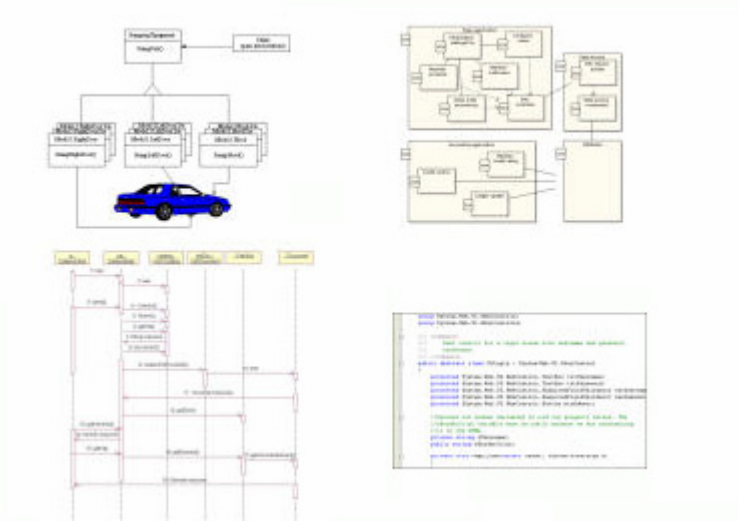
以上时间分配会根据项目特点和训练进程调整。



think@umlchina.com，umlchina1@hotmail.com

QQ: 3504847，电话: 0755-22093452/13925218052

结合模式的面向对象分析设计（UML/UP 实作高阶）



介绍

以对象技术为核心，结合分析模式和设计模式，有效完成软件的分析设计。

本课程可以作为开发团队接受 UMLChina“UML 应用实作细节”训练之后的进阶课程，更进一步强调细化面向对象分析设计实践中的相关细节和技能；专注于分析设计技术改进的团队，也可以单独选择本课程。

本课程秉行 UMLChina 一贯的“只关心细节”的原则，完全和受训团队当前项目结合，**【训练的过程就是指导团队用所授方法实作团队当前项目的过程】**，目标明确，效果明显。

学员对象

需要改进分析设计技术的开发团队。要求开发团队已经有面向对象分析设计的实际经历（或者已经接受过 UMLChina 的“UML 应用实作细节”训练）。

训练目标

通过大量练习和项目实践，使开发团队真正理解面向对象分析设计技术，能够在项目开发中正确和灵活地应用。

训练内容安排（2-3 天，12-18 学时）

1. 对象技术要素（15%学时）

--面向对象基本概念

--本课程所涉及的 UML 元素（类图、顺序图、状态图、构件图…）

--相关工具、资料介绍

2. 分析（40%学时）

--类对象的捕获，分析类图

--典型分析模式

--讨论和练习，项目实践

--顺序图和责任分配模式

--讨论和练习，项目实践

--状态图

--讨论和练习，项目实践

3. 设计（45%学时）

--从分析到设计

--数据层模式

--讨论和练习，项目实践

--业务层模式

--讨论和练习，项目实践

--表示层模式

--讨论和练习，项目实践

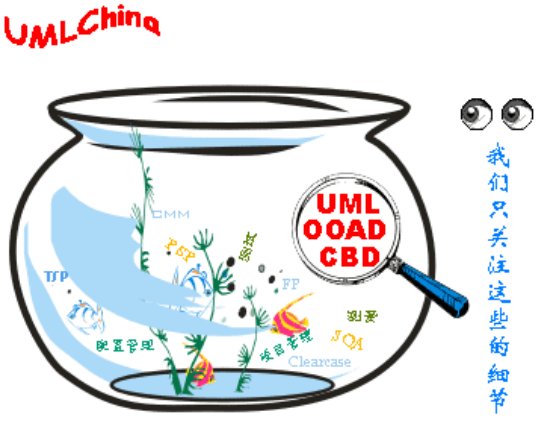
--子系统切割、构件

--讨论和练习，项目实践

--使用工具帮助生成代码

--N-tier 关键技术

以上时间分配会根据项目特点和训练进程调整。



think@umlchina.com，umlchina1@hotmail.com

QQ: 3504847，电话: 0755-22093452/13925218052

斐力庇第斯从马拉松跑回雅典报信，虽然已是满身血迹、精疲力尽，但他知道：没有出现在雅典人民面前，前面的路程都是白费。

学到的知识如果不能最终【用】于您自己的项目之中，也同样是极大的浪费。而这最后一段路最是艰难。

UMLChina 聚焦最后一公里，所提供服务全部与您自己的项目密切结合，帮您走完最艰难的一段路。