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。

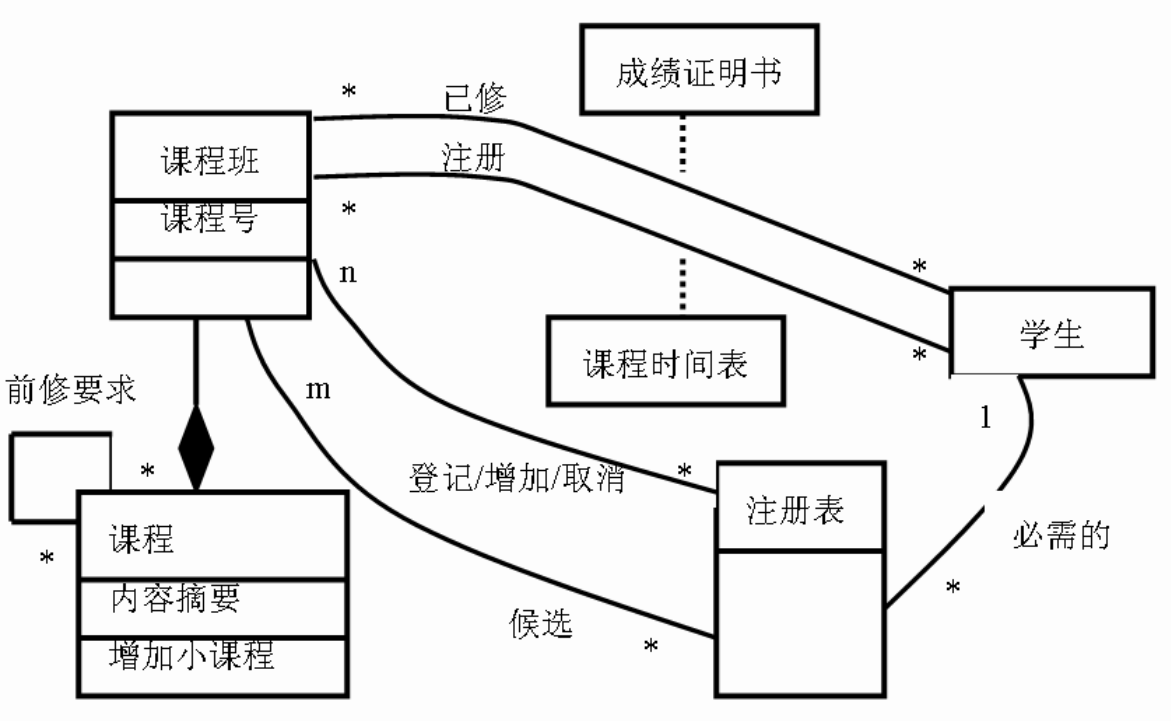


图 1　课程注册用例图

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。

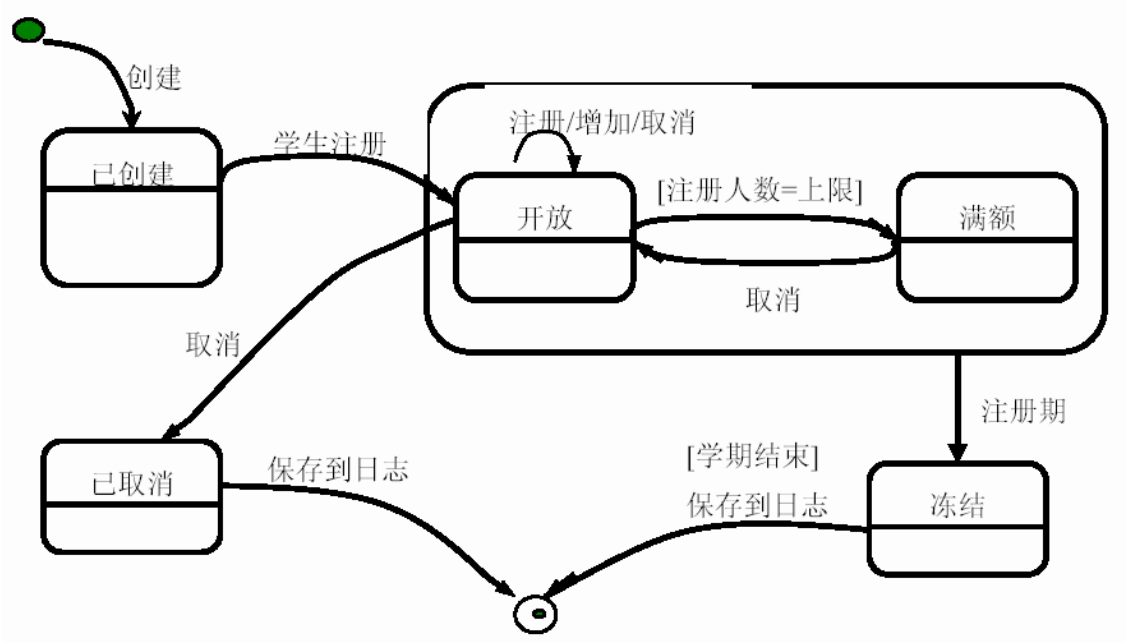


图2 课程班类状态图

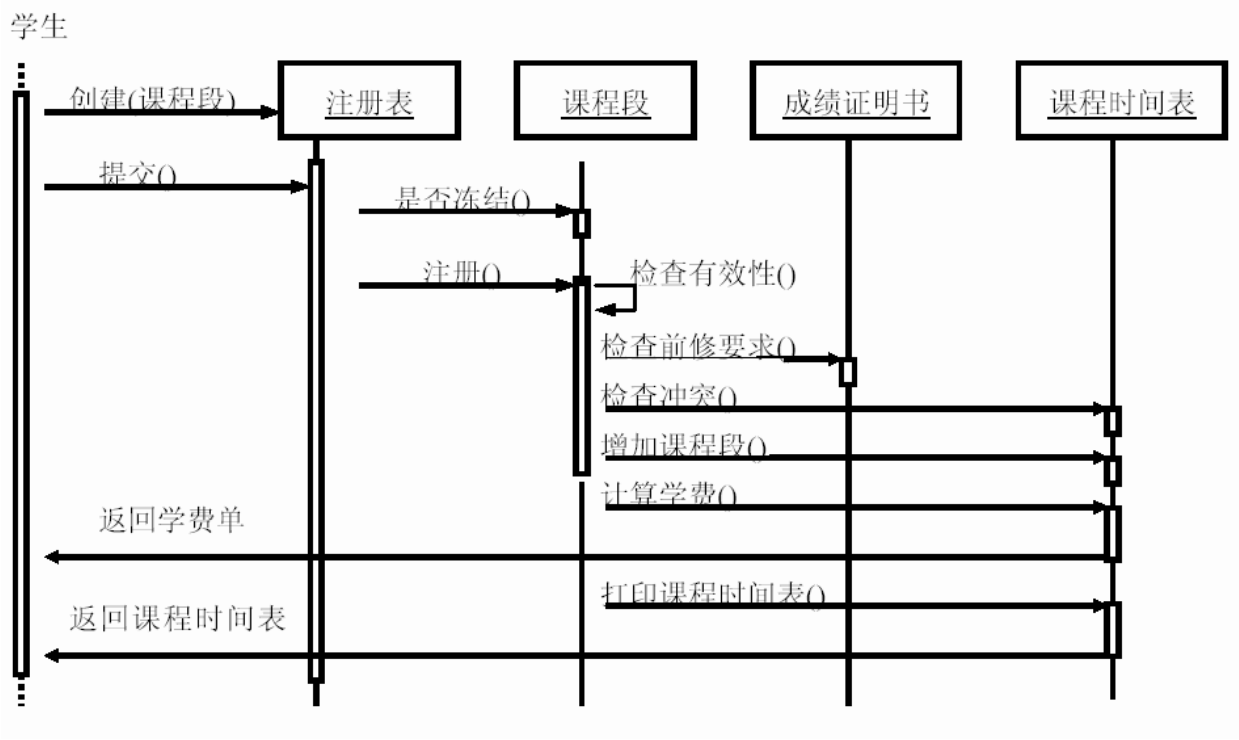


图3 注册课程序列图

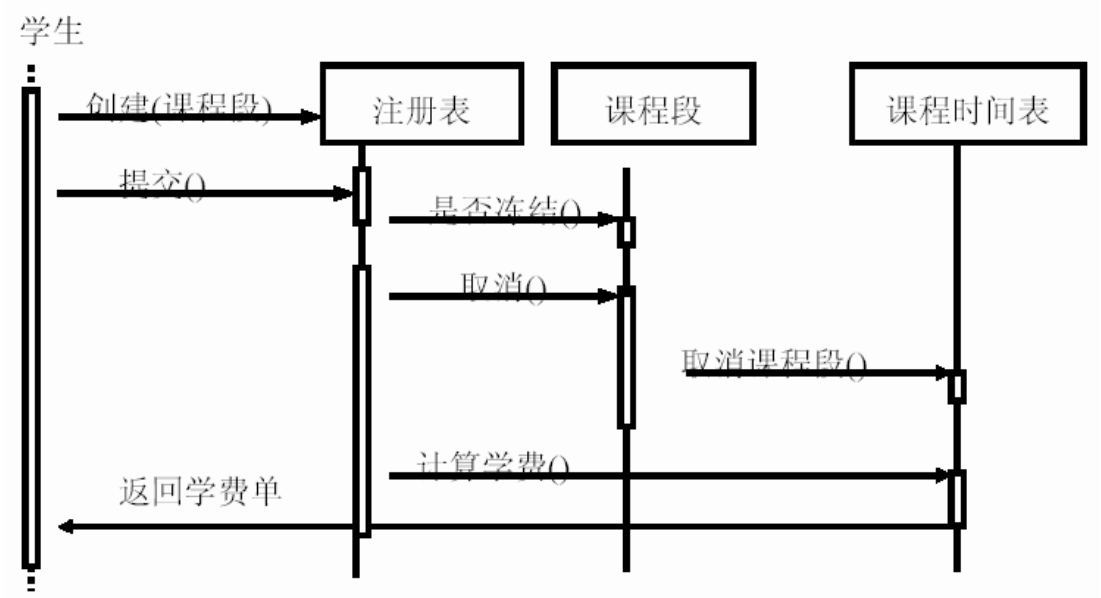


图4 取消课程序列图

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。

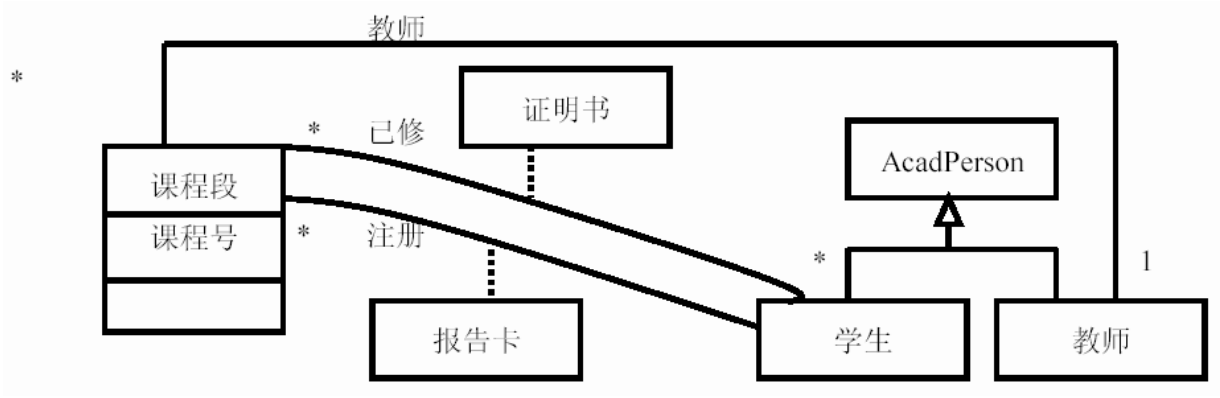


图 5 成绩管理实现类图

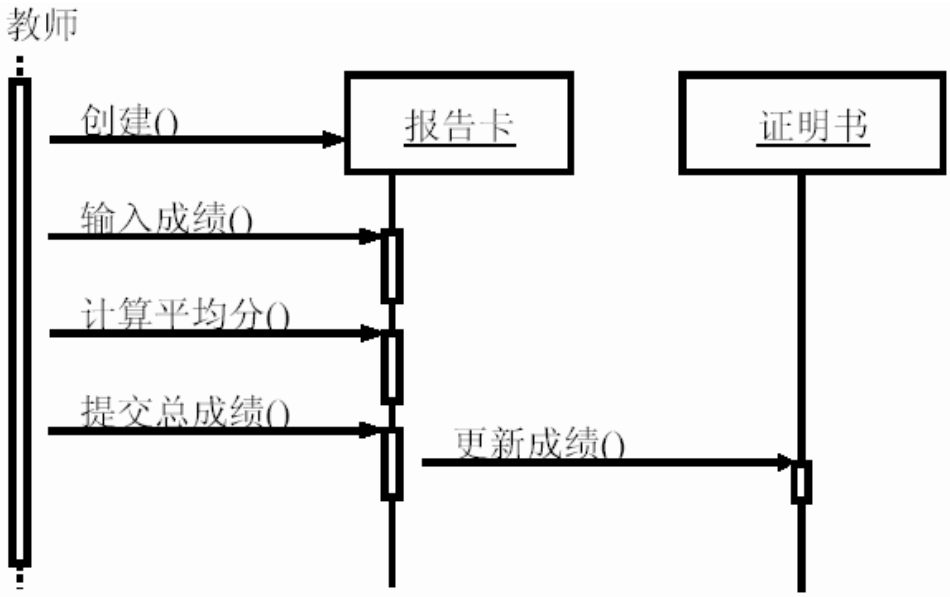


图 6 成绩评价序列图

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。

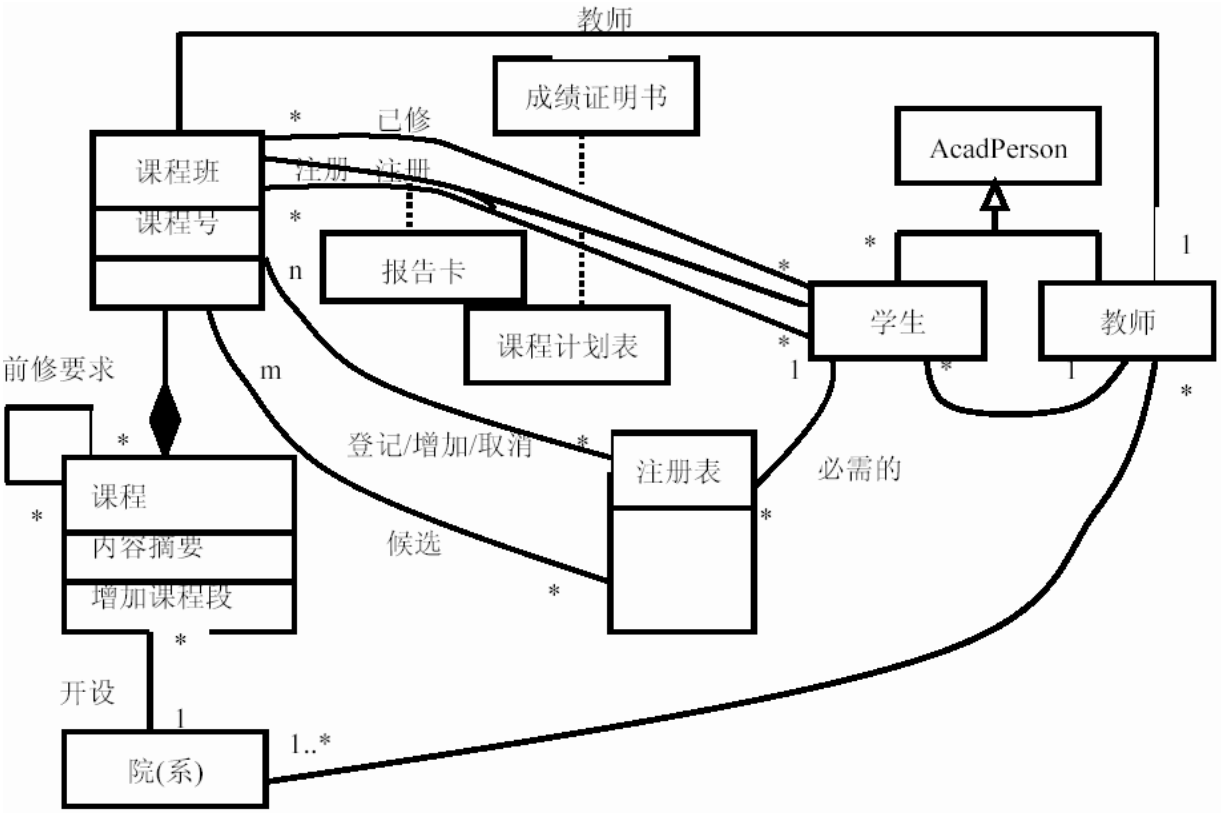


图 7 课程管理用例图

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.

更多的操作系统访问控制模式

Eduardo B. Fernandez、John C. Sinibaldi 著，droplet 译

查看评论

摘要

这篇文章描述了操作系统访问控制架构性的模式。这是对上一篇文章[Fer02]中所描述模式的一个补充。这些模式控制主体对资源的访问（资源被表示为对象），其中包括认证模式、进程创建模式、对象创建模式和对象访问模式。

简介

我们这里描述操作系统中的访问控制模式。这些模式是对[Fer02]中所介绍模式的补充。在那篇文章中描述的模式如下：

- **文件访问控制模式**。怎样在操作系统中控制对文件的访问？将授权模式应用到主体对文件的访问过程中。被保护的对象现在是一个文件构件，它可以是一个目录，也可以是一个文件。
- **受控的虚地址空间模式**。怎样控制进程根据一个预定义的访问类型集合去访问它的特定虚地址空间？将虚地址空间划分成与程序逻辑块相对应的段。用一个特殊的描述符描述对这些段的访问权限。
- **引用监视器模式。**当一个进程访问对象时怎样强制它必须通过授权？定义一个抽象的进程，它截获所有对资源的访问，并检查它们是否符合授权要求。
- **受控的执行环境模式。**如何为进程定义一个执行环境？在每个进程上附加一组描述符表示权限，并用引用监视器强制进行访问控制。

在这篇文章中，我们增加以下模式：

- **认证器模式。**怎样确认主体的身份？使用单个访问点接受主体与系统的交互并用相应的协议确认主体的身份。

● **受控进程生成器模式。**怎样定义新创建进程的权限？在进程创建过程中定义它。

● **受控对象工厂模式。**进程创建新的对象时如何确定它的权限？当进程请求一个工厂去创建新对象，这个请求里包含了新对象的特性。在这些特性中包含了对象的访问权限。

● **受控对象监视器模式。**怎样控制主体对对象的访问。用一个引用监视器可以截获所有进程的访问请求。引用监视器检查进程是否有访问对象的权限。

假设资源被表示成对象（这在现代操作系统中非常普遍）。图 1 展示了这些模式所组成的模式语言。比如，认证在文件访问和受控对象访问中是必须的，当一个主体被授权通过某种方式访问某些对象，我们必须确认请求者不是入侵者。其他三个模式完成了受控执行环境的定义，现在对象的创建和访问都是受控的。这个语言同时也表明文件访问被引用监视器所控制。

背景

操作系统是计算机中提供安全保护的一个基础设施。操作系统支持应用程序的执行，所以每一层上的安全限制都应在操作系统中得到加强。操作系统同时要保证自身的安全，因为任何的妥协都会使它的用户帐号或是文件里的数据被非法访问。一个脆弱的操作系统使得黑客不但能够访问操作系统文件里的数据，而且可以访问那些使用操作系统服务的数据库系统里的数据。操作系统通过使进程相互独立来保护进程和与进程相关的文件里的数据[Sil03]。为达到这个目的，操作系统控制对诸如内存地址空间、I/O 设备等资源的访问。许多操作系统采用访问矩阵作为安全模型。一个访问矩阵定义进程（一般成为主体）对特定资源（资源在现代操作系统中常被表示为对象）可以进行何种类型的访问。使用这个模型时，我们必须确定主体在访问资源前是否通过了认证（使用 Authenticator），我们也需要在创建进程时确定进程所具有的访问权限（使用 Controlled-Process Creator），同时需要将进程放在一个受控的执行环境中执行，在这个环境中进程不能做超越它权限的动作（Controlled Execution Environment）。我们也需要定义新建对象的访问权限（Controlled Object Factory），同时控制在执行过程中对对象的访问（Controlled Object Monitor）。后者通过截获访问请求并检查它们的授权信息来达到访问控制的目的。所有这些功能就是在这两篇文档中所介绍的模式所要完成的。

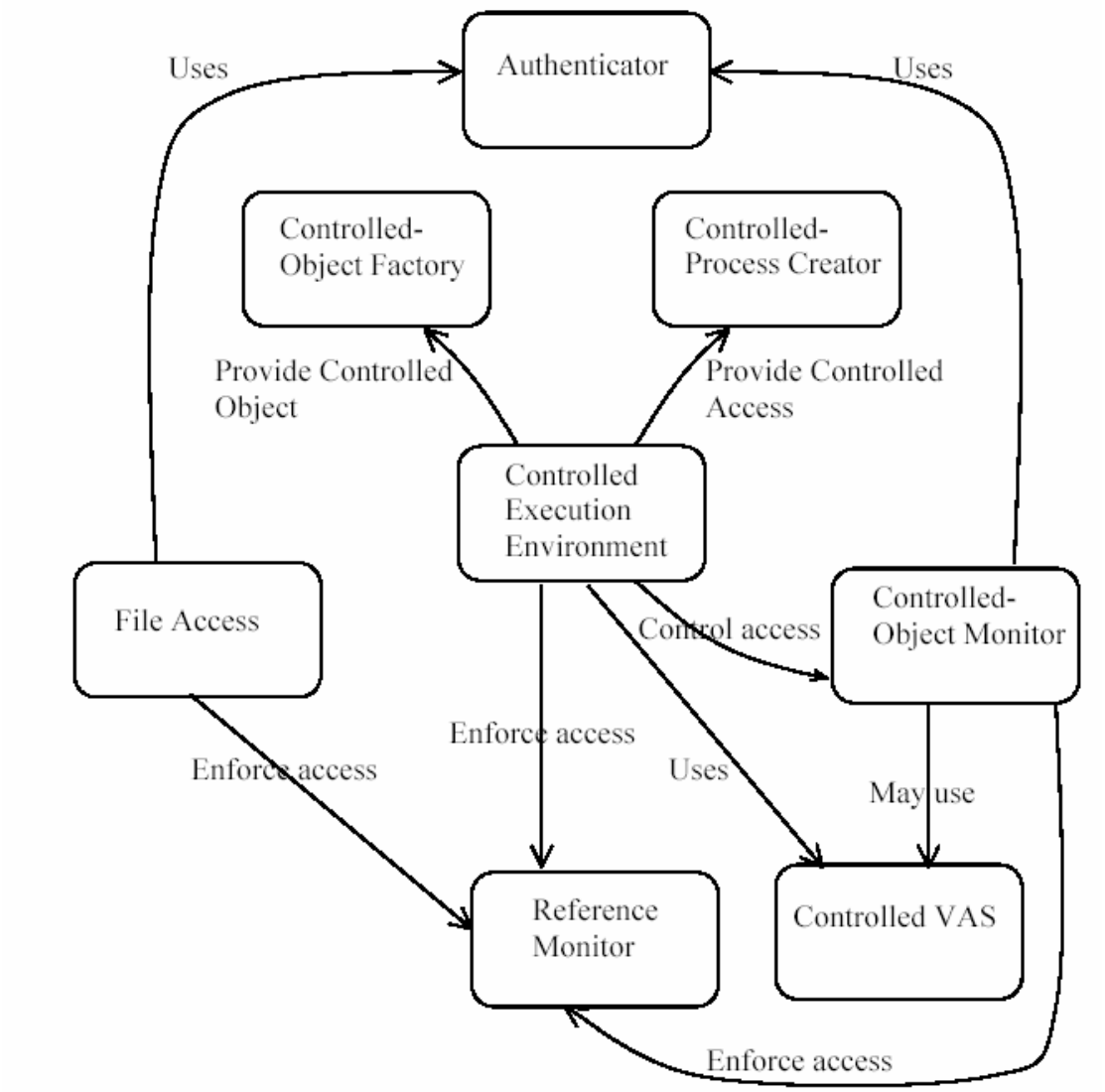


Figure 1. O.S. Access Control Pattern Language

图 1 OS 访问控制模式语言

操作系统在用户第一次登录时认证用户（可能在他访问特定资源时还要求再次认证）。然后用户执行一个程序，这个程序可以由多个并发的进程组成。进程一般通过操作系统提供的系统调用来创建。一个进程请求操作系统创建子进程的同时赋予这个子进程访问某些资源的权限。应用程序在运行时可能会创建一些对象。有些对象在程序初始化时创建，有些在执行过程中动态创建。进程对对象的访问权限必须在进程创建时确定。应用程序也可能需要诸如 I/O 设备这样的资源或者其他一些从资源池中得到的资源。在这些资源创建的时候，相应的权限就应该分配给相应的应用程序。这些权限被定义成授权规则或策略，并在进程访问资源时被检查。这就意味着我们需要截获每个访问请求，这可以由引用监视器来完成。

认证器（Authenticator）

意图

怎样确定用户的身份？

上下文

当主体，通常是一个用户，发起请求时，操作系统创建一个受控的会话。然后，认证用户（表现为用户执行的程序）就可以根据他的权限去访问资源。访问敏感资源时可能需要再一次认证。在分布式系统中如果访问外部节点的资源，需要再一次认证。

问题

怎样才能防止非法用户访问我们的系统？一个恶意的攻击者伪装成一个合法用户去访问资源。如果被伪装者具有比较高的权限，事情就非常严重。

约束

这个模式有以下的约束条件：

- 不同的用户要求不同的认证方式。我们必须能够处理所有的认证方式，否则就会造成系统的安全隐患。
- 我们需要以一种可靠的方式来认证用户。这就意味着我们需要可靠的协议，同时能够保护认证的结果。否则，用户可能跳过认证步骤，或者修改认证结果，使得系统暴露在安全攻击之下。
- 在安全和费用之间需要权衡，越安全的系统，花费也就越多。
- 如果认证是一个频繁进行的动作，性能需要注意。

解决方案

使用单个访问点来接受主体对系统的访问，同时使用协议来识别主体的标识。这个协议可能使用用户的输入作为标识，也有可能是其他更复杂的方法。图 2 描述的是这个模式的类图。主体，通常是用户，请求对系统资源的访问，认证器接收这个请求并根据认证协议认证用户。如果认证成功，认证器创建一个可信标识（这有可能是一个令牌，或是其他的东西）。

动态图

图 3 描述了认证的动态过程。用户请求到认证器。认证器通过认证协议确认用户信息，同时创建一个可信标识。用户取得一个可信标识的句柄。

变体

Single Sign-On

Single Sign-On（SSO）是一个过程，在这个过程中主体的安全标识可以在多个安全域中使用，并且可以多次使用。认证的结果就是用户取得一个安全令牌，在后续的访问中这个安全令牌就代表了认证用户。

PKI Authenticator

公钥加密是一种常用的标识用户的方法。它的认证过程与图 2（图 4）所示的模式过程类似。一个认证类使用包含公钥签名的证书进行认证。认证的结果是用户取得一个可以在后续访问中使用的认证令牌（这同样是 SSO 的一个变体）。

已知应用

- 许多商业操作系统使用密码来认证用户。
- RADIUS 使用集中认证服务来认证网络用户或分布式系统的用户。
- SSL 认证协议使用 PKI 来认证。
- SAML，一个安全 WEB 服务，定义了它在实现 SSO 体系时的应用[sam]。

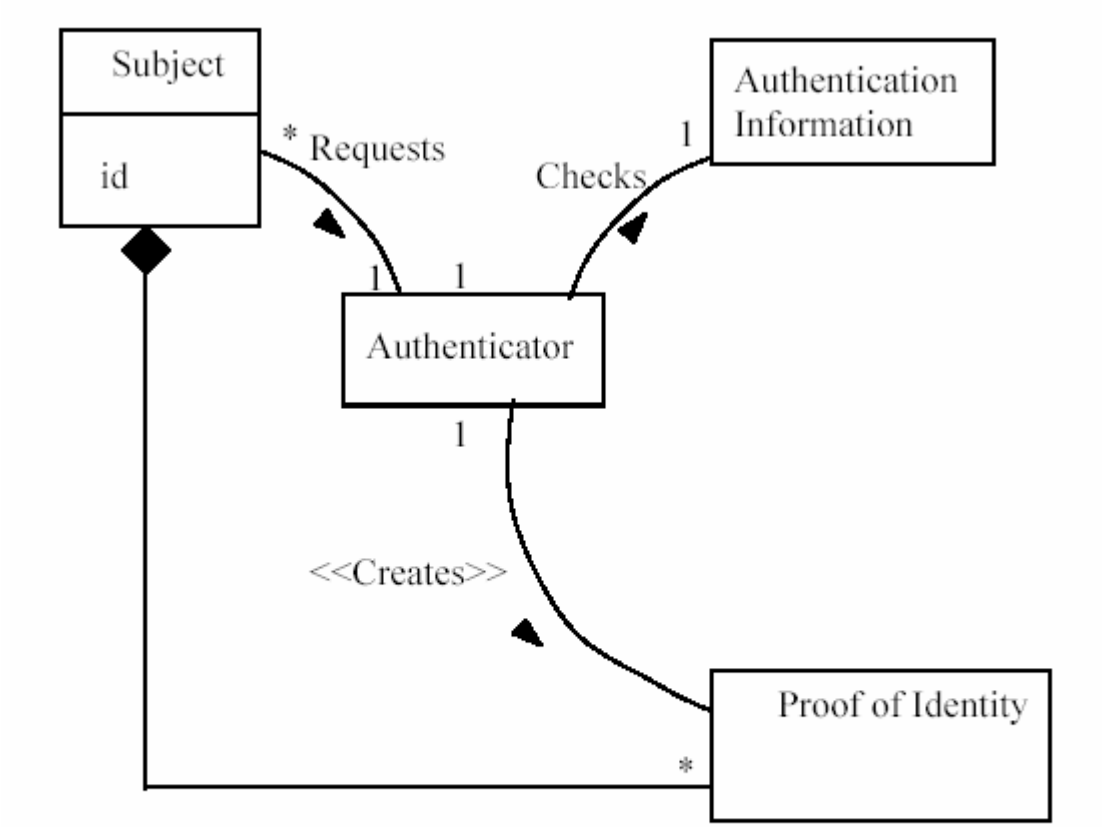


Figure 2 Authentication Pattern

图 2 认证器模式

结论

这个模式有以下几个优点：

- 根据不同的协议和不同的认证信息，我们可以以不同的方式认证不同的用户。
- 因为认证信息是分离的，我们可以把它存储在一个保护的区域内，所有的主体对它最多有读权限。
- 我们可以使用不同强度的算法和协议。这要在安全和花费之间权衡得失。有三个因素需要考虑：用户知道的信息（用户密码），用户有的信息（用户的 ID），用户是什么（用户的身份）。

- 认证可以是集中式的，也可以是在分布环境中。
- 我们可以生成一个安全的用户 ID，这个 ID 可以用在后续的认证中。这样会提高性能。

（可能）有以下几个缺点：

- 认证过程需要耗费一些时间。
- 系统的复杂性和花费随着不同的安全级别将会增加。

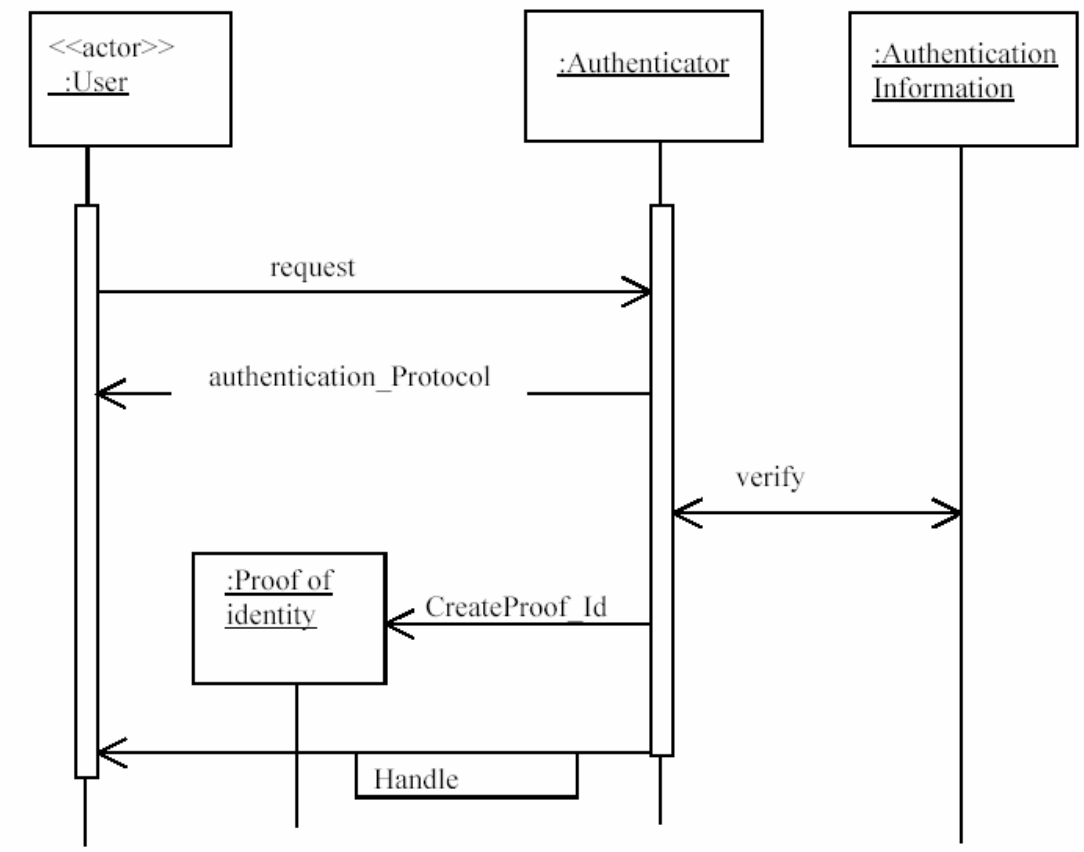


Figure 3 Authentication Dynamics

图 3 认证器的时序图

相关模式

分布式认证器[Bro99]讨论了认证器在分布式环境中的实现。

分布式过滤和权限控制框架包括了认证器[Hay]。

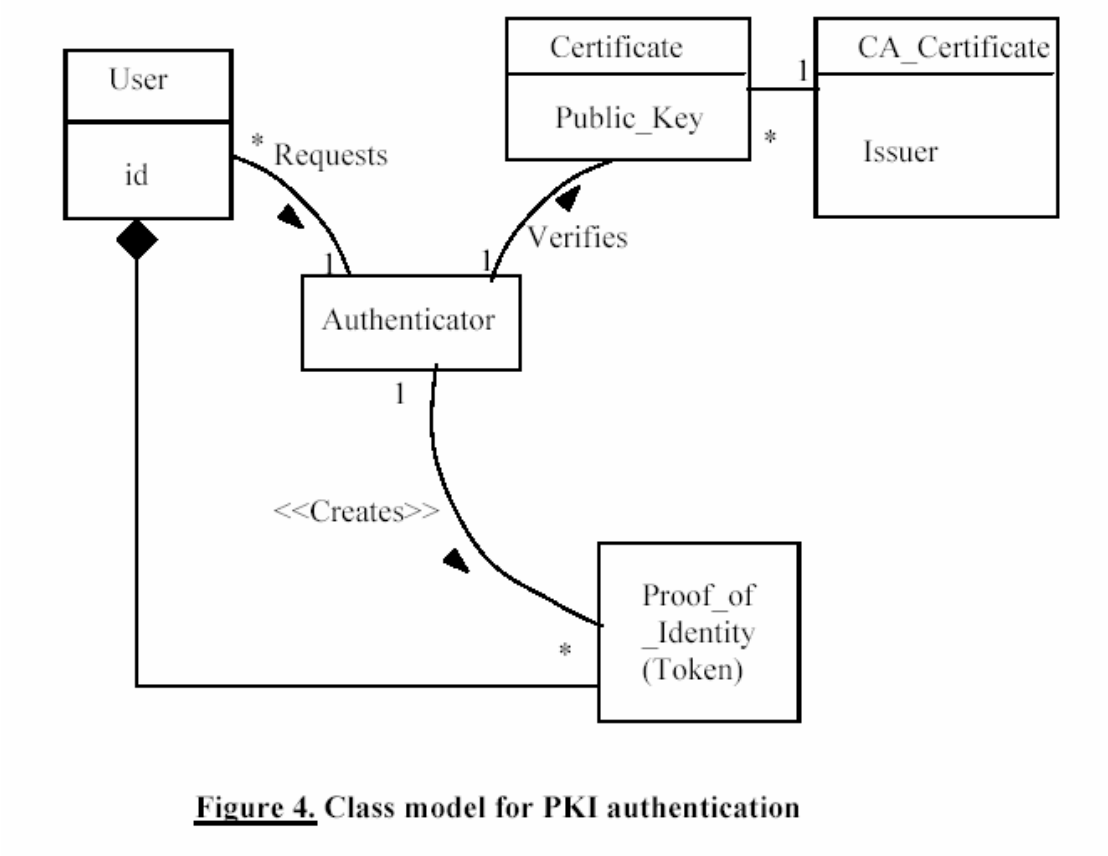


Figure 4. Class model for PKI authentication

图 4 PKI 认证器的类图

可控进程生成器

意图

定义合适的权限并将它赋给新创建的进程。

上下文

操作系统中的进程和线程需要根据应用的需求而创建。

问题

计算机系统中有许多进程或线程。进程根据应用程序的需求而创建，并且操作系统也是由许多进程组成。如果进程不受监控，那么它们可以相互交互，也可以非法访问对方的数据。

所以它们对资源的访问权限必须根据正确的策略来定义，如仅知所需策略。许多情况下进程需要进行的动作可能超越它的权限，比如访问那些它没有权限访问的文件。

约束

这个模式有以下的约束：

- 必须有方便的方式选择策略去定义进程权限。没有根据策略定义权限可能导致相互矛盾的情况出现和没有系统化的访问控制，这样可能会使得权限失去作用。
- 子进程在某些情况下可以伪装成父进程，但是这样伪装必须严格控制，否则一个脆弱的子进程可能泄漏信息或者破坏数据。
- 进程创建子进程的数量应该受限，否则可能引起拒绝服务攻击。

解决方案

由于新进程是通过系统调用是或者发送给操作系统的消息而创建的，我们就有机会决定新进程的权限参数。典型的，操作系统象创建子进程一样创建新进程。有几种策略确定赋给子进程的权限。一种策略是子进程可以继承所有或部分父进程的权限。另一种策略是父进程将自己权限的一个子集赋给它的子进程（更安全的一种方式）。

图 5 展示了这个模式类图。受控进程生成器控制操作系统新进程的创建。创建请求包含了父进程定义的子进程的权限。这些权限是父进程权限的一个子集。

动态图

图 6 展示了进程创建的动态过程。一个进程请求创建新进程。访问权限被包含在创建请求中传递给操作系统。

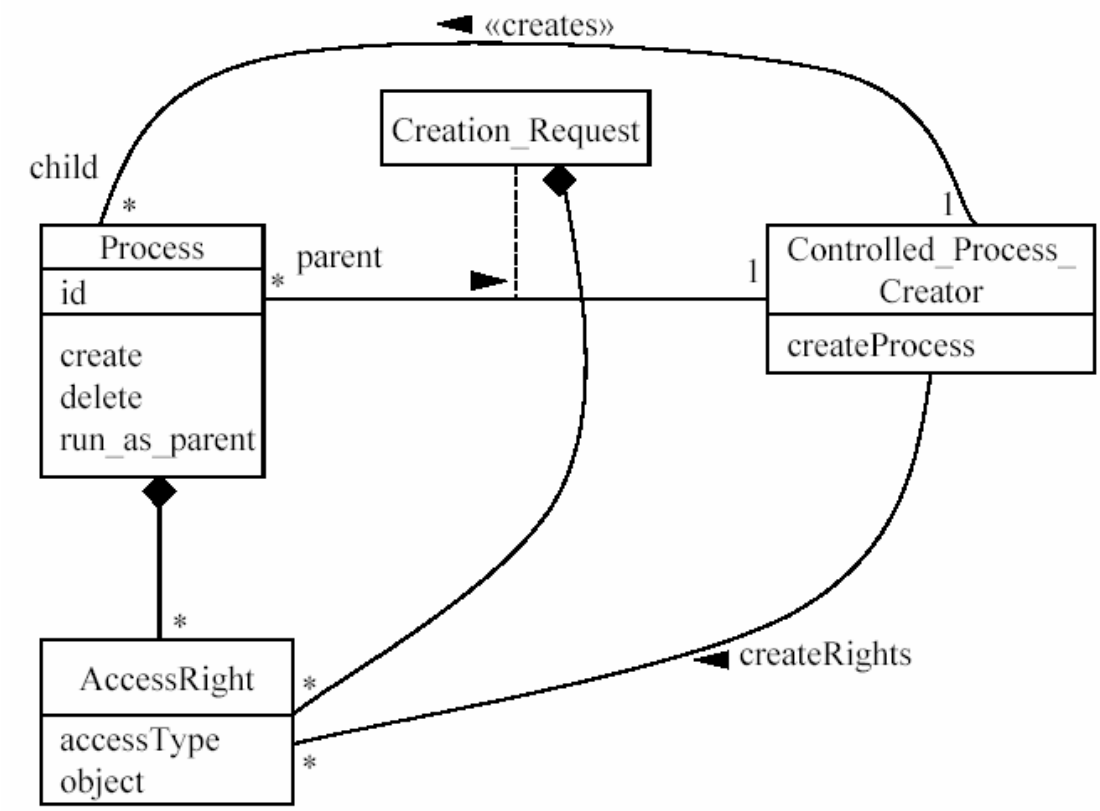


Figure 5. Class diagram of Controlled-Process Creator

图 5 受控进程生成器类图

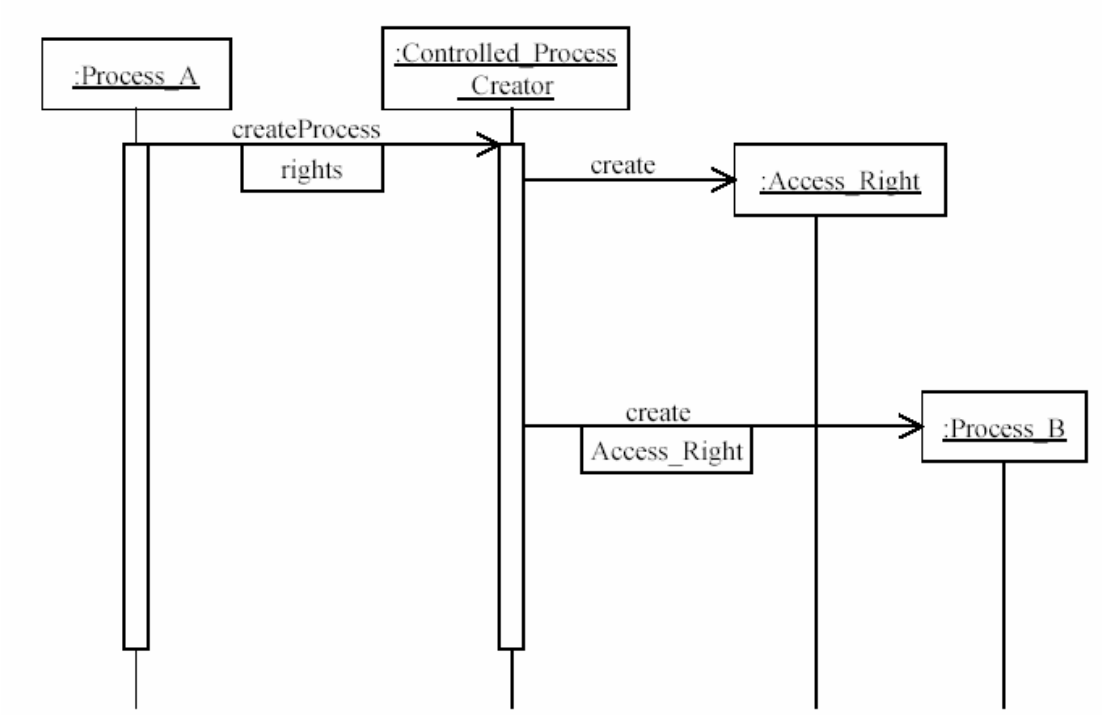


Figure 6. Child creation dynamics

图 6 子进程创建的时序图

已知应用

在许多操作系统中，例如，UNIX，权限是继承自父进程权限的一个子集。一些强化的系统，如Hewlett Packard's Virtual Vault不允许继承，它为每一个子进程重新定义权限集。

结论

此模式有以下的优点：

- 进程可以根据不同的权限策略创建。
- 创建子进程的数目可以控制。这可以防止拒绝服务攻击。
- 权限集可能包括父进程的进程ID，这样可以允许子进程以父进程的权限集运行。

受控对象创建器模式

意图

对象为特定的目的而创建，如果允许其他进程访问它，就必须在它创建时指定这些权限。

上下文

计算系统需要根据新创建对象的重要程度来控制对它们的访问。

问题

在计算环境中，进程会在运行过程中创建对象。有些对象在程序初始化时创建，而其他对象在进程运行时动态创建。进程对这些对象的访问权限需要在进程创建时就指定，否则进程可能会误用它们。

约束

- 应用程序创建不同类型的对象，但是我们需要根据它们的权限以一致的方式去访问它们。当然，使用标准的安全策略相对比较困难。
- 我们允许从资源池中创建对象，它们的权限可动态指定，这里的权限当然不是固定不变的。
- 应该有特定的策略来定义谁可以访问新对象，当我们定义对象的访问权限时就可以使用它。这是一个基本的安全要求。

方案

当进程通过工厂创建一个新对象时，创建请求包括了对象各种参数。在这些参数中，有定义主体对创建对象的访问权限。

动态图

图9显示了对象创建的动态过程。

结论

这个模式有以下的优点：

- 可以根据对象的重要程度定义对它们不同的访问权限。
- 从资源池中创建的对象可以动态添加其权限。
- 操作系统可以使用所有者权限，例如，对象的创建者可以拥有对所有它创建对象的访问权限。

已知应用

WIN32 API允许进程调用不同的系统调用创建对象，在调用时进程会传递一个包含权限信息的结构引用。对象创建时，内核就把相应的权限与它绑定。内核返回给调用者一个句柄用于访问对象。其他操作系统使用预定的权限；比如在UNIX操作系统中，与所有者同组的其他用户拥有对文件相同的组权限。

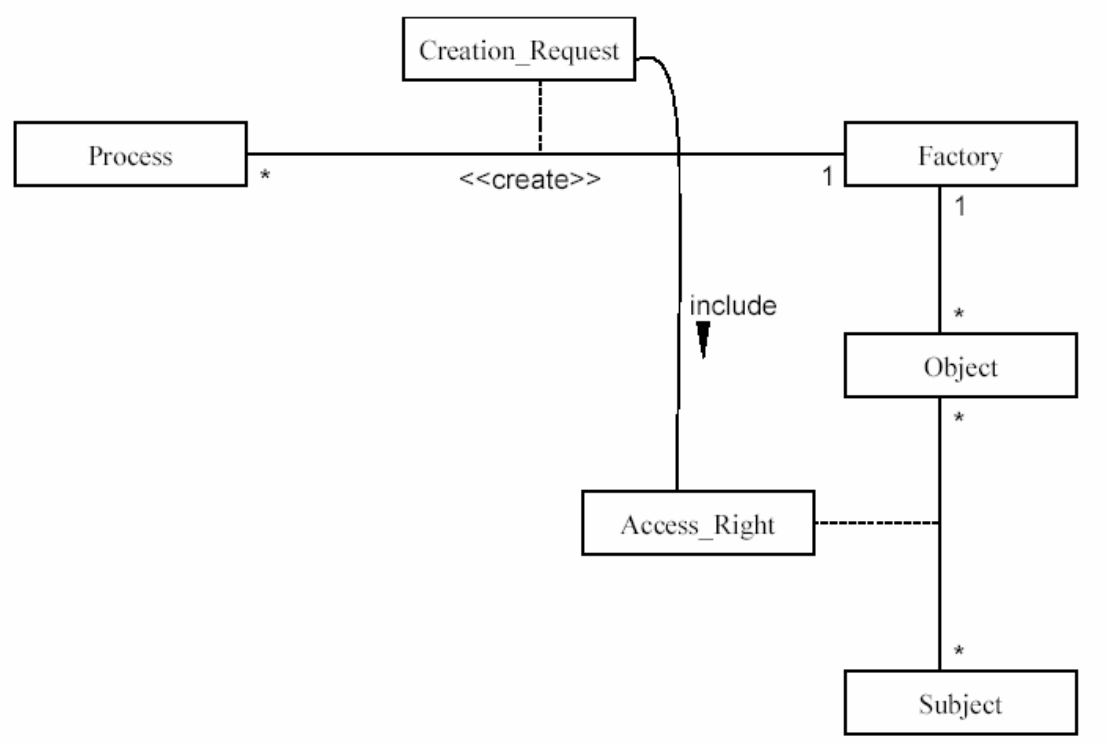


图8 受控对象创建器模式类图

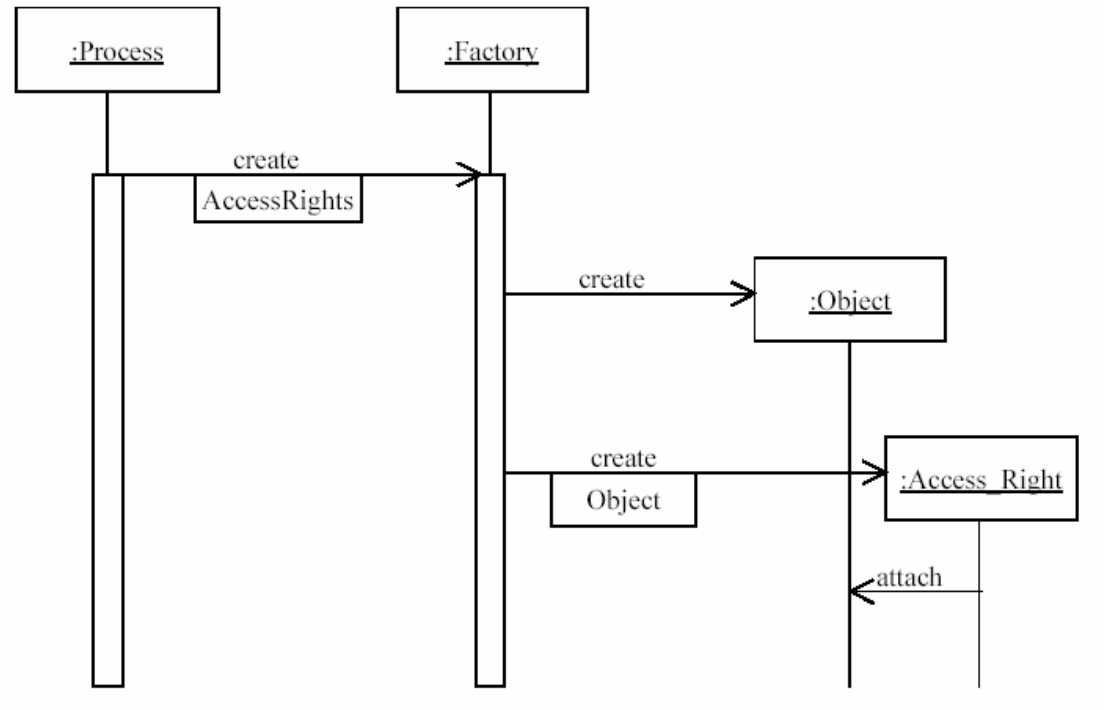


图9 对象创建时序图

受控对象监视器模式

意图

如何控制主体对对象的访问。

上下文

操作系统包含不同权限的对象。

问题

当对象创建时，我们定义进程对它们的访问权限。当进程访问它们时需要强制使用授权规则或是策略。

约束

有以下的约束条件：

- 不同的授权策略定义了不同的访问限制；当进程访问对象时我们必须强制使用这些限制。

- 我们需要控制不同类型的访问，否则对象可能会被误用。

解决方案

使用引用监视器截获进程的访问请求。引用监视器检查进程访问请求的权限是否符合访问规则。

图10显示了这个模式的类图。更具体的引用监视器模式的实现参见[Fer02]。那里的改进展示了系统如何将安全对象和规则联系在一起。

动态图

图11显示了安全主体访问安全对象的动态过程。访问请求发送给引用监视器，由引用监视器检查访问规则。如果规则允许，内核处理请求并向主体返回结果。这里需要注意的是，返回结果中包含了对象的句柄或令牌，对安全对象的后续访问可以直接进行而不需要再检查权限。

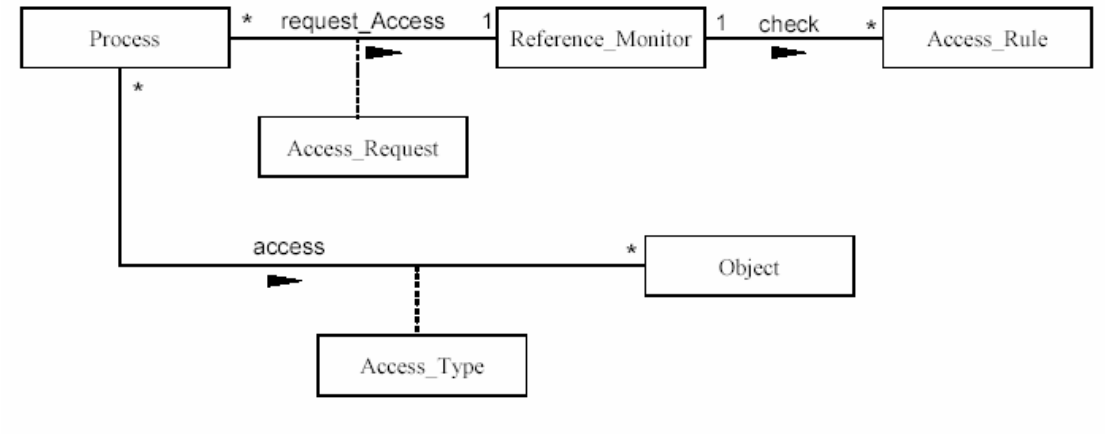


图10 受控对象监视器模式类图

结论

此模式有以下优点：

- 访问规则可以用权限矩阵实现，这样可以对每个主体都定义不同的权限。
- 每个被截获的访问请求都会根据授权规则允许或拒绝。

（可能）有以下缺点：

- 需要保护授权规则。
- 每个访问都需控制，这增加了系统的负担。

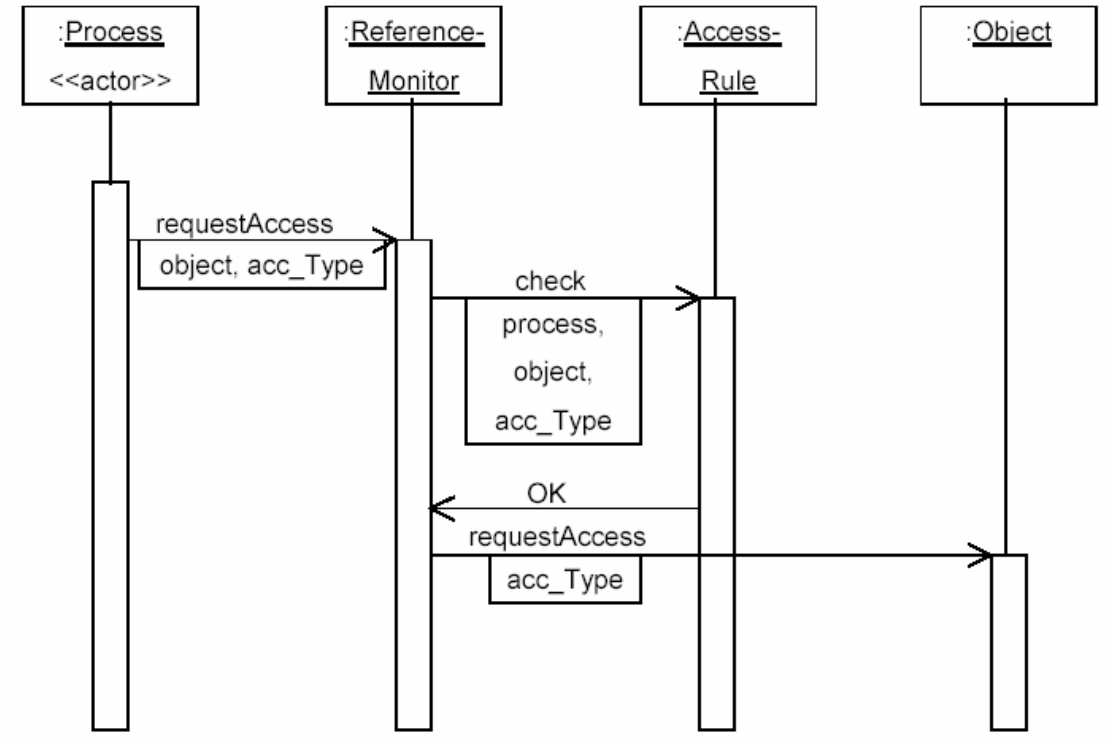


图11 验证访问请求的时序图

实现

一种可能的实现是：在用户登陆时验证用户。新建对象继承了包含在令牌中的原始用户的ID。这个与用户进程相关的令牌决定访问权限。只有通过这些受控的方式才能访问安全对象。

每个用户想要访问的对象都可能有相应的访问权限表。它描述了用户对对象有什么样的访问权限。它同样描述了系统中的其他对象有什么权限。通常，权限可以是允许或拒绝。这就是在Windows环境中常见的访问控制实体（ACE）[Har01,Mic00,Zac99]。这些实体的集合就是在Windows或其他操作系统中常说的访问控制列表（ACL）。

另一种与ACL相反的方式是能力（capabilities）集。一个能力对应访问矩阵中的一行，而ACL与对象相关（访问矩阵中的一列）。能力显示主体有权力访问安全对象。能力可以附带一些认证特征，这样对象可以信任主体提供的能力信息。一个全局表的每一行代表了每个认证用户的能力。或者能力集可以实现为一个用户列表，它包含用户对那些对象有访问权限[Kin01]。

模式的综合使用

下面的例子综合使用了以上的模式：

Windows NT

Windows NT安全子系统使用了上面描述的模式。它包括三个组件[Har01，Kel97，Mic00]：

- 局部安全授权（LSA）
- 安全帐号管理器（SAM）
- 安全引用监视器（SRM）

局部安全授权（LSA）和安全帐号管理器（SAM）配合起来认证用户并创建用户的访问令牌。安全引用监视器运行在内核中提供强制的访问检查。当请求一个对象时，需要比较文件的安全描述符和用户令牌中包含的SID信息。安全描述符由访问控制列表（ACL）中的访问控制实体（ACE）组成。如果对象有访问控制列表，安全引用监视器（SRM）检查每一个ACE以确定是否允许访问。一旦SRM授予主体访问对象的权力，后续的检查就不需要了，因为这时主体已经拥有了对象的句柄。

访问权限包括：不能访问，读，修改，完全控制和特殊访问。对目录的访问又增加了几种权限：列表，增加，读。

更进一步，Windows使用句柄的概念定义对系统中受保护对象的访问。每个对象都有一个安全描述符，描述符中包含了对象的自由访问控制列表（DACL）。同样，每一个进程都有一个安全令牌用于标示进程。它被内核用来确定进程的访问权限。ACL中的ACE描述了每个特定的进程SID有何种权限。内核在收到访问请求时检查对象的ACL。

当进程需要对象句柄时请求访问对象。例如，调用CreateFile()。CreateFile函数用于创建一个新文件或打开一个已存在文件。创建文件时安全描述符作为参数传递给函数。如果打开已存在文件，这个安全描述符就附加在文件句柄上，它描述了文件的访问权限，如GENERIC_READ。如果进程有相应的权限，请求成功并且返回访问句柄。所以，同一对象的不同句柄可能有不同的访问权限[Har01]。一旦取得句柄，后续的访问，如读文件就不需要权限检查了。同样，句柄也可以传递给另一个可信的函数用于后续处理。

Java 1.2 Security

Java的安全子系统用到了这里描述的安全模式。Java的安全控制器根据权限和策略创建访问权限。它用checkPermisson方法来验证每个方法的字节码，并使用策略对象确定相应的权限对象。值得注意的是checkPermisson方法会遍历整个调用栈并确定栈中每个方法的权限。java.policy文件被安全管理器使用，它里面包含了每个字节码的授权信息。

致谢

我们感谢Wolfgang Keller，他的非常有价值的建议使这篇文章更加完美。

参考资料

[And01] R. Anderson, *Security Engineering*, Wiley 2001

[Bro99] F.L. Brown and E.B. Fernandez, "The Authenticator pattern", *Procs. of Pattern Languages of Programs Conf. (PLoP99)*, http://jerry.cs.uiuc.edu/~plop/plop99

[Fer99] E.B.Fernandez, "Coordination of security levels for Internet architectures", *Procs. 10th Intl. Workshop on Database and Expert Systems Applications, DEXA99*

[Fer01] E B. Fernandez and R.Y. Pan, “A pattern language for security models”, *Procs. of PLoP 2001*, http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/acceptedpapers.html

[Fer02] E.B.Fernandez, "Patterns for operating systems access control", *Procs. of PLoP 2002*, http://jerry.cs.uiuc.edu/~plop/plop2002/proceedings.html

[Gar02] S. Garfinkel, *Web Security, Privacy & Commerce*, 2nd Edition O’Reilly 2002.

[Har01] J. M. Hart, *Win32 System Programming*, Second Edition, Addison Wesley 2001

[Has02] J. Hassell, *RADIUS*, O'Reilly, 2002.

[Hay00] V. Hays, M. Loutrel, and E.B.Fernandez, "The Object Filter and Access Control Framework", *Procs. of PLoP 2000,* http://jerry.cs.uiuc.edu/~plop//plop2k/proceedings/proceedings.html

[HP] Hewlett Packard Corp., Virtual Vault, http://www.hp.com/security/products/virtualvault

[Kel97] M. Kelley, "*Windows NT Network Security, A Manager's Guide,"* Lawrence Livermore National Laboratory, 1997.

[Kin01] C. King, et.al., *Security Architecture*, Osborne McGraw Hill 2001.

[Lan99] C. R. Landau, "Security in a Secure Capability-Based System," *Operating Systems Review,* October 1999.

[Mic00] Microsoft, *Windows 2000 Security, Technical Reference*, 2000

[sam] SAML, http://www.saml.org and http://www.oasis-open.org/committees/security/

[Sch00] D. Schmidt, et.al. "Pattern-Oriented Software Architecture," Wiley 2000.

[Sil03] A. Silberschatz, P. Galvin, G. Gagne, *Operating System Concepts (6th Ed.),* John Wiley & Sons, 2003.

[Vis99] P. Viscarola, "*Windows NT Device Driver Development*," Macmillan Technical Publishing, 1999.

[Zac99] W. H. Zack, *Windows 2000 and Mainframe Integration*, Macmillan Technical Publishing 1999.

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

远程认证者和授权者模式

Eduardo B. Fernandez、Reghu Warrier 著，杨德仁 译

查看评论

摘要

许多分布式系统包括具有各种计算设备的节点，这些设备要访问共享资源。这种环境需要一个安全和容易管理的认证和授权机制。我们在此描述了一个用远程认证协议实现这个目标的模式。这是一个复合模式，包含2个已知的模式：代理和基于角色的访问控制。

意图

在访问松散耦合的分布式系统中的共享资源时，提供认证和授权便利。

例子

一个跨国公司在不同国家如美国和巴西有许多雇员。支持美国职员的认证和授权信息要保存在美国的服务器中，支持巴西职员的信息要保存在巴西的服务器中。现在假定一个美国职员旅行到巴西，需要访问巴西数据库服务器中的数据。有两种方法可以实现：

1. 在巴西服务器中复制职员用户信息，得到访问数据的授权。

2. 借用有相似权限的巴西雇员的用户名称，用于访问需要的信息。

两种方法都有缺点。系统管理员需要以协调方式在每个被访问的系统中创建和管理用户帐户，以维护安全策略实施的一致性。如果雇员的用户名被借用，就会危及用户责任。

上下文

松散耦合的分布式系统诸如互联网，由许多计算节点组成，其中有些节点需要共享资源。例如，在几个国家具有分部的公司。

问题

在不需要冗余的用户登陆信息下，如何在分布式环境中提供认证和授权？

在过去几年，电信、互联网和电子商务从做生意的一种选择日益发展成主流消费者的活动。对公司数据安全的关心程度急剧增长，对用户注册多个领域和多个服务正在变成需要而非奢侈。具有中央服务的系统能提供容易的管理、更多的责任和安全认证。

约束

在多处存储用户认证和授权信息，产生冗余，难于管理，容易造成不一致。

虽然认证信息能存储在任何地方，但这个位置应该对用户透明。

用户一般工作在角色上下文中，这些角色是标准的，无论跨多少领域，至少在公司或机构内部。

借用本地用户的登陆权限，难于分清用户责任。要在用户访问资源时保存用户标识。

方案

设置一个单点入口，能把用户重新导向正确的服务器，以便用户的登陆和访问信息被确认。

我们用特殊的认证/授权服务器实现这种重新导向。这种服务器用户嵌入式网络设备诸如路由器、modem服务器、交换机等。认证服务器负责接收用户连接请求、授权用户并返回客户端需要给用户传递服务的所有配置信息。图1显示了这种方法。客户端通过代表包含用户登陆信息的服务器的代理服务器请求服务。这个请求被路由到远程服务器，后者根据基于请求主体的角色以及对保护对象的权限来验证。

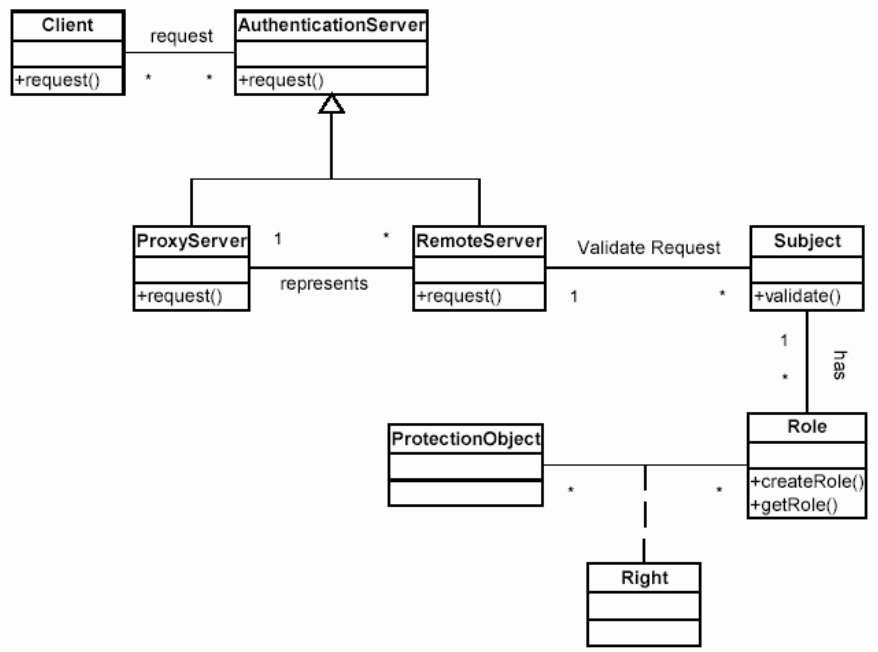


图1 远程认证/授权模式的类图

动态结构

典型系统用下述信息：

访问请求：由客户端发送请求认证和授权网络访问连接尝试。

访问接收：由服务器发送对访问请求信息的响应，通知客户端连接已经被认证和授权。

访问拒绝：由服务器发送对访问请求信息的响应，通知客户端连接请求被拒绝。若信用没有得到认证或连接没有得到授权，服务器便发送这个信息。

访问挑战：由服务器发送对访问请求信息的响应，告诉客户端这个信息需要回答。

帐户请求：由客户端发送

帐户请求：由客户端发送去指定被接收的连接的帐户信息。

帐户响应：由服务器发送响应帐户请求消息。这个消息承认成功的收到和处理帐户请求消息。

消息由头和属性组成。每个属性说明了连接尝试的一块信息。

下述场景（图2）阐明了一个在客户端和转发/远程服务器之间的基于代理的通信：

1. 客户端向转发服务器发送访问请求。
2. 转发服务器把访问请求转发给远程服务器。
3. 远程服务器给客户端发送访问挑战。
4. 转发服务器把访问挑战发送到客户端。
5. 客户端为挑战计算响应并经过第二次访问请求把它转发给转发服务器。
6. 转发服务器把这个访问请求转发给远程服务器。
7. 如果响应与希望的响应匹配，则远程服务器用访问接收回答，否则使用访问拒绝。
8. 转发服务器给客户端发送访问接收信息。

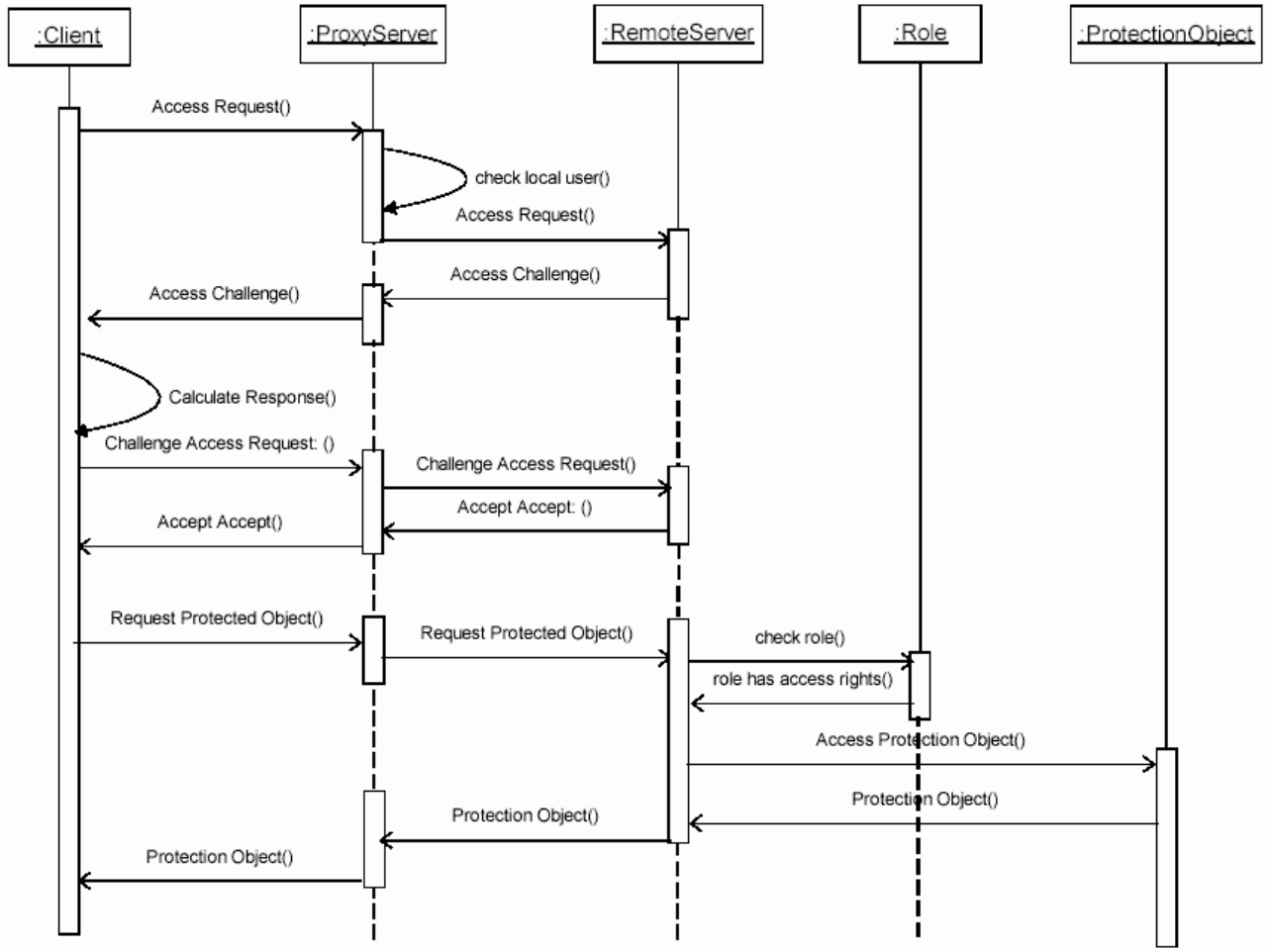


图2 客户端认证的顺序图

实现

一个认证服务器能同时作为转发服务器和远程服务器，有些域作为转发服务器，为其它域作为远程服务器。转发服务器能为许多远程服务器作为转发者。远程服务器能有多个向它转发的服务器，并能为任意数目的区域提供认证。转发服务器能向其它转发服务器转发从而形成一个代理链。需要查询服务去查找远程服务器。

结论

这个模式有下述优点：

漫游准许多个管理实体允许各自用户向对方实体网络拨号请求服务。

在单点存储用户登陆和访问权限使之更安全和容易管理。

用户的登陆ID、口令等存储在内部拨号数据库或能被从一个SQL数据库访问。

用户信息存储的位置对用户透明。

角色和访问权限是标准的，可以跨多个位置。

服务器和客户端应该支持基础协议。

单元诸如活动卡[ACS]允许复杂的请求/挑战交互。

也有一些缺点：

用到的额外信息增加了负载，这样就降低了简单请求的性能。

这种系统比直接认证客户端的系统更复杂。

实例解释

当美国职员旅行到巴西时，他登陆到远程认证者/授权者，后者把请求重新路由到存储其登陆信息的美国服务器上。

已知应用

远程认证拨号用户服务(RADIUS)是一个广泛部署的IETF协议，以完成网络访问的中央认证、授权和记帐[Has02, Rig00]。起初为拨号远程访问开发的RADIUS现在已经被虚拟专用网(VPN)服务器、无线访问点、认证以太交换机、DSL访问和其它网络访问类型[Hil]支持。图3显示了使用挑战响应方法的RADIUS服务器中的客户端的典型认证顺序。

使用代理RADIUS，RADIUS服务器接收来自RADIUS客户端（如NAS）的认证(或帐务)请求，把请求转发给远程RADIUS服务器，接收来自远程服务器的应答，并给客户端发送应答。代理RADIUS的一般用途是漫游。漫游允许两个或多个管理实体让对方用户拨入自己实体的网络请求服务。

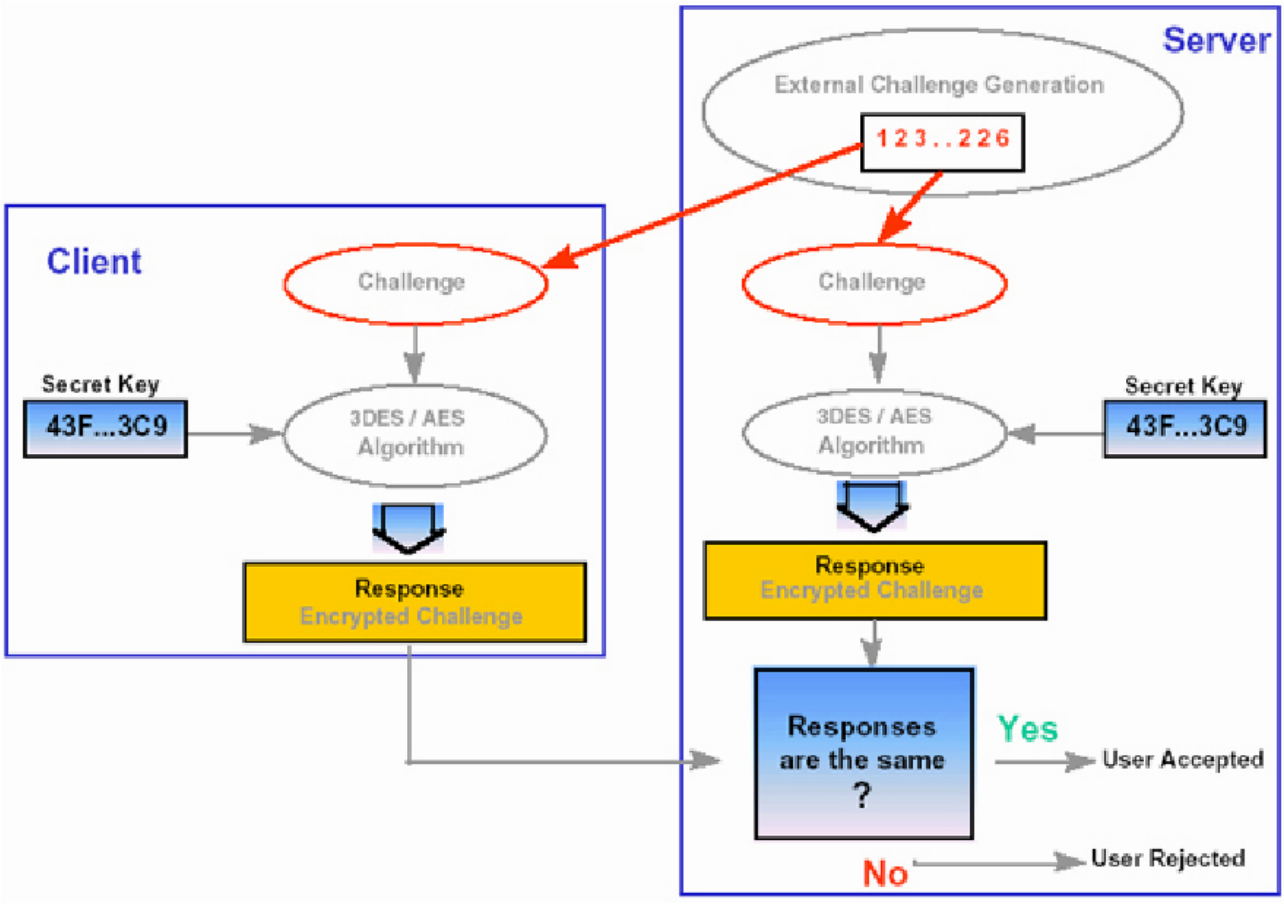


图3 RADIUS挑战/响应模式

今天市面上使用许多商业RADIUS服务器。它们包括：

FreeRADIUS

FreeRADIUS服务器[Fre]用于Unix操作系统的后台程序，允许设置通常用于拨号用户的认证和帐务的radius协议服务器。FreeRADIUS 是开源产品，具有开源的所有优点。

Steel-Belted Radius

Steel-Belted Radius是RADIUS的完整实现。它提供丰富的用户认证、授权和帐务功能，完全支持代理RADUS。它能：

向其它RADIUS服务器转发代理RADIUS请求。

作为处理来自其它RADIUS服务器请求的目标服务器。

把帐务信息传递给目标服务器：执行认证的服务器或另外的服务器。

NavisRadius

NavisRadius是提供认证、授权和记帐证（AAA）服务的RFC标准RADIUS协议的补充。NavisRadius为服务供应商和载体提供一个集成网络范围远程访问安全的方案。NavisRadius支持RADIUS由IETF RADIUS RFC 2865 (RADIUS认证)和2866 (RADIUS 记帐)定义的标准，用于提供广大范围的网络服务。

早期的认证服务器用于CKS, MyNet和Security Dynamics [CTR96]产品中。

相关模式

整个体系结构是单点检查模式[Yod97]的应用。它利用代理模式[Gam95]作为基本组件。最后，利用基于角色的访问控制模型[Fer01, Yod97]定义了用户权限。文献[Bro99]中给出了用于认证分布式对象的一个模式。

参考文献

[ACS] ActivCard Synchronous Authentication (http://www.activcard.com/activ/services/library/synchronous_authentication.pdf)

[Bro99] F.L. Brown and E.B. Fernandez, "The Authenticator pattern", Procs. PLoP99.

[CTR96] “Security: Resellers getting the advantage of growth”, Computing Technology Review, Febraury 1996, 14-17.

[Fer01] E B. Fernandez and R.Y. Pan, “A pattern language for security models”, Procs. of PLoP 2001, http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/acceptedpapers.html, http://www.cse.fau.edu/~ed/SecModels.pdf

[Fre] http://www.freeradius.org/mod_auth_radius/

[Gam95] E. Gamma, R. Helm,R. Johnson, and J. Vlissides, Design patterns –Elements of reusable object-oriented software, Addison-Wesley 1995.

[Has02] J. Hassell, RADIUS, O’Reilly, 2002.

[Hil] J. Hill, “An Analysis of the RADIUS Authentication Protocol”, InfoGard Laboratories, http://www.untruth.org/~josh/security/radius

防火墙的模式语言

Eduardo B. Fernandez 等 著，yoochen 译

查看评论

简介

各种组织和个人对计算机信息安全的要求日益增长。因此，必须保护计算机系统免于攻击[Fer01b]。对于与局域网连接的计算机来说，攻击可能来自外部网络或是其他局域子网。保护局域网的一个普遍方法是建立防火墙，用来作为网关 [Zwi00]。

通过建立控制进入（和离开）局域网信息的扼制点来保障信息安全，防火墙被证明非常有效[Bar99]。因此防火墙限制未被认证的用户访问局域网，并限制局域网访问那些未被信任的外部站点。防火墙可以用作一种执行安全措施和决策的机制，并允许被保护的网络对外部开放有限信息。简言之，防火墙允许认证的通信，并拒绝未认证的或未授权的通信。

我们开发了一种模式语言来描述防火墙的功能。图1标明模式语言中的模式和模式之间的关联和依赖关系。地址过滤防火墙（Address Filter Firewall）定义了基于网络地址的基本过滤功能。基于代理的防火墙（Proxybased firewall）用于应用层，控制对应用的访问。基本防火墙和代理防火墙都可以用状态过滤（stateful filtering）来补充。基于内容的防火墙考虑基于文档内容的过滤。本文中，我们仅讨论地址过滤和代理防火墙模式的细节。

基本上，防火墙是一个单元或是一组单元，执行网络间的访问控制策略。基于防火墙的网络保护和安全策略实施的发展主要集中在建立适用于特定网络、操作系统和计算机的组件[Eps99, Hen01]。然而用于不同系统的防火墙的基本底层架构非常相似。

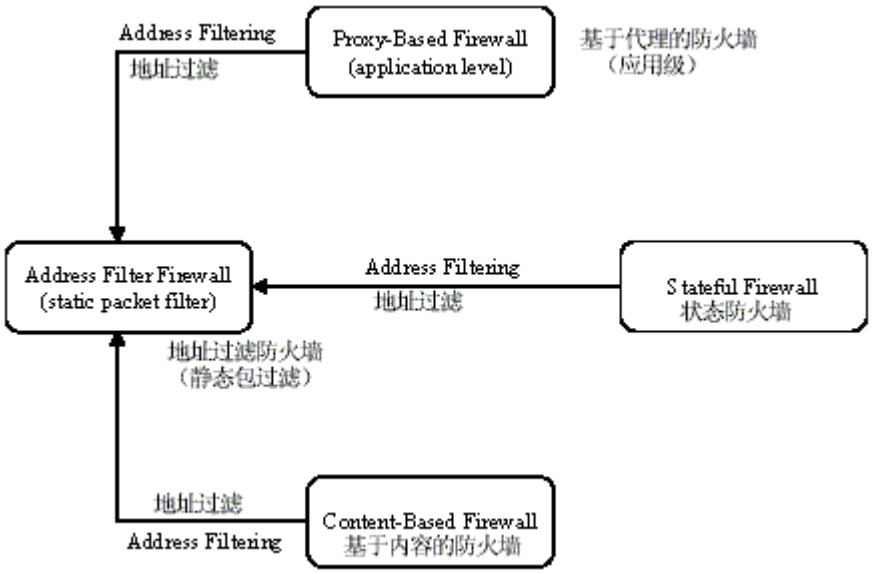


图1 ：防火墙模式语言

地址过滤防火墙（Address Filter Firewall）

目的

根据网络地址过滤流入流出的计算机系统的网络通信.

环境

与互联网和外部网络相连的局域网中的计算机系统

问题

局域网通常被外部（外部网络）攻击。局域网可以被分区，因此攻击可能来自其他局域网。

约束

- 以用户透明的方式过滤计算机系统的输入输出通信
- 网络管理员部署、配置多种防火墙；因此明了要过滤什么和怎样过滤的模型很重要
- 防火墙的配置必须反应机构的安全策略，否则，决定过滤什么很困难。
- 要过滤什么总是在变化：因此以改变防火墙的配置来适应过滤什么的变化要容易实现。

- 规则指定了允许，阻塞，抛弃哪种类型的通信。否则实现特殊的策略很困难。
- 为实现审核和防御，记录客户端请求日志是必要的。

解决方案

仅当存在批准来自客户端地址的通信的规则时，该客户端才可以访问局域网。因此，每个客户端与局域网的连接都被某个规则控制。防火墙由一系列访问规则组成，这些访问规则由机构（属于该局域网）根据它的策略制订。局域网可以有一个或多个防火墙。如果一个特殊请求未满足任何一个显式的规则，那么将应用缺省规则。

动态特性

我们用序列图描述基本防火墙模式的动态特性，序列图对应于两个基本用例。

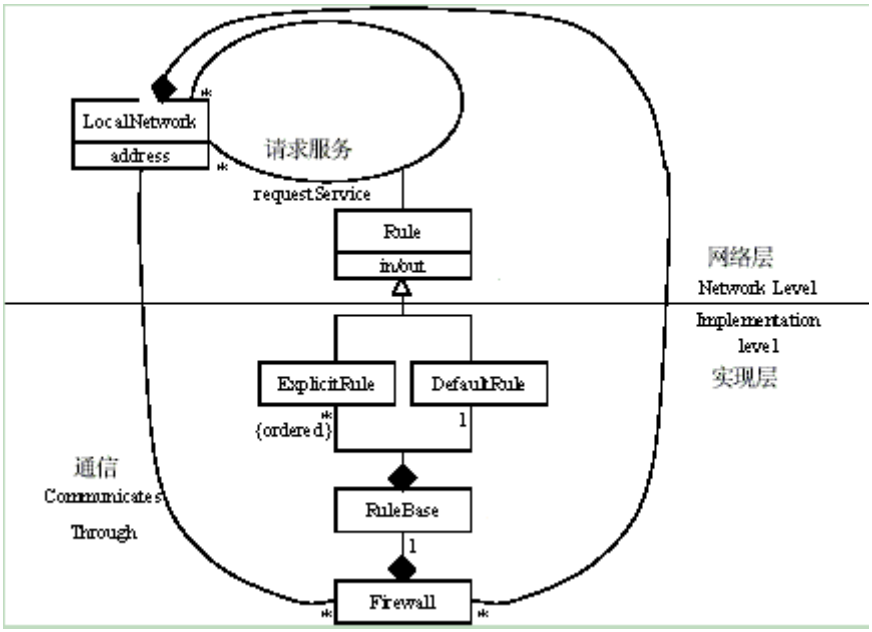


图2： 基本防火墙模式的类图

过滤客户端请求

- 摘要：远端网络要访问局网，不仅要传输而且要获取信息。该访问请求穿过防火墙，防火墙根据规则集合决定接受或是拒绝请求，即防火墙过滤该访问请求。图3说明了这个用例。
- 角色：外部客户。
- 前置条件：防火墙中必须有过滤请求的规则集。

- 描述：

  a) 外部网络请求访问局网。

  b) 防火墙根据规则集过滤请求。如果规则集中没有与之匹配的规则，将用缺省规则来过滤请求。

  c) 如果请求被接受，防火墙允许对局域网的访问。

- 分支流：如果请求被拒绝，防火墙拒绝外部网络对局网的访问请求。

- 后置条件：防火墙接受信任客户对局网的访问。

定义新规则

- 摘要：防火墙的管理员在规则集中增加新规则。防火墙检查要增加的新规则是否已在规则集中，图4阐明了这个用例。

- 角色：管理员

- 前置条件：管理员必须有增加规则的权限

- 描述：

  a) 管理员触发增加新规则

  b) 如果规则集中没有该规则，新规则被加入

  c) 防火墙接受新规则的加入

- 分支流：如果规则集中已有该规则，新规则不加入规则集。

- 后置条件：新规则被加入防火墙中。

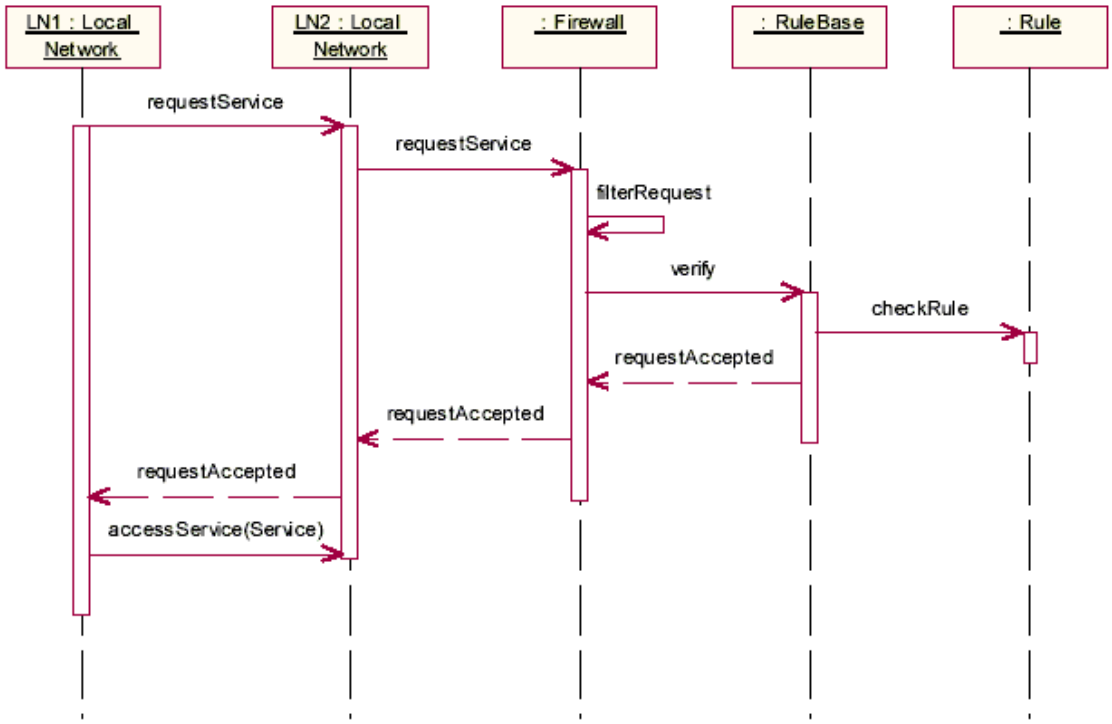


图3 ： 过滤客户请求的序列图

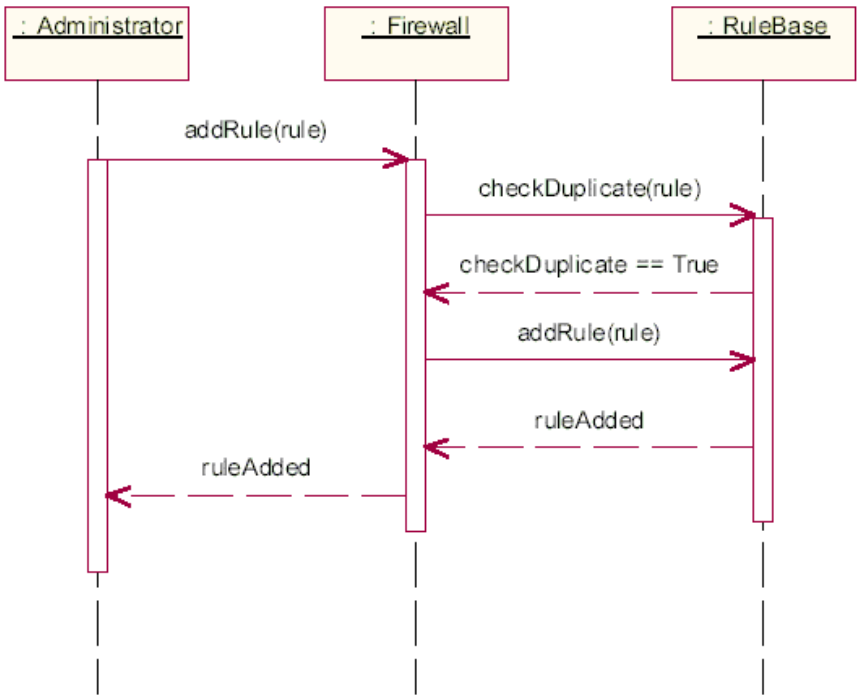


图4：定义新规则的序列图

结论

基本防火墙模式有一下优点：

- 基于网络地址，防火墙过滤所有经过的通信，并对应用透明。
- 可以通过防火墙的规则表明机构的过滤策略
- 防火墙有助于发现可能的攻击，有助于正式用户对其行为负责[Sch03]。
- 防火墙适于记录流入、流出消息的系统日志
- 成本低，许多操作系统将防火墙作为系统的组成部分
- 高性能，防火墙只需查看报文包头。

基本防火墙模式（可能）有以下不足：

- 防火墙的有效性可能受限于规则集（优先级）
- 防火墙的有效性限于局网的入口点，一旦潜在的攻击者通过了防火墙，系统的安全将遭到破坏。
- 基本防火墙仅对通过防火墙的网络通信实施安全策略。
- （基本）防火墙无法阻止更高层次的攻击
- 防火墙通常对被保护系统的可用性，性能和成本造成负面影响[Sch03]。
- 不同站点、网络和系统所执行的防火墙策略是不同的。新增的规则可能妨碍规则集中已有的规则；因此，必须小心谨慎地增加和更新访问规则。
- 非状态察觉
- 包过滤不能识别伪造的地址，因为包过滤仅检查IP包的报文头
- 黑客可以在不用于路由的报文头和有效负荷中加入恶意命令或数据。

已知应用

这个模型对应于基本（包过滤）防火墙架构，该架构用于商业防火墙产品，如：基于Network Associates' Gauntlet Firewall[Eps99]的ARGuE (Advanced Research Guard for Experimentation)；OpenBSD Packet Filtering Firewall [Rus02]以伯克利分布式软件系统（Berkeley Software Distribution system）的基本防火墙架构为模型；还有Linux Firewalls [Zie02]以 linux操作系统的基本防火墙架构为模型，基本防火墙模型被用作其他包括了更多高级特性的防火墙的底层架构，举例来说：CyberGuard [Hen01]，主要采用了全状态检测（stateful inspection）防火墙架构。

相关模式

认证模式[Fer01a]定义了基本防火墙模式的安全模型。基于角色访问控制模式，是认证模式的特殊化模式，适用于根据角色和权限定义网络和访问规则的情况[Fer01a]。防火墙模式是个特殊的Single-Point-of-Access [Yod97]的例子。另一个防火墙模式在[Sch03]中描述。

应用代理防火墙

目的

根据访问应用的类型，检查（过滤）网络的流入流出通信

环境

与互联网和外部网络相连的局域网中的计算机系统，需要比包过滤更高层次的安全

问题

地址过滤防火墙仅仅通过检查网络地址决定对消息的访问。然而，潜在的攻击可能包含在包的数据段中，这些包的网络地址被证实是可访问局网的[Sch03]。另外，地址过滤防火墙无法阻止IP欺骗(IP spoofing)。需要将局域网和外部客户网络虚拟地分开，才能彻底检查网络通信。

约束

- 地址过滤的约束仍是有效的
- 网络管理员部署、配置不同的防火墙：因此明了要过滤什么、怎样过滤和应用代理如何实现的模型很重要

- 防火墙的配置必须反应机构的安全策略，否则，决定在应用数据中检查和修改什么很困难
- 要过滤什么总是在变化：因此以改变防火墙的配置来适应过滤什么的变化要容易实现。如果要请求对防火墙不支持的代理的服务，那么将阻塞这个请求。
- 为实现审核和防御，记录客户端请求日志是必要的。

解决方案

客户只与请求服务的代理交互，代理再与受保护的服务通信。受保护的服务只与应用代理防火墙通信。仅有请求服务的应用代理时，客户才可以从服务器接收服务。每个应用代理都有自己预定义（由管理员声明的）访问和修正规则，这些规则被用于检查、改变并过滤流入（或流出）的消息。

图5说明了这个模式的类图，是图2的扩展。图5包括在每个局域网中现有的分散服务，和过滤请求服务的应用代理。

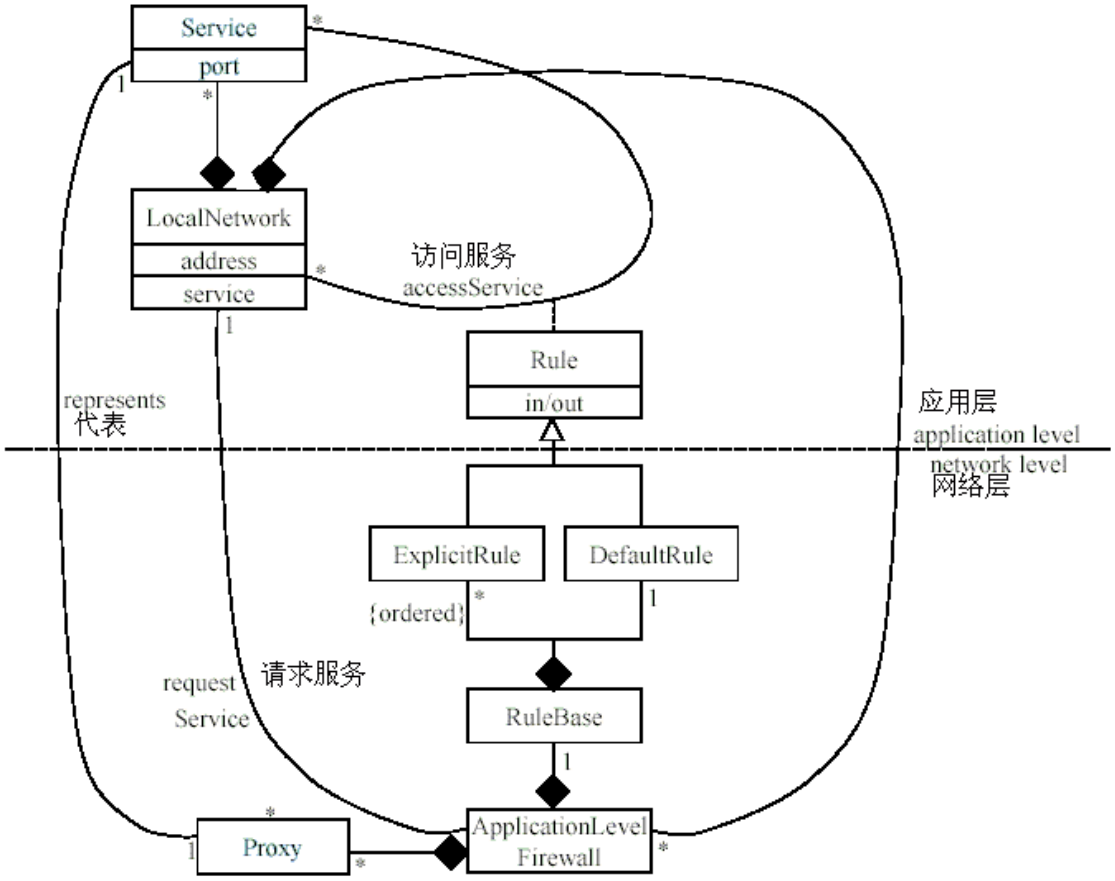


图5：应用代理防火墙模式的类图

动态特性

这里展示了一个过滤服务请求的用例。

为客户提供服务的用例

- 摘要：外部客户想访问本地服务器上的服务。访问穿过防火墙，防火墙根据应用代理和规则集，决定是否拒绝或接受请求（数据内容修改或不修改）。图6说明了这个用例。
- 角色：外部客户。
- 前置条件：无。
- 描述：

a) 外网向应用代理防火墙发出访问服务的请求。

b) 防火墙根据应用代理和访问/修正规则过滤请求。如果没有满足规则集中任何规则，那么将应用缺省规则过滤请求。

c) 如果未经修改或修改后的请求被接受了，那么将允许客户通过应用代理访问服务。

- 分支流：如果应用代理防火墙不支持服务请求，或者防火墙认为客户不可信，那么防火墙不批准该服务请求。
- 后置条件：防火墙接受来自可信客户对局域网的服务请求。

结论

应用代理防火墙模式有以下优点：

- 根据对用户透明的预定义应用代理，防火墙检查、修改（如果需要）并过滤所有请求。
- 可以通过应用代理和规则表明机构的过滤策略。
- 在包中的数据段中包含了可疑命令时，可以修改信息的特定部分
- 防火墙有助于发现可能的攻击，有助于正式用户对其行为负责[Sch03]
- 防火墙可以避免内部系统[Sch03]的协议栈中可能的执行错误。内部网络的IP（互联网协议）地址总是对外部网络隐藏。

- 防火墙适于记录流入、流出消息的系统日志
- 因为防火墙检查包括头和数据段的整个包，所以可提供高级别的安全

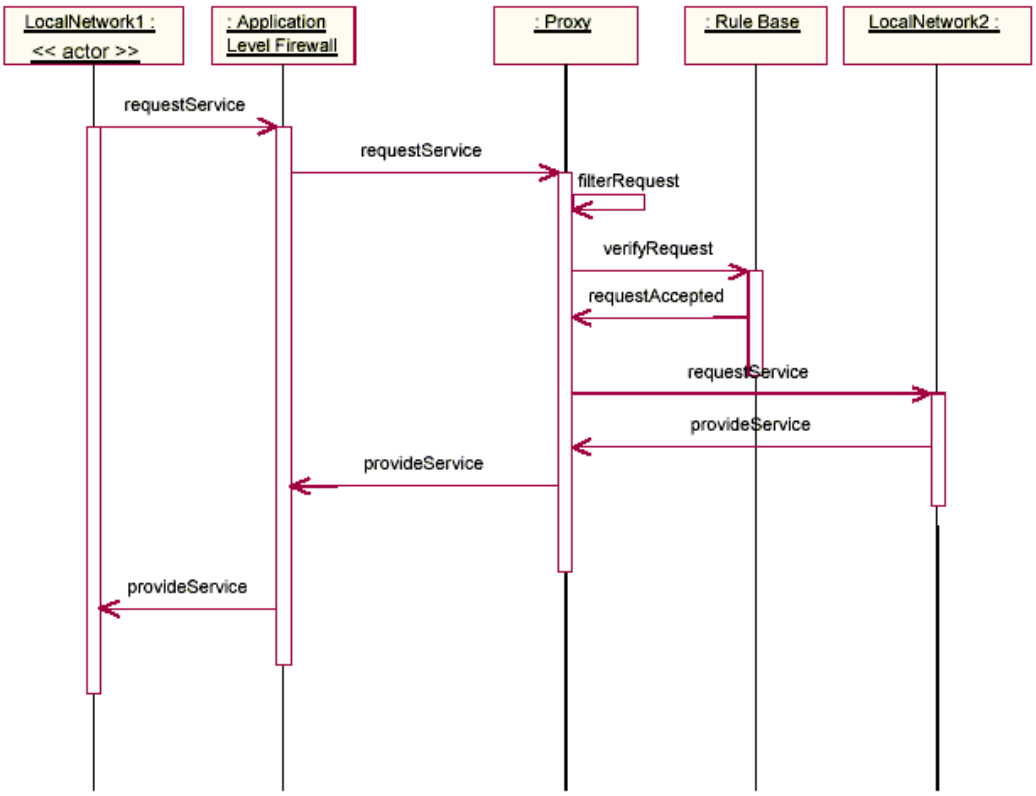


图6 过滤服务请求的序列图

应用防火墙模式有以下不足：

- 对特殊的代理有额外的执行成本，另一方面，普通服务的代理已存在
- 应用代理的开销和对包数据段的检查导致速度变慢
- 防火墙复杂性增加，在应用和/或用户和系统交互中，应用代理防火墙可能需要改变。
- 不同站点、网络和系统执行的防火墙安全策略不同。对特定应用代理增加新规则，可能干扰规则集中的已有规则，因此，必须小心谨慎地增加和修改访问规则。
- 非状态察觉

已知应用

应用代理防火墙使用在大多数防火墙产品中使用的基本地址过滤防火墙模型，如ARGuE Guard [Eps99].一些使用应用代理的特殊防火墙产品有Pipex Security Firewalls [Pip03] 和 InterGate Firewall.

相关模式

地址过滤防火墙是应用代理防火墙模型的基础。

参考文献

[Bar99] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. FIRMATO: A Novel Firewall Management Toolkit. In *Proceedings of the 20th IEEE Symposium on Security and Privacy*, pp 17-31, Oakland, California, April 1999.

[Che03] W. Cheswick, S.M.Bellovin, A.D.Rubin, *Firewalls and Internet security,* 2nd Ed., Addison-Wesley, 2003.

[Eps99] J. Epstein. Architecture and Concepts of the ARGuE Guard. In *Proceedings of the 15th Annual Computer Security Applications Conference*, pages 45-54, Phoenix, Arizona, December 1999.

[Fer01a] E. B. Fernandez and R. Pan. A Pattern Language for Security Models. In *Proceedings of the PLoP Conference*, 2001. http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/acceptedpapers.html.

[Fer01b] E. B. Fernandez. An Overview of Internet Security. In *Proceedings of the World's Internet and Electronic Cities Conference (WIECC 2001)*, Kish Island, Iran, May 2001.

[Gat00] B. Gatliff. Web by Proxy. *Embedded Systems Programming Magazine*. May 2000. http://www.embedded.com/internet/0005/0005ia1.htm

[Hay00] V. Hays, M. Loutrel, and E.B.Fernandez, "The Object Filter and Access Control framework", *Procs. Pattern Languages of Programs (PLoP2000) Conference*, http://jerry.cs.uiuc.edu/~plop/plop2k

[Hen01] P. Henry. An Examination of Firewall Architectures. CyberGuard Corporation White Paper, April 2001. http://www.cyberguard.com.

[Nou00] N. A. Noureldien and I. M. Osman. A Stateful Inspection Module Architecture. In *Proceedings of TENCON 2000*, vol. 2, pp 259-265, Kuala Lumpur, Malaysia, September 2000.

[Pip03] Pipex Firewall Solutions. http://www.security.pipex.net/about.shtml.

[Rus02] R. E. Rustad, Jr. Guide to OpenBSD Packet Filtering Firewalls. Kuro5hin Article, Nov. 2002, http://www.kuro5hin.org/story/2002/11/23/14927/477.

[Sch03] M. Schumacher. Firewall Patterns. In *Proceedings of the Euro PLoP Conference*, 2003.

[Vddd] InterGate Firewalls from Vicomsoft. http://www.vicomsoft.com/

[Wal01] B. Walder. Firewalls. *Computer Weekly Online*, 2001. http://www.nss.co.uk/Articles/firewalls01.htm

[Yod97] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security". *Procs. PLOP'97*, http://jerry.cs.uiuc.edu/~plop/plop97 Also Chapter 15 in *Pattern Languages of Program Design*, vol. 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.), Addison-Wesley, 2000.

[Zie02] R. Ziegler and C. Constantine. Linux Firewalls: Packet Filtering. *News Riders*, March 2002. http://informit.com.

[Zwi00] E. D. Zwicky, S. Cooper, and B. Chapman. Building Internet Firewalls.O'Reilly and Associates, 2nd edition, 2000.

一种请求及分配有限资源的分析模式

Fei Dai、Eduardo B. Fernandez 著，wnb 译

查看评论

1 目的

如何公平有效地为多个请求分配有限数量的非可重用资源？

2 内容

在许多情况中，可用资源的数量远远小于请求的数量。例如，流行体育竞赛的门票在比赛开始前就已经全部告罄；公司中重要的职务往往有多个角逐者。在中国，原始股票由于其价格低廉往往首次提供就全部卖出。从系统的观点来看，这些资源都是不可重用资源。不存在资源的返回和重新分配的问题。与之相对应的是可重用资源，如交通工具、宾馆房间、飞机座位，这些可重用资源在使用后返回并可在未来重新分配使用。

由于其操作可选则的多样性和复合性，对有限不可重用资源的分配非常复杂。通常，需要预先请求（发送请求并等待分配），但在某些特殊情况下，客户可能没有预定或等待而直接需要资源，比如当资源接近有效期限时。例如当重要足球比赛球票的请求处理完成后，人们仍然可以在比赛开始前通过到售票处直接购买球票（直接请求）。通常分配在某一特定日期或事件到来后将被放弃，例如等待机票的人们在飞机即将出发时，当确定不会再有预定机票的乘客到来时，可以获得机票。其它情况还包括，请求处理完成后分配也就完成了（如足球票的例子）。

当然也还存在其它的分配资源的方式：可以采用先来先服务的方式，通过比较优先级，通过抽签、谈判或其它策略。本文假定资源相互之间是不可区分的，它们都属于特定的类型。例如，某一请求声明了一些指定价格的票，或一定数量的某类股票。

3 问题

分析模式应该满足下列基本用例：

请求资源：客户请求获得某一类型的固定数量的资源。这些资源相互之间是不可区分的。请求放入队列中等待资源分配。客户可能是人、机构或计算机实体（进程）。

分配资源：在经过一定时间后，或者有足够的资源（还没有分配），从请求等待队列中检索出请求并将资源分配给该请求。

获取资源：在请求被接受后，客户需要立即获得分配的资源或者在一定时间后获得该资源。客户对资源的请求也可能得到部分的满足，例如，某一客户需要 10 张球票被分得 5 张。

撤消请求：客户撤消其请求并释放已经分配得到的资源

下列因素将对可能获得的解决方案产生影响

对请求的处理应该是公平的（对所有请求采用同样的处理方式），尽管某些分配策略可能带有不公平的色彩。例如优先级分配。

解决方案仅仅包含基本的语义，代表最小的应用，包含一些用例。这是使得该模式成为语义分析模型的基础[Fre00]。这类模式有助于建立概念模型。

对实体的有效使用。没有冗余和关联的实体。

许多应用需要纸质文档。标准文档应该清楚地体现在模型中。例如，一系列预留球票的发售可能包含一个发售文档标明价格、日期等。这使得该模型更加直观并且方便了这些文档的建立。

4 解决方案

使用抽象队列作为分析模型的核心。根据分配策略将请求放入队列中。例如，基于先来先服务策略将会把一个新请求放入队列尾；基于优先级策略将新的请求按照其优先级进行放置。每个请求必须放入队列中。资源只能从队列前端的请求开始进行分配。直接分配将被视为特殊的情况，在此情况下，请求将被放入队列前端。当一个已经分配的请求撤消时，其所占用的资源将立即成为可用资源。客户可以选择是否接受对其请求的部分分配。

4.1 类图

图 1 显示了一些必要的类。客户对一些类型的资源发出请求。对每种类型的资源有一个队列，分配由 **AcquisitionRecord** 描述。**AllocationStrategy** 用于分配资源。注意请求需要几个类型的资源而实际的分配可能只有其中的部分。属性 *acceptPartialAssignment* 指示客户是否同意接受更小的分配。

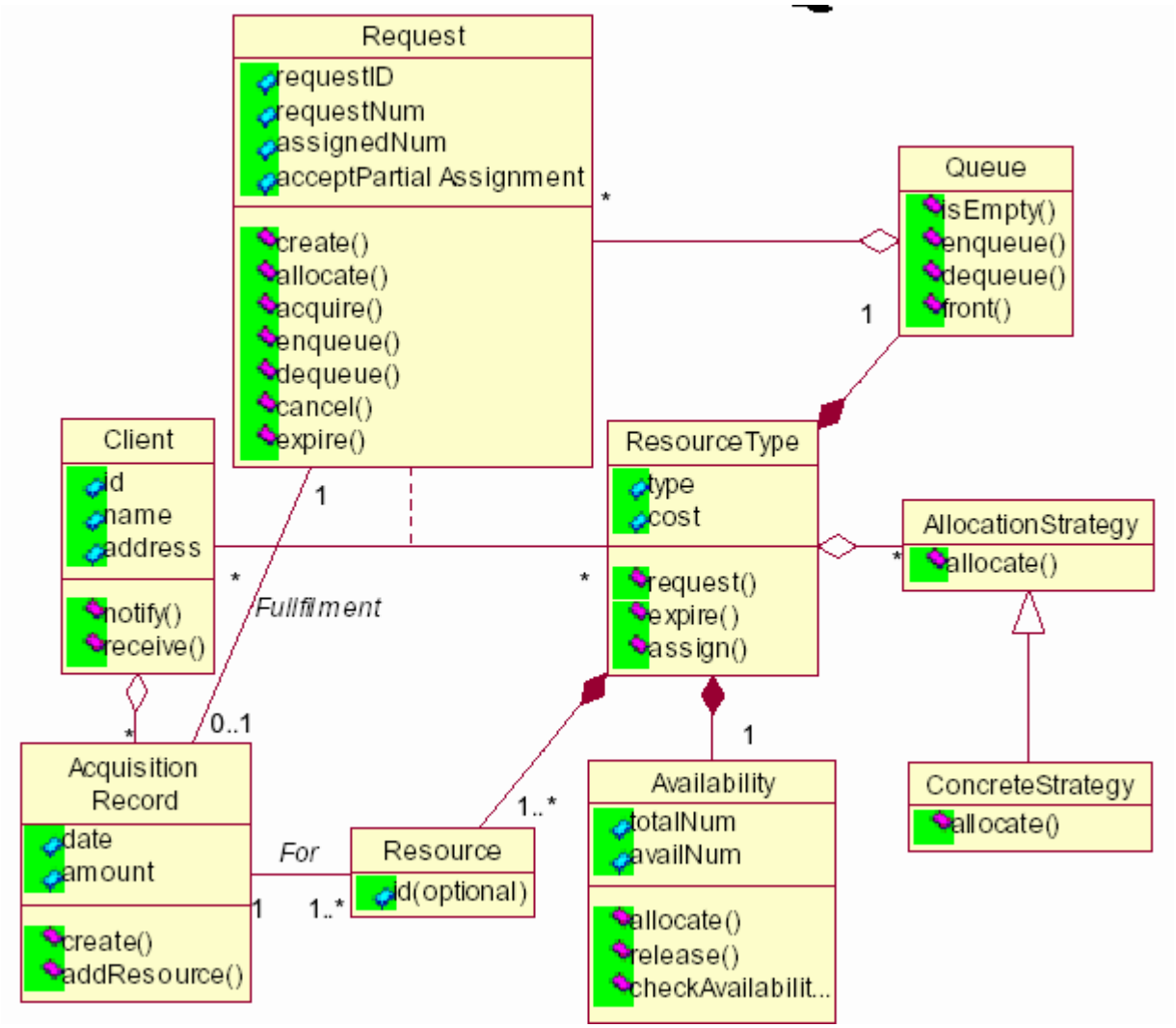


图 1 模式的类图

4.2 动态性

图 2 展示了请求对象的状态图。请求最先建立，加入队列中，然后将资源进行分配。任何时候都可以撤消请求或者在某一时间限定后终止。图 3 显示了资源请求、分配及获取的序列图。首先，请求对象被建立并填加进队列中。前端队列的请求经过检验，将资源分配给它们并作出通知。建立了请求资源的客户以及 AcquisitionRecord。

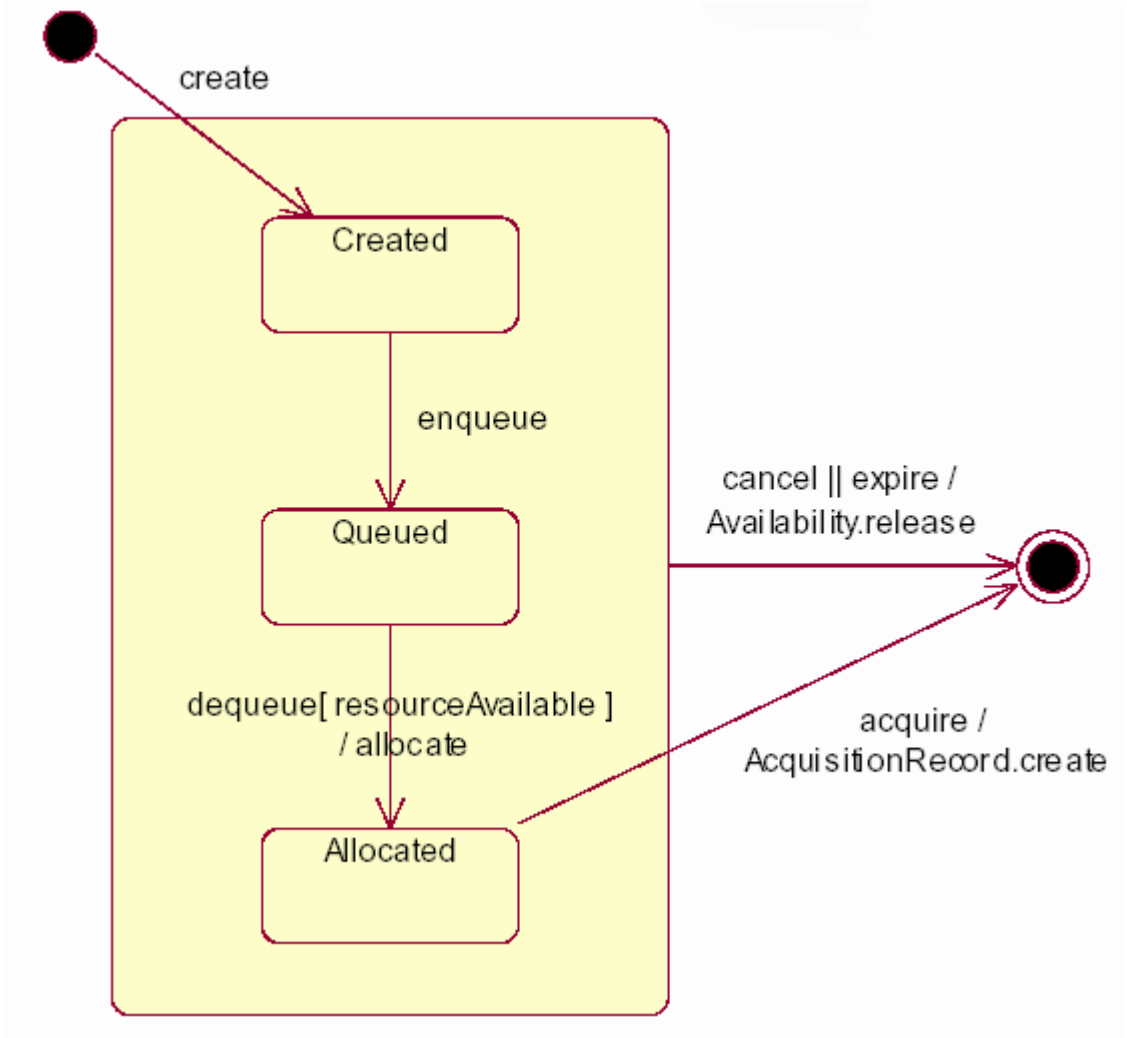

create
Created
enqueue
Queued
dequeue[resourceAvailable]
/ allocate
Allocated
cancel || expire /
Availability.release
acquire /
AcquisitionRecord.create

图 2 请求对象的状态图

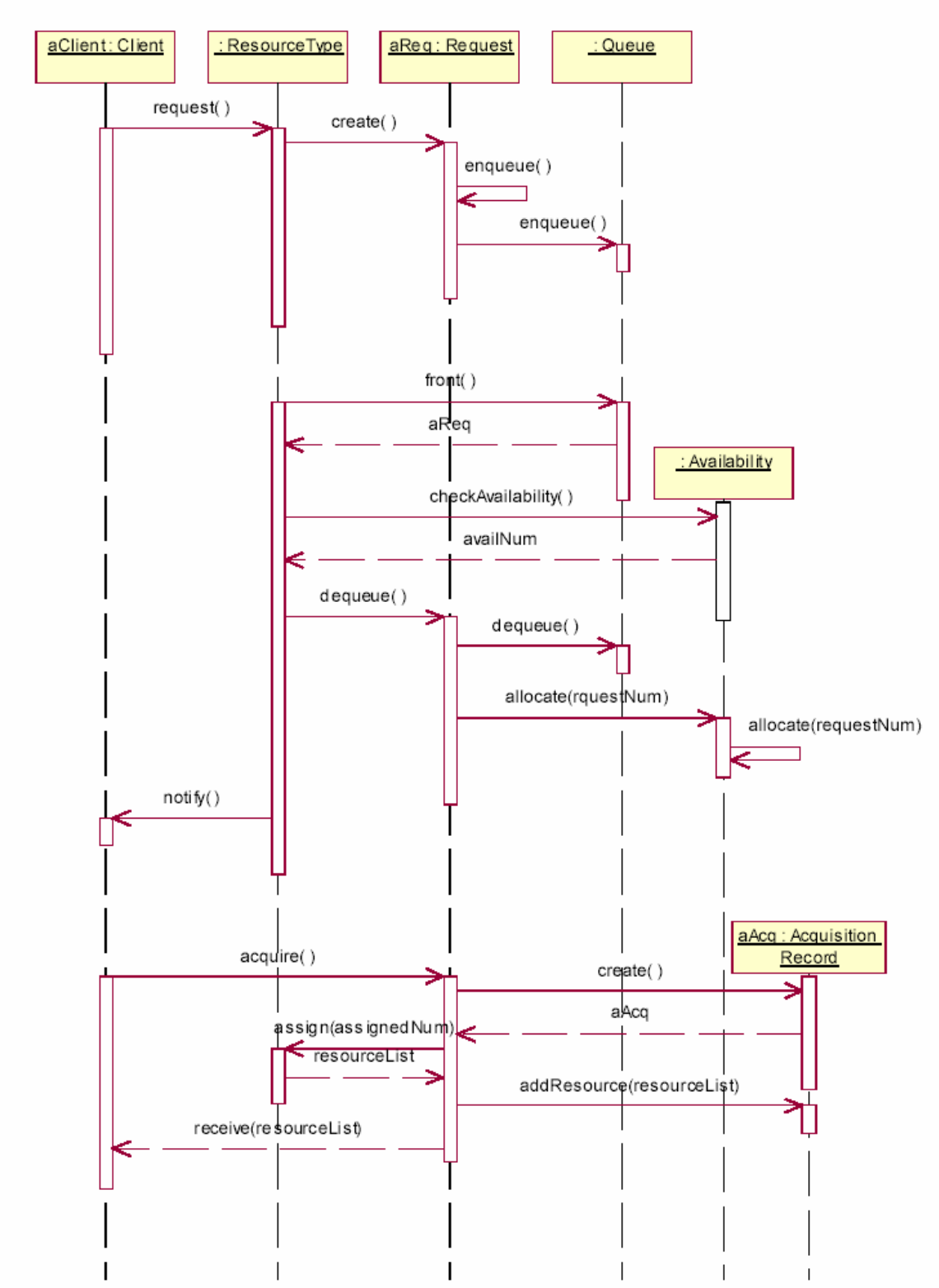


图 3 请求、分配、获取资源的序列图

5 解决的实例：中国股票市场的股票发售问题

在中国，当公司在市场上发行新的股票时，他们将其直接销售给市场的参与者，包括大投资商（基金、投资银行等）和为数众多的个体投资人。由于发售时价格比市场价格低，因此购买数量往往超过有效的股票数量。为了给每个投资者提供良好的机会，管理层采用了彩票系统来分配购买操作。典型的股票发售周期有以下阶段：

公司公告将发售的股票代码、名称与价格。在下面的预申请阶段（通常 1-3 天），每个投资人可以提出包含请求数量在内的请求。请求者在其帐户中需要有购买其申购数量相当的资金，对应数量的资金将被冻结直至第二阶段结束。在冻结期间的利息由发行股票的公司获得。

每个请求被赋予一个有效的抽奖号码，抽奖号码是连续的顺序号。这些号码按照比例分配给各个请求。最后，通过随机抽取末尾号码确定中奖的顺序号。中奖的号码的频度按照整个请求的数量和能够提供的股票数量确定。有效的顺序号代表购买股票的权利，将发送给相应的投资人，这一阶段通常需要 1 周时间完成。

在第三阶段（通常是一天时间），获得购买权的投资人使用其购买权益购买股票。当然，他们也可以放弃购买权利。如果在结束时仍然有股票剩余，则这些股票将卖给预先确定的投资银行。

图 4 显示了该使用模式的类图。说明如下：

去除了资源类，因为同一公司的股票是不可区分的。只要知道每笔交易的股票的数量；

只有一个 LotteryStrategy 对应所有的股票。发布策略是唯一的并且对所有投资人都一样。

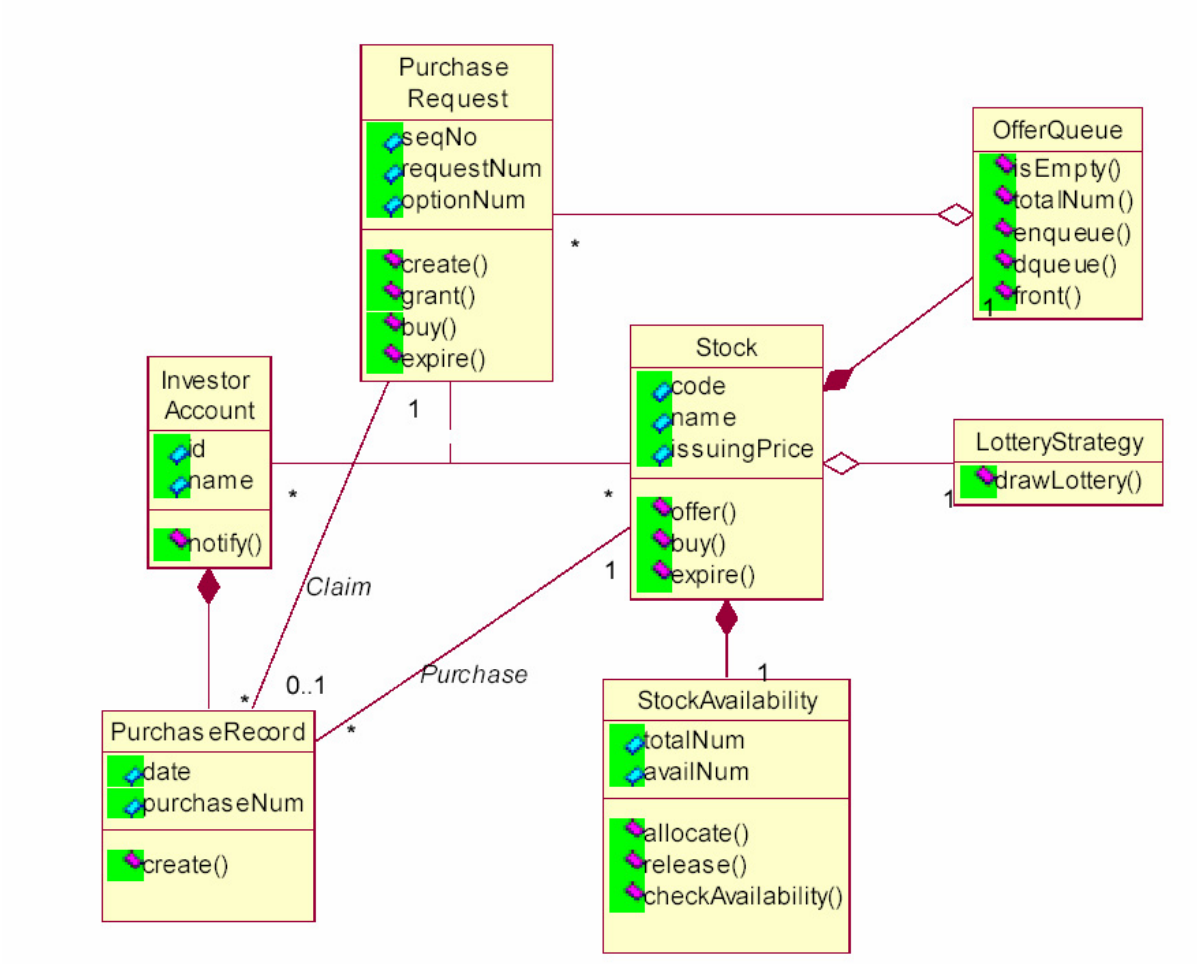


图 4 购买股票应用的类图

6 应用案例

使用该模式的应用包括：

体育比赛球票预定及购买（例如，2006 年世界杯）；

工作申请，多个申请者申请几个工作；

中国股票申购问题，投资人首先通过彩票系统获取购买权，并按照其所获得的购买权申购对应数量的股票；

由政府提供的廉价公寓的销售问题（例如，中国深圳）；

为产品分配组件。

7 结论

模式满足下列要求

队列机制为处理请求提供了良好的处理机制，每个希望获得资源分配的请求加入到队列中。

正如在应用部分所展示的，该模式应用范围宽广：球票、股票、工作申请等。

请求的每个方面都进行了表达，没有冗余的类产生。

该模式为请求和 AcquisitionRecord 文档产生了清晰的类。

该模式可以通过定义变量实现定制。在股票的例子中，我们没有考虑资源类，因为没有区分购买的股票的要求。

为了使该模式更加一般化，本文省略了一些细节

可能存在不同的需要区别对待的客户类型；

在分配时可能存在一些需要进行交互的协商。实现该机制需要在设计阶段建立一些额外的类以定义用于协商的用户接口；

某些系统可以对每类资源使用多个请求。例如，在航空港，对不同类的乘客应该采用不同的队列。

8 相关的模式

该模式对 Reservation and Use pattern 的某些方面进行了扩展。同时，该模式没有考虑资源的使用问题，仅仅考虑到资源获取以前的情况。其它不同的地方包括：资源被视为不可重用的，请求在一定的时间条件后将被取消。Order and Shipment pattern 模式是对该模式的补充。提供了占有资源的细节，即分配的资源的支付与发出。另外该模式还包含以下两个子模式：

Strategy pattern 模式。用在此处为使用不同策略进行资源分配提供可能。

Reservation pattern 模式。为 Reservation and Use pattern 的子模式。对应类：**Client**, **Request**, **ResourceType**。请求与预定方式类似，等待完成，其差别是，实际的预定方式包含确认，在此没有表达。

安全模型的一种模式语言

Eduardo B. Fernandez 等 著，Happy 译

查看评论

摘要

安全问题是 Internet 中比较重要的问题，对于设计一个包含安全模块的新系统，安全性也是必需的。模式是一个很好的工具，可以帮助设计者构建安全的系统。我们在这里将讨论三种模式，它们都是最通用的安全模型：授权（Authorization）、基于角色的访问控制（Role-Based Access Control）和多级安全（Multilevel Security）。这些模式可以应用于系统的所有级别，我们将展示它们在文件授权模式定义中的使用。

1、简介

安全是任何计算系统非常重要的方面，并且在机构把他们的数据库向 Internet 开放后，这已经成为日益严重的问题。大多现在正使用的 Web 系统在设计时并没有过多考虑安全性，即使打上补丁，也还是不能有效抵御攻击。所以，开发系统过程中，在设计的每个阶段都考虑到安全性是很重要的，不能够仅仅满足功能性需求，还要满足安全性需求[Fer00a]。为做到这一步，我们要从表示机构安全策略的高层模型开始。现在，大多数系统使用三种模型：访问矩阵、基于角色的访问控制（RBAC）模型和多级模型。

安全性包含在系统所有的架构层中[Fer99]，因此，分层架构(Layer)模式[Bus96]是本模式语言的入手点。这个模式提供了一个结构，在这个结构中，我们可以为所有层定义模式，共同完成一个安全的系统。它的主要概念是将系统分解成若干抽象的层次级别，高层使用低层的服务。这里给出一种全面看待事物的方式，并且描述了每层所需的机制。我们在以前曾讨论过 [Fer99]：为什么所有这些层必须协调起来，确保其安全性。图 1 展示了我们所考虑的特定分层，这个图展示了每层的一些参与者和它们相应的交叉层。

由于问题的复杂性，我们需要一系列模式语言，每层一种。本文我们从抽象模型的一种模式语言开始，它包括三种基本的模型，上边提到了，这些模型为应用程序架构最上层定义了安全性约束，但在底层实施。这些模型已经被安全性社区广泛研究[Pfl97,Sum97]，我们不想再增加新模型或扩展现有模型。我们的目的是将这些已被接受的模型表示成面向对象的模式，以能够作为构建安全系统的指导，同时，也展示这些模式如何应用于所有系统级别。

首先，我们提出描述资源访问控制的授权模式，接着在第 3 节描述 RBAC 模式，它是授权模式的一种扩展。第 4 节是多级安全模型。在这之后，我们将讨论这些模式如何应用于底层，并展示一种文件访问控制的模式，作为授权模式特殊形式的例子。最后，我们将讨论这些模式的一些共同方面。

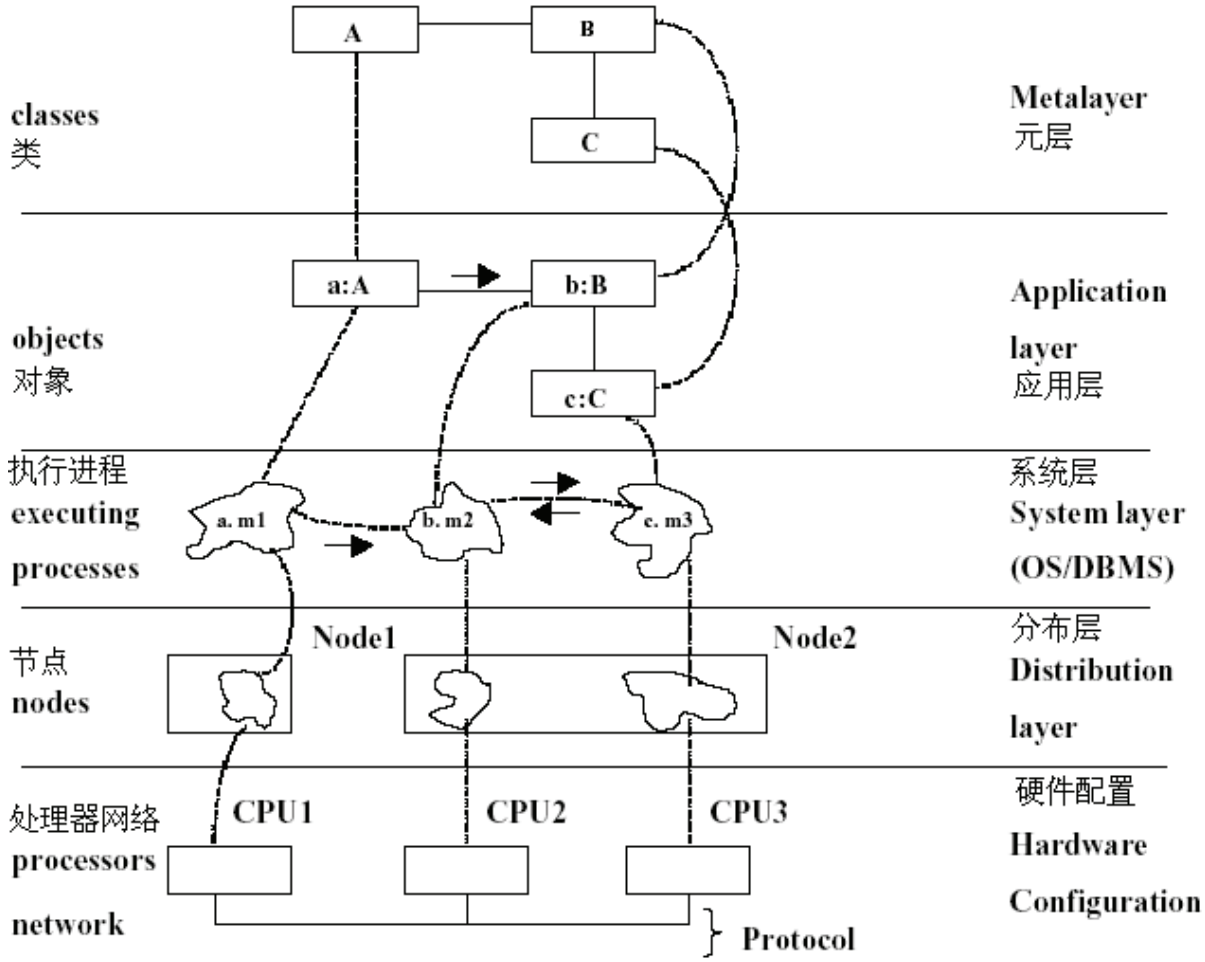


图 1 分层模式的一个实例

2 授权模式

上下文

任何存在主动实体请求某种受控资源的计算环境。

问题

如何描述主动计算实体（主体）对被动资源（保护对象）可允许的访问类型（授权）。

先决条件

- 授权结构必须独立于资源类型，例如对用户访问概念实体、程序访问操作系统资源，要以统一的方式来描述。
- 根据特定条件，谓词或保护措施可能要限定授权的使用。
- 有些授权可以被它的持有者委托给其他主体。

解决方案

用类和类的关联来表示授权规则的元素。类 **Subject** 描述一个主动实体，类 **ProtectionObject** 描述一个被请求的资源，一个授权规则由这两个类的关联来定义。一个关联类 **Right**，包含访问许可的类型（读，写等），包含一个谓词，对授权持有者来说必须为 true，还包含一个复制标志可以为 true 或 false，表明这个权限是否可以转让。另外还有一个操作 check_rights 用来让主体或对象检查请求的合法性。图 2 展示了涉及到的元素。

图 3 是一个时序图，展示了一个请求的验证。ActiveSubject 是一个角色（Actor），例如一个请求某种资源的执行进程，这个请求被一个 Reference 监视器解释。

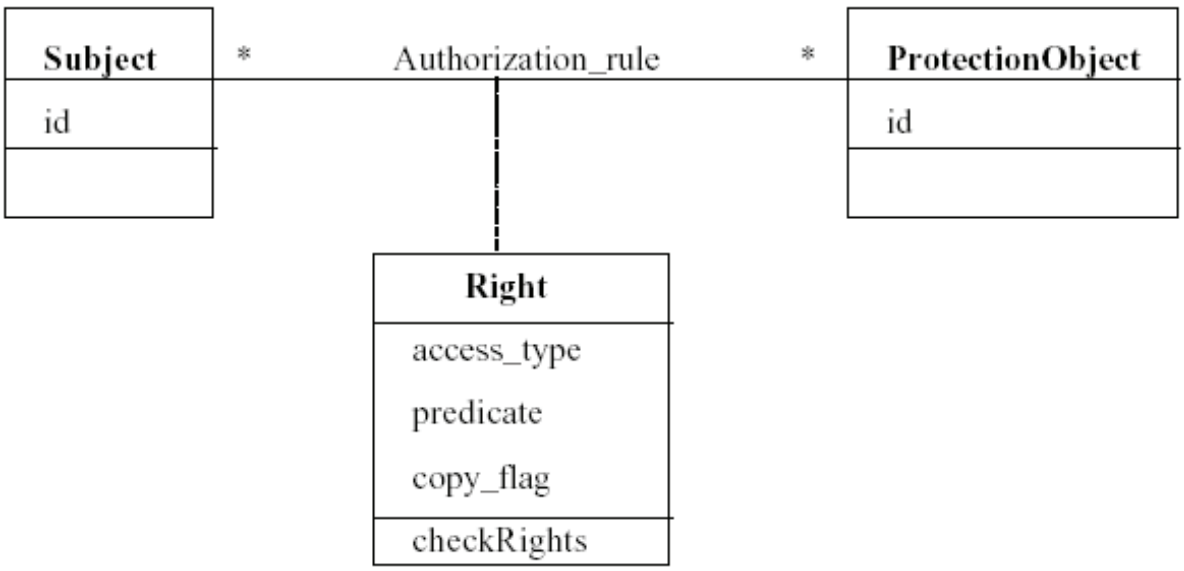


图 2 授权模式

结论

这个模式的优点包括：

- 可以应用于所有类型的资源。主体可以是执行进程、用户、角色、用户组等。保护对象可以是事务、内存区域、IO 设备、文件或其他 OS 资源。访问类型可以是读、写、执行或对象其他高层的某种方法。
- 对于任何可以限定规则应用的条件，谓词是一种通用表示。
- 规则中的复制标志控制权限的转让。不过有些系统不允许转让权限，那么这儿的复制标志将总是为 false。
- 有些系统把管理员授权和一般用户授权分开，以满足更高的安全性（职责分离原则）[Woo79]。
- 请求无需在规则中指定确切的对象，它可以由一个已有的保护对象隐式指定[Fer75]。同样，主体和访问类型也可以隐式，这样当进行一些附加处理时，能提高时间花费的灵活性（引出所需的特定规则）。

已知应用

此模型对应访问矩阵的成分，访问矩阵是一个基本的安全模型[Sum97]。它的首个面向对象形式出现在[Fer93]。后来，它在其他若干论文和产品中也出现过[Ess97,Kod01]。它是多数商业产品访问控制系统的基础，例如 Unix、Windows、Oracle 和很多其他产品。

相关模式

在后面提出的 RBAC 模式就是这个模式的一个特例。在第 5 节，我们将展示另一个特例——操作系统文件访问控制模式。

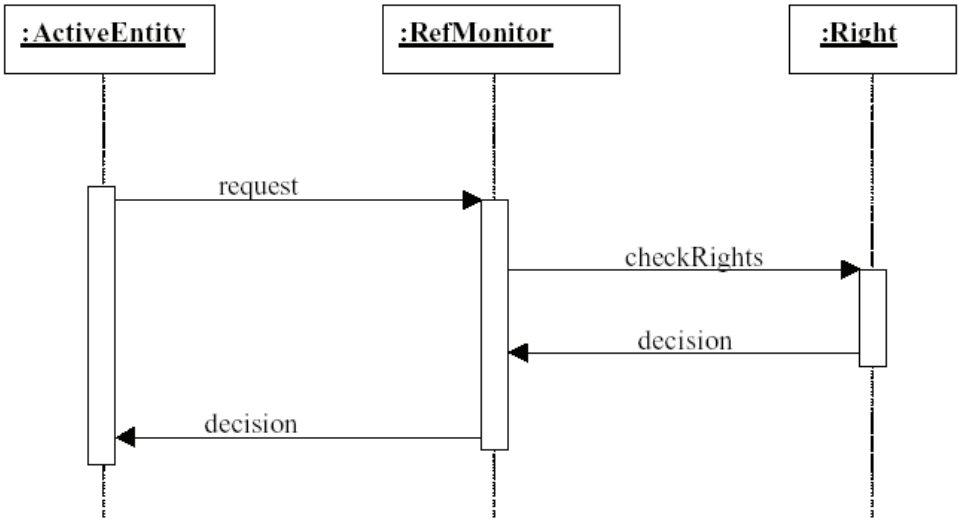


图 3 验证请求

3 基于角色的访问控制（RBAC）模式

上下文

多数机构具有很多不同的工作岗位，需要不同的技能和责任，为了安全考虑，用户应该根据它们的工作岗位获得权限。这对应一种基础安全策略，need-to-know 原则的应用[Sum97]。工作岗位可以视为人们在执行职责时扮演的角色，尤其在基于 web 的系统中，有很多不同的用户：公司职员、客户、合作伙伴、搜索引擎等等。

问题

在一个机构中，如何根据角色为用户分配权限。

先决条件

- 机构中的人根据他们职能不同，对信息的访问有不同的需求。
- 我们想协助机构基于 need-to-know 策略，为它的成员定义精确的访问权限。
- 为单个用户赋予权限需要存储很多授权规则，并且管理员很难跟踪这些规则。
- 用户也许拥有不止一个角色，并且我们可能想实施诸如职责分离之类的策略，那样用户不能在同一个会话中扮演两个角色。
- 一个角色可以分配给单个用户或用户组。

解决方案

延伸上一个模式的概念，把角色当作主体，图 4 展示了基于角色访问控制的一个基本模型。类 User 和 Role 分别表示已注册的用户和预定义的角色，用户被赋予角色，按照角色的职能，赋予用户特定权限。关联类 Right 定义一个角色用户被授权访问保护对象的权限类型。实际上，综合 Role、ProtectionObject 和 Right，就是一个授权模式的实例。同样，谓词表示依内容而定的限定，用来选择特定对象，copy_flag 属性表示一个布尔值，true 或 false。

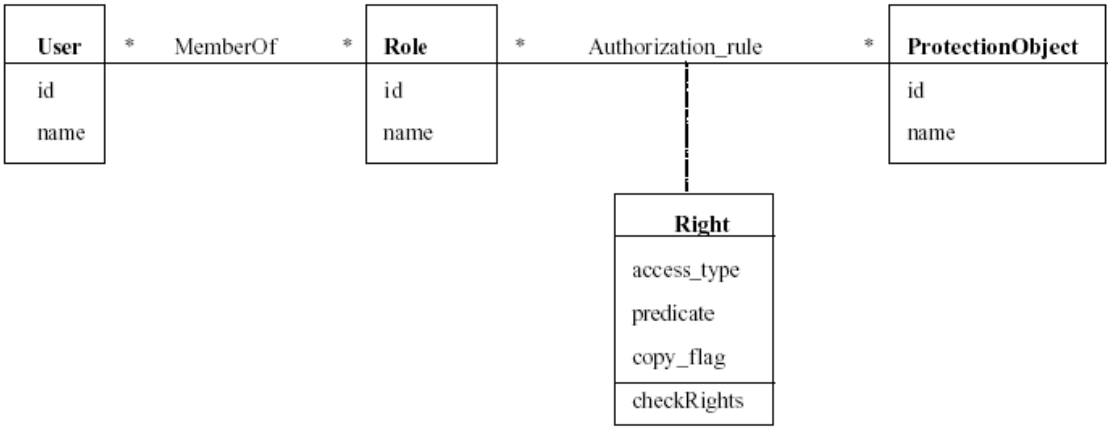


图 4 基本 RBAC 模式

图 5 中的模型考虑到复合角色（Composite 模式的应用），以及管理权限和其他权限的分离（职责分离策略的应用），这个模型同时也包含了一个约束条件来实施职责的分离。管理员具有为组和用户赋予角色的权限，它是一个特殊的用户，可以为用户赋予角色和为角色分配权限。安全管理的权限通常包括：

- 角色授权规则的定义。
- 用户组的创建、删除。
- 用户的角色赋予。

图 5 还包括了会话（Session）的概念，对应于执行时，用户排他方式地扮演某个角色。最后，Group 类描述一组可以被赋予同样角色的用户。

结论

这个模式的优点如下：

- 允许管理员降低安全性的复杂度，用户要比角色多得多。
- 机构关于工作岗位的策略可以直接反映为角色的定义和用户的角色赋予。
- 为得到更大的灵活性和规则的简约性，角色可以进行结构化。
- 为了功能的灵活性，用户可以同时激活多个会话，有些任务可能需要多个视图或不同类型的行为。
- 我们可以增加 UML 约束，以表明有些角色不能用于同一个会话或被赋予给同一个用户（职责分离）。

- 用户组可以作为角色成员，这样可以大大降低授权规则和角色分配的数目。

可能的缺点包括：

- 附加的概念复杂度（新的角色概念，以及多重角色赋予等概念）。

另外还有一些可能的角色结构[Fer94]，在特定环境中比较有用。同时，使用角色来扩展第 5 节的多级模型也是可以的。

已知应用

我们的模式使用面向对象方式表示在[San96]中描述的模型，那个模型已经成为多数研究论文和实现 RBAC 概念的基本原理[FBK99]。RBAC 在很多商业系统中都有应用，包括 Sun J2EE[Jaw00]，Microsoft 的 Windows 2000，IBM 的 WebSphere 和 Oracle 等等。Java JDK1.2 的基本安全模块已经表示可以支持很多种 RBAC 策略[Giu99]。

相关模式

在[Fer93]和[Yod97]中出现一个更简单的版本（相对于图 3），在[Kod01]中，有一个软件实现的模式语言（没有考虑合成角色、组和会话）。图 5 的模式包含了授权模式和合成模式，其他相关模式还有角色（Role）模式[Bau00]和抽象会话（Abstract Session）模式[Pry00]。

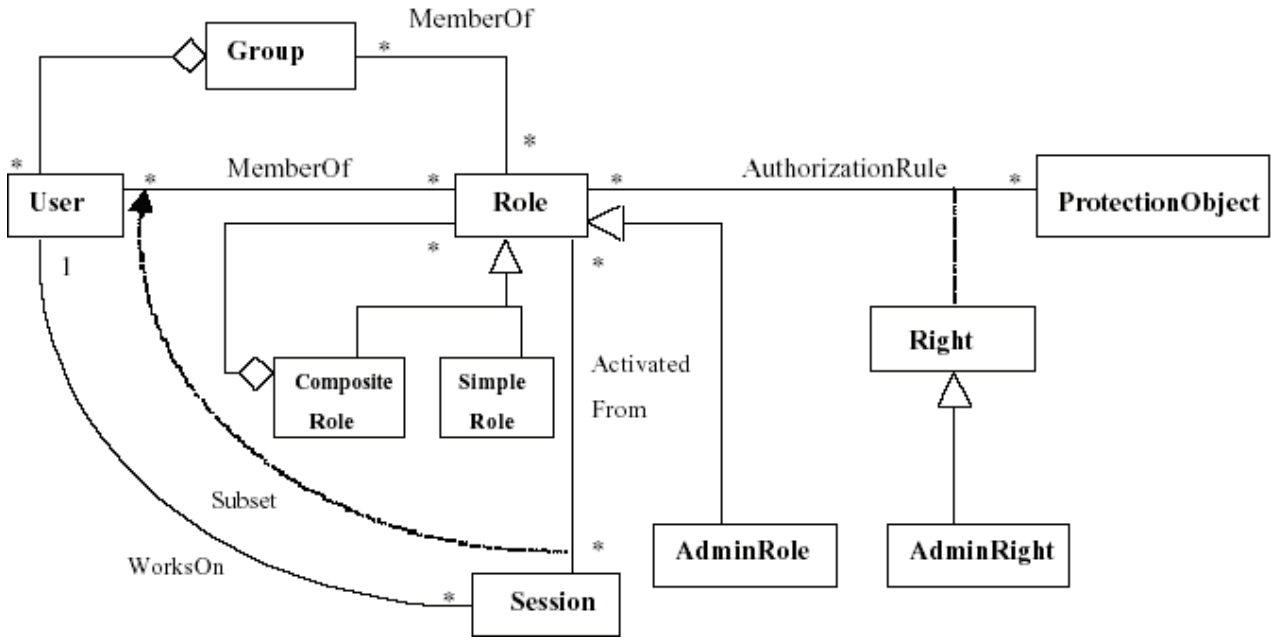


图 5 RBAC 模式

4 多级安全模式

上下文

在某些环境中，数据和文档具有不同敏感级别，例如秘密级、机密级。用户可以凭借许可证访问文档。

问题

如何在安全分级的环境中决定访问方式。

先决条件

- 模型应该根据数据的敏感度来保护它的机密性和完整性。
- 模型应该能够用于所有的架构层。
- 能够有不同的规则组来决定访问方式。
- 必须有一种便利的方式为用户和数据分配分类级别。

解决方案

在这个模型中，用户和数据被赋予分类或许可证。分类包括级别（高度机密、机密等）和分区（工程部、市场部等）。根据 Bell-Lapadula 模型定义的规则实现用户对数据的机密性访问，通过 Biba 模型[Sum97]定义的规则来实现完整性。图 6 展示了这个模型的基本结构。

但是，基本模型不允许级别分配，所以需要一组特定授信的进程来分配级别和修改分类，也就是完成所有管理的功能。

结论

优点包括：

- 用户和数据的分类比较简单，可以根据机构策略进行。
- 在可靠假设前提下被证明是安全的[Sum97]。

可能的缺点包括：

- 通过为数据标上标签以表明它们的分类，这样确保了安全性。但如果没有这么做，将反而降低整个安全度。
- 我们需要附加的受信程序为用户和数据分配级别。
- 数据需要能够结构化成层次化的且具有敏感分级的，同时用户也要能结构化成许可证形式，这通常难以做到，或者在商业环境中是不可能的。

已知应用

这个模型已经用于若干军事项目和一些商业产品中，包括 DBMS（Informix）和操作系统（Pitbull[Arg]和 HP 的 Virtual Vault[HP]）。

相关模式

角色的概念也可以用在此处，角色分类可以代替用户分类。

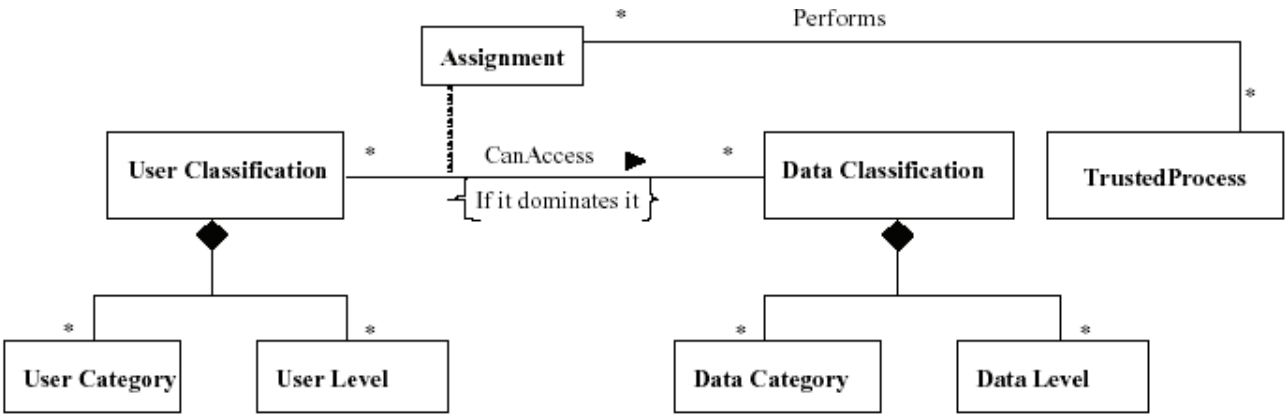


图 6 多级安全模式的类模型

5 底层

上面这些模式定义了抽象的模型，可以应用于系统的所有层。在每一层可以使用该层具体的实体来表示抽象模型的概念，以形成更加具体的模式。为得到一个安全的系统，有必要为每层定义模式语言，这将留作以后的工作，这里提供一个示例。

第 3 节的授权模式可以应用于操作系统中用户从工作站访问文件的情况，这是一个图 1 中系统层的模式。

5.1 文件授权模式

上下文

操作系统的用户需要定义文件，这些文件可以从被不同授权的工作站访问，并且对文件的访问只限于授权用户。

先决条件

对于这种情况，具有如下先决条件

- 可能有不同类型的主体，如用户、角色和组。
- 主体可以被授权访问文件、目录和工作站。
- 一个主体在每个已授权工作站上都有一个 home 目录，但是同样的 home 目录可以在多个工作站或主体之间共享。
- 用户可以形成组来访问。
- 有些系统使用角色代替用户作为主体，或两者都作为主体。
- 操作系统中文件系统具有不同的实现方法。

解决方案

把授权模式特殊化，以文件代替对象，以访问许可代替权限，如图 7。它定义了文件访问，工作站方式也是类似地应用授权模式定义。文件和目录是递归结构的，结果是一个语义分析模式（Semantic Analysis pattern SAP），组合了两个版本的授权模式成为一个合成模式。一个 SAP 是对应于一组基本用例的分析模式[Fer00]。

结论

这个模式可能的优点：

- 它可以使用于多种主体，包括用户、角色和组。
- 被访问的对象可以是单个文件、目录，或者是递归结构的目录和文件。

- 隐式的授权是可能的，例如，访问一个目录可以隐式指定以类似的方式访问该目录下所有的文件。

一些缺点如下：

- 该模式的实现不必依据访问矩阵模型。例如，Unix 使用一种伪访问矩阵，使用 need-to-know 策略并不恰当。但可以为这个模式加上约束，强迫模式所有的示例都遵从一个访问矩阵模型。

其他方面包括：

- 有些系统对工作站不能使用授权。
- 多数操作系统使用读、写、执行作为访问类型，更高层类型的访问也是有可能的。
- 多数操作系统有所有者的概念，它是一个特殊的用户，对它创建的文件拥有所有的权限。
- 有些系统中，文件映射到虚拟内存地址空间，这个模式同样适用于这样的情况。

已知应用

这个模式出现在 Unix、Windows、Linux 和多数现代操作系统中。

相关模式

这是授权模式的一个特例，如果使用角色，就还和基于角色访问控制模式有关，文件系统使用了一个合成模式。

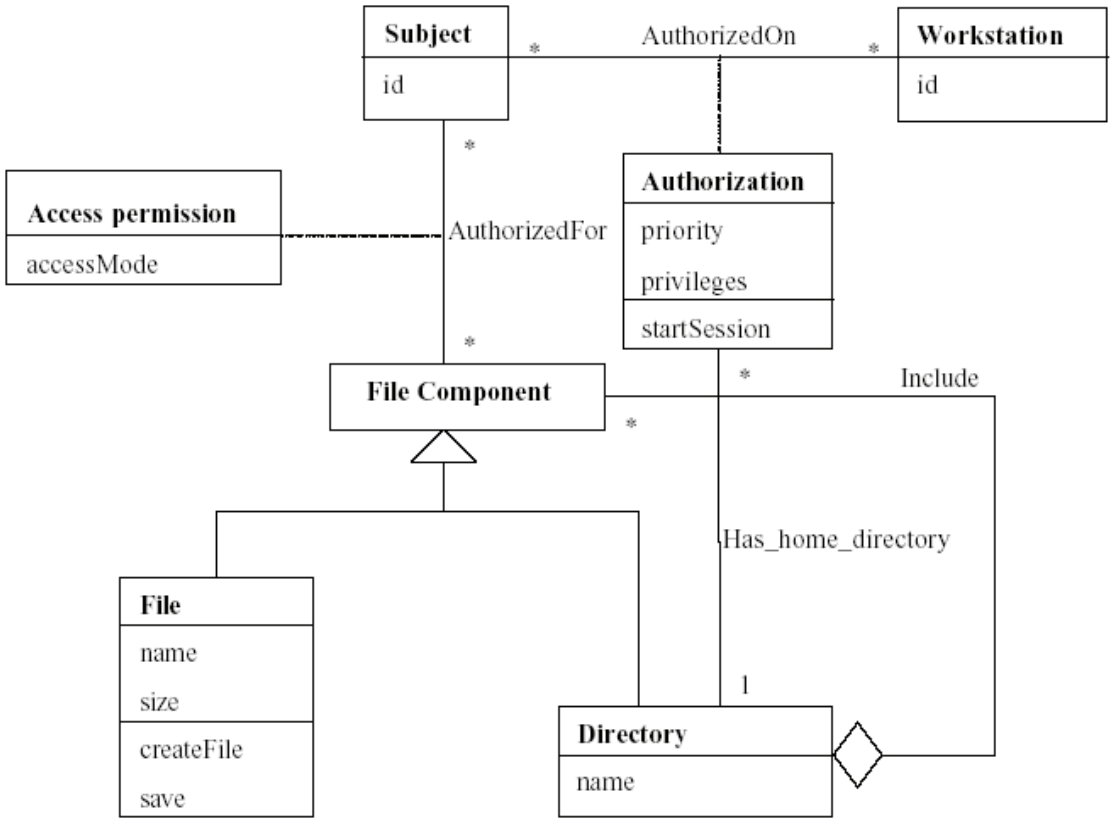


图 7 文件授权的一种模式

6 模式的一般讨论

以上这些模式的具体实现要看它应用在架构中的哪一层，其中一个重要的问题是如何以较低的负荷来进行访问控制。授权模式可以通过访问控制列表（ACL）来实现，多数操作系统就采用这种方法。另一种方式是使用 Capablities，硬件和操作系统层控制资源即采用这种方式。近来有些文章描述应用控制对类的访问[Fra99]，使用元类和反射机制来实现这些模型是另一种有趣的方式[Wel99]。对资源的请求必须被 ACL 或 capabilities 中的信息解释和验证，这需要一个特殊的程序（另一种模式），叫做 Reference Monitor[Sum97]，一个可能的实现就是 POSA2 中的 Interception 模式[Sch00]。

已经提出的某些架构层的模式：

- Yoder 和 Barcalow[Yod97]描述了单访问点（Single Access Point，一种通用安全系统设计模式），检查点（Check Point，一种 Reference Monitor）、角色（见第三节）以及其他等。
- Fernandez 讨论了元数据和模式之间的关系[Fer00a]。第三节的 RBAC 模型最先出现在此。

- Cryptographic 模式描述在[Bar00]。
- 关于安全模式，包括一些网络安全模式的一般性讨论位于[Sch01]。
- 一个访问控制、过滤分布对象的框架，组合了若干模式，位于[Hay00]。
- Java 的若干安全模式位于[Jaw00]。

对于在每一层增加更多的模式，收集和统一这些模式的工作还有待继续进行。

还有一些基于其他策略或策略组合的安全模型。其中重要的两个模型是 Clark-Wilson 模型和 Chinese Wall 模型[Sum97]。Clark-Wilson 模型着重强调完整性，并定义两个基本原则：

- 组织良好的事务——确保对数据的修改保持在一致的状态
- 职责分离——改变必须至少被两方认可。

Chinese Wall 模型与其说是模型，不如说是一种策略。信息被分组成若干利益冲突的类，一个人在每个类中至多允许访问一组信息。

目前有 Clark-Wilson 模型的模式[Wel99]，但是没有 Chinese Wall 模型的模式。

参考文献

[Arg] Argus Systems Group, "Trusted OS security: Principles and practice", http://www.argus-systems.com/products/white_paper/pitbull

[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

[Bra00] A. Braga, C. Rubira, and R. Dahab, "Tropyc: A pattern language for cryptographic objectoriented software", Chapter 16 in *Pattern Languages of Program Design 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.). Also in *Procs. of PLoP'98*, http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/

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

[Ess97] W. Essmayr, G. Pernul, and A.M. Tjoa, "Access controls by object-oriented concepts", *Proc. of 11th IFIP WG 11.3 Working Conf. on Database Security*, August 1997.

[FBK99] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn, "A Role-Based Access Control model and reference implementation within a corporate Intranet", *ACM Transactions on Information and System Security*, Vol. 2, No. 1, Feb. 1999, pages 34-64 .

[Fer75] E. B. Fernandez, R. C. Summers and T. Lang, ``Definition and evaluation of access rules in data management systems,'' *Proceedings 1st International Conference on Very Large Databases,* Boston, 1975, 268-285.

[Fer93] E.B.Fernandez, M.M.Larrondo-Petrie and E.Gudes, "A method-based authorization model for object-oriented databases", *Proc. of the OOPSLA 1993 Workshop on Security in Object-oriented Systems* , 70-79.

[Fer94] - E. B. Fernandez, J. Wu, and M. H. Fernandez, ``User group structures in object-oriented databases", *Proc. 8th Annual IFIP W.G.11.3 Working Conference on Database Security,* Bad Salzdetfurth, Germany, August 1994.

[Fer99] E.B.Fernandez, "Coordination of security levels for Internet architectures", *Procs. 10th Intl. Workshop on Database and Expert Systems Applications*, 1999, 837-841.

[Fer00] E.B. Fernandez and X. Yuan, "Semantic Analysis Patterns", *Procs. of the 19th Int. Conf. on Conceptual Modeling (ER2000),* 183-195.

[Fer00a] E.B. Fernandez, "Metadata and authorization Patterns", Report TR-CSE-00-16, Dept. of Computer Science and Eng., Florida Atlantic University, May 2000.

[Fra99] G. Frascadore, "Java application server security using capabilities", *Java Report*, March 1999, 31-42.

[Giu99] L. Giuri, "Role-Based Access Control on the web using Java", *Procs. of RBAC'99*, ACM 1999, 11-18.

[Hay00] V. Hays, M. Loutrel, and E.B.Fernandez, "The Object Filter and Access Control framework", *Procs. Pattern Languages of Programs (PLoP2000) Conference*, http://jerry.cs.uiuc.edu/~plop/plop2k

[HP] Hewlett Packard Corp., Virtual Vault, http://www.hp.com/security/products/virtualvault

[Jaw00] J. Jaworski and P.J. Perrone, *Java security handbook*, SAMS, Indianapolis, IN, 2000.

[Kod01] S. R. Kodituwakku, P. Bertok, and L. Zhao, "A pattern language for designing and implementing role-based access control", *Procs. KoalaPLoP 2001*.

[Pfl97] C.P. Pfleeger, *Security in computing*, (2nd Ed.), Prentice-Hall 1997.

[Pry00] N. Pryce, "Abstract session: An object structural pattern", Chapter 7 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

[San96] R. Sandhu et al., "Role-Based Access Control models", *Computer* , vol. 29 , No2, February 1996, 38-47.

[Sch00] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-oriented software architecture, vol. 2 , Patterns for concurrent and networked objects,* J. Wiley, 2000.

[Sch01] M. Schumacher and U. Roedig, "Security engineering with patterns', submitted to PLoP 2001.

[Sum97] R. C. Summers, *Secure Computing: Threats and Safeguards*, McGraw-Hill, 1997.

[Wel99] I. Welch, "Reflective enforcement of the Clark-Wilson integrity model", *Procs. 2nd Workshop in Distributed Object Security*, OOPSLA'99, ACM, November 1999, 5-9.

[Woo79] C. Wood and E. B. Fernandez, ``Authorization in a decentralized database system," *Proceedings of the 5th International Conference on Very Large Databases*, 352-359, Rio de Janeiro, 1979.

[Yod97] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security". *Procs. PLOP'97*, http://jerry.cs.uiuc.edu/~plop/plop97 Also Chapter 15 in *Pattern Languages of Program Design*, vol. 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.), Addison-Wesley, 2000.

一种关于物品修理的分析模式

Eduardo B. Fernandez, Xiaohong Yuan 著, 刘巍 译

查看评论

摘要

文中提到的分析模式用于对修理店中的修理活动建模。(本模式)着眼于修理店中修理活动的基本情况，从最初的(修理情况)估算直到整个修理过程的结束。至于修理店的细节情况和被修理的物品留待在专门的应用或其他补充的模式中去讨论。这种模式代表的最小应用可用于各种情况下而且可以与其他相关模式共同用于描述更复杂的应用。也可被进一步抽象用于描述其他情况而不仅仅是修理过程。

1 介绍

我们介绍一种用于描述修理店中的修理过程的分析模式。这种模式属于我们所谓的语义分析模式[Fer00a]的范畴，所以本文提到的模式仅注重对修理店中的修理活动的基本面的描述，而不去关心修理店、修理的物品或顾客的具体类型。至于修理店的细节情况和被修理的物品留待在专门的应用或其他补充的模式中去讨论。这类模式目的在于给出一个将需求转化为具体的设计的起点。这类模式代表的最小应用可用于各种情况下，而且可以与其他相关模式共同用于描述更复杂的应用。也可被进一步抽象用于描述其他情况而不仅仅是修理过程。

把一件损坏的东西拿到修理店去修理，在日常生活中是一件很平常的事。顾客将损坏的东西，如电脑或者汽车送到修理店时，一个负责接待的技工会对要做的修理作出大致的估算，他同时会对修理的内容作纪录。所有修理活动都被记录到修理日志中。一次修理可能因为缺少零件或其他原因而被耽搁，也可能被取消。

2 目的

本文提及的模式描述了修理店中的修理活动，包含了对要做的修理预先作个估算、指派某些技术工人去修理和对修理的项目进行记录。

3 上下文

顾客将电脑送到电脑修理店是类似的修理店中修理过程链条中一部分（如图一）。顾客可能将电脑送到几个修理店中进行估算（Estimate类），然后再选择一家店去修理。一个接待技工作修理的估算工作，而修理工负责电脑的修理和对修理项目的记录（RepairEvent类）。每个电脑都有一个在这些修理店中所有修理项目的记录（RepairLog类）。

上述例子是普通的物品修理如汽车、手表、家用电器、复印机等很多种类修理中的一个特殊情况。为了使这个分析模式用于概念上的建模，我们将所有这些方面的例子进行抽象，并定义一个通用的模式涵盖所有的情况。通用模式可被修改以适用于特定的情形。

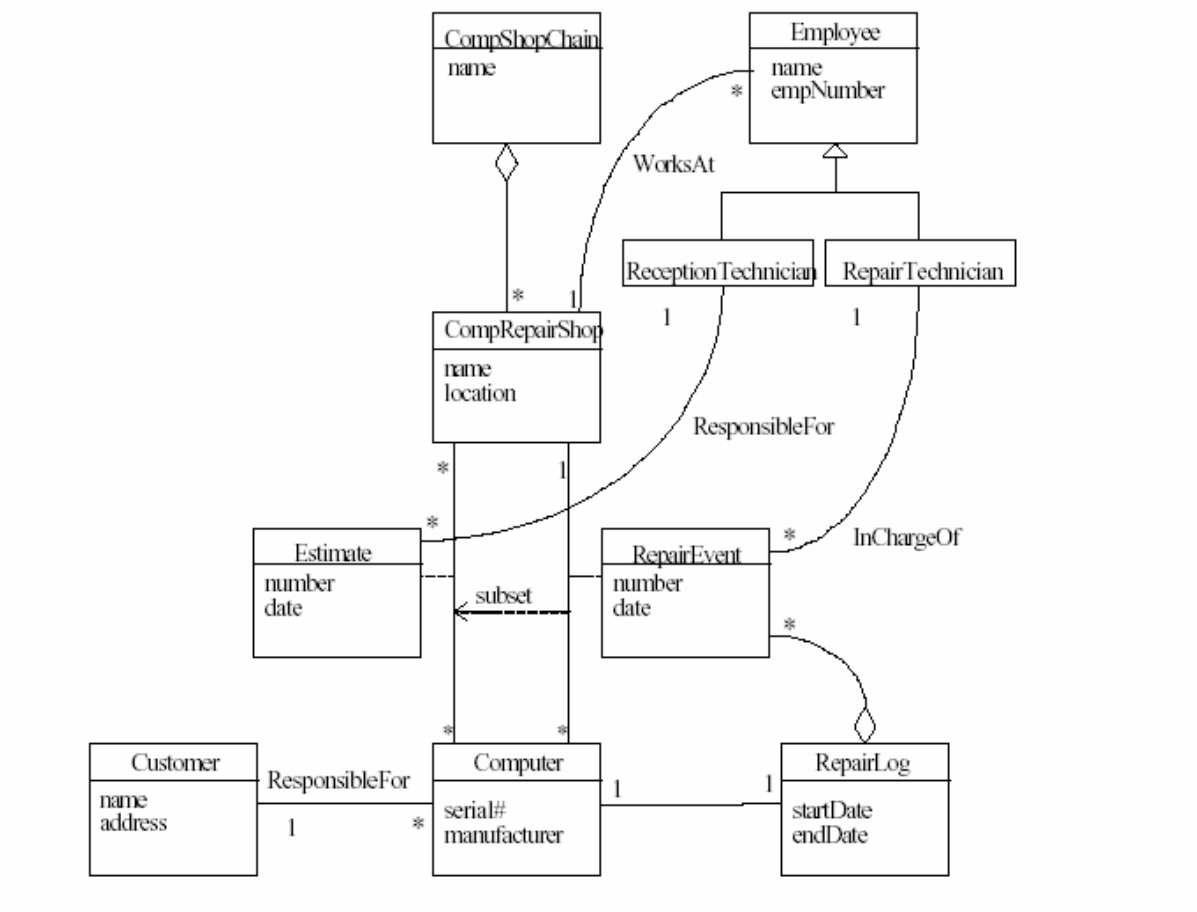


图 一. 电脑修理店的类图

4 问题

如何对修理店中的修理活动建模呢？我们需要开发出一个包括对欲修理物品进行估算、指派修理工进行修理、记录修理项目的初始模型。

5 约束

- 概念模型很难建立。修理店的修理活动尤其不是一个简单的问题。一个初始的通用模型可以起到很多帮助。
- 我们想为修理店的修理活动建模。然而，有很多情况都具备相似的结构而在细节上不同。
- 用户会到几个地方对修理服务进行预估，然而只会到其中一家进行修理。
- 模型必须包含描述实际情况的文档表示，如估算、修理活动、修理记录。否则建立这些必要的文档会很复杂。

6 解决方案

6.1 需求

本解决方案适用于实现下述用例：

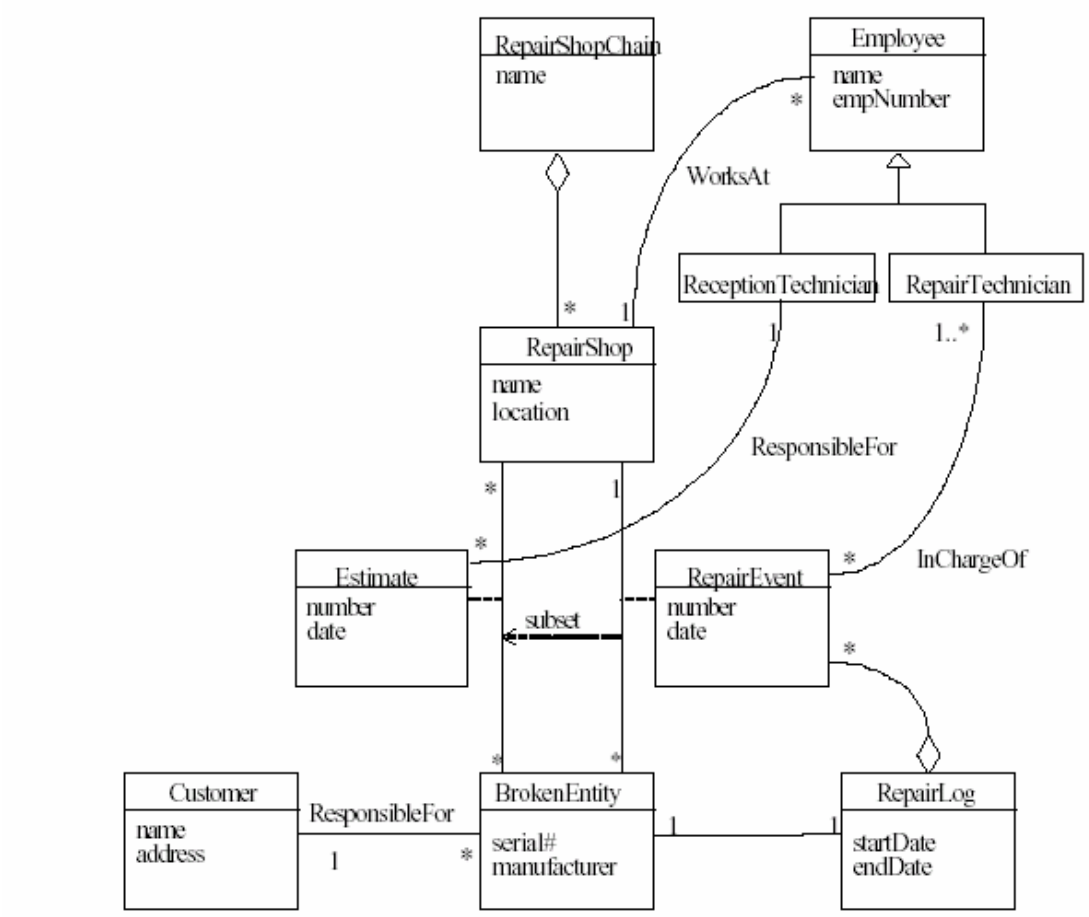
(1)对修理进行估算。接待技工会根据客户描述的问题，检查待修理的物品并给出一个修理价格的估算。

(2)对物品进行修理。客户决定在此进行修理后，修理工会被指派对物品进行修理，修理过程可以由于不同的原因而被中止。

6.2 类模型

图二是实现这些用例的类图，是图一经过提取和扩展而形成。一个**BrokenEntity**（待修物品）定义为**RepairShop**（修理店）的修理对象，**Customer**（顾客）拥有**BrokenEntity**（待修物品）的所有权（负担修理费用）。**BrokenEntity**和**RepairShop**之间的“多－多”关系反映了修理估算可以在许多修理店中进行。它们之间的“多－单”的关系表述了上述的估算中的一个可能在实际中用到。至少被修理过一次的电脑会有关于这些修理情况的记录。所有修理店的合集被指定为类**RepairShopChain**（修理店链）。修理店的雇员中典型地被分为两类角色：

ReceptionTechnician（接待技工），负责修理估算；RepairTechnician（修理技工），负责执行修理工作。



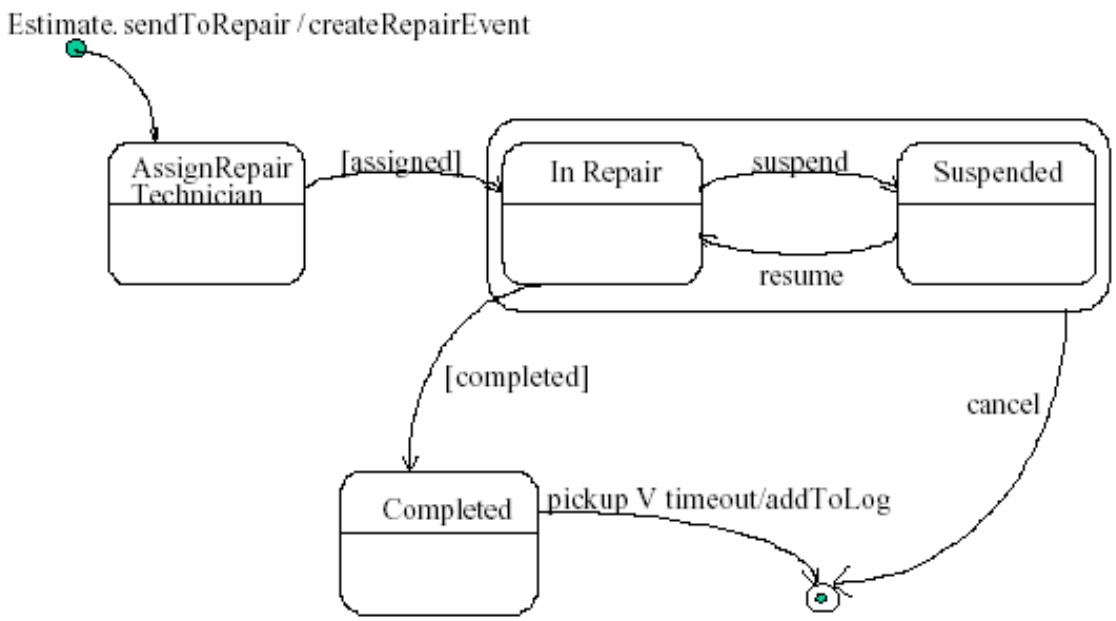
图二. 修理店模式的类图

6.3 动态模型

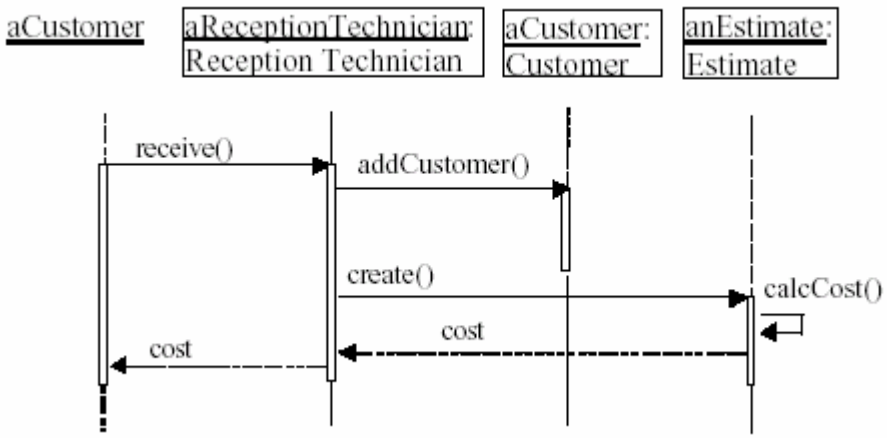
图三表示了类RepairEvent（修理活动）的状态图。修理活动的状态有*AssignRepairTechnician*（指派修理技工）、*InRepair*（修理进行中）、*Suspended*（修理暂停）和*Completed*（修理结束）。包含*InRepair*和*Suspended*的父状态含有修理取消可以从这两个状态中发生。

图四是进行一次估算的序列图，此处假定顾客是新客户，并将此信息添加到客户列表中。

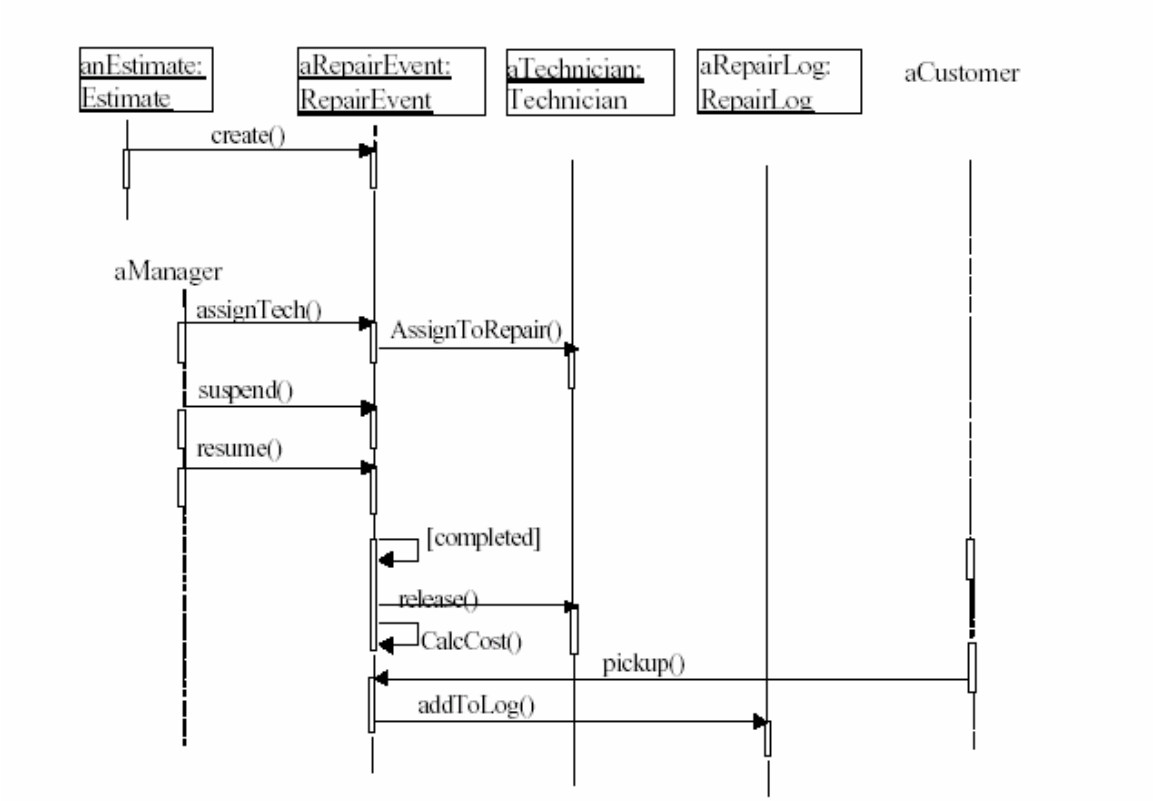
图五显示了指派修理技工工作内容的序列图。



图三. RepairEvent类的状态图



图四、修理估算的序列图



图五、修理工作指派序列图

7 结论

这个模式具备下述优点：

- 提供了现实生活情况（见第8部分）的一个通用的描述模式，即可在不同的应用中使用。
- 典型文档用类来描述，例如估算。
- 为修理活动提供了一个系统化的结构，进而优化了修理店中修理活动的处理。

缺点在于这种模式描述的情形并不都很相似：

- 在例如修理手表或修理汽车的情形，顾客将待修物品带到修理店；也可托运到修理店。在修理一个冰箱的情况下，修理工必须要上门修理。

• 接待技工有时可以根据顾客对问题的报告直接估算出修理费用。而在其他情况下，必须派一个技工进行检查并将（损坏）情况报告后，据此给出修理费用的估算。不管是否在此处修理，顾客可能因为这个检查而付费。一般负责检查的技工会在顾客决定修理后担任修理工的角色。

• 一些修理店会专注于某个特定类型的甚至是某个特定厂商或特定型号的产品的修理。另一些修理店则提供许多类型产品的修理服务，例如冰箱、洗碗机、空调设备和其他家用电器。

• 顾客可能与修理提供商签订一个合同，不必为每次修理单独付费。而是每年根据合同付出一定数量的费用。这样，修理费用就不必进行估算。

• 在产品的保修期（一定时间）内，顾客不必为修理服务付费。所以修理费用在此情况下也不必进行估算。

• 根据修理工是否有空闲，顾客可能要等待数小时或甚至数天才能获得服务。修理的时间也会因为待修物品问题的不同而不同。

上述这些说明这个模式需要进行修改以适应于特定的应用。

模式中没有提到的一些方面有：

• 待修物品的周围环境和前后关系因素的描述

• 计费和付费的策略

• 如何处理顾客的差异因素，例如个人客户、公司客户、首选客户等

• 例外情况，例如客户取回了修理的物品（对修理不满意）

• 如何在没有修理所需零件时定购所需零件

• 如何跟踪零件库存情况

• 如何在没有空闲修理工时对顾客的修理需求进行排队

• 当顾客需要送货时，如何将物品送到顾客处

• 顾客在有修理需求时，先与修理工约定好时间，然后再将待修物品送到修理店或预先约定时间由修理工到用户家里（或其他地方）上门修理的情况。

为使这个模式更通用，上述这些情况没有在考虑范围内。这些情况在补充的模式中或者在直接的应用模型中会有描述。

8 已知应用

以下是该模式应用的例子：

- 一名顾客将汽车送到修理店（或者汽车经销商）处修理的案例
- 一名顾客将损坏的计算机送到计算机商店去修理的案例
- 一个公司为一台复印机预约修理的案例
- 一名顾客预约一台冰箱的上门修理的案例

9 相关模式

当修理所需零件不足时，修理店需要定购零件。所以Order/Shipment [Fer00b]（订货/发货）模式对此进行了补充（编注：见《非程序员》第14期）。顾客可能需要与修理工提前预约时，Reservation（预约）和Use（使用）模式[Fer99]可以被用到（编注：见《非程序员》第7期）。Stock Manager（库存管理）模式[Fer00c]也是补充模式（编注：见《非程序员》第7期），可以被用于跟踪修理店的零件库存情况。考虑到订单付费的资金方面的处理细节可以在[Hay96]和[Fow97]中找到。

本模式包含两个其他模式：Collection（集合）模式（修理店是修理店的集合???），角色模式[Bau00]（雇员在修理技工和接待技工中的角色）。

这个模式可以更通用[Fer00a]，用来给修理活动建模的结构同样也可用于描述申请入学许可或申请几个机构的服务并选择其中之一这类活动。例子包括接收入（医）院和学生申请和注册。如此泛化在建立概念模型时很有用；然而，这样需要根据具体情况要改动的部分要更多。

鸣谢

我们要感谢我们的主管Mary Lynn Manns女士在论文写作中提出的宝贵建议，并为本文的成文提供很大的帮助。

参考文献

[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

[Fer00a] E.B. Fernandez and X. Yuan. “Semantic analysis patterns”, *Procs. of 19th Int. Conf. on Conceptual Modeling, ER2000,* 183-195.

[Fer00b] E. B. Fernandez and X. Yuan. “Analysis patterns for the order and shipment of a product”, *Procs. of PloP 2000.* http://jerry.cs.uiuc.edu/~plop/plop2k

[Fer00c] E. B. Fernandez. “Stock Manager: an analysis pattern for inventories”, *Procs. Of PloP 2000.* http://jerry.cs.uiuc.edu/~plop/plop2k

[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.

[Hay96] D. Hay, *Data model patterns-- Conventions of thought*, Dorset House Publ., 1996.

征稿

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

一个产品的订货和配送（系统）的分析模式

Eduardo B. Fernandez, Xiaohong Yuan, and Sandra Brey 著，li.sining 译

查看评论

摘要

本文给出的这些分析模式描述了顾客提交产品订单和订货之后的产品的配送过程。我们首先给出了两个基本的模式－－订单和配送，然后把基本模式组合成为一个我们称之为语意分析的模式，与实际应用中增加灵活性相反，我们强调应用模型的语意方面。这类模式的目的是作为把需求转化为真正系统设计的起点。这个模式代表一个最基本的应用，可以把它应用在不同的场合，也可以与其它的相关模式组合在一起来描述更复杂的应用。产品的订货和配送是现实生活中一个非常通用的问题，这个模式集中在订单和交货以及它们之间的相关过程的基本特征。

1. 引言

我们给出了顾客如何订货，以及之后的配送的分析模式，首先给出两个基本的模式，产品的订购和配送。然后把基本模式组合成为一个我们称之为语意分析的模式，与实际应用中增加灵活性相反，我们强调应用模型的语意方面。这类模式的目的是作为把需求转化为真正系统设计的起点。这个模式代表一个最基本的应用，可以把它应用在不同的场合，也可以与其它的相关模式组合在一起来描述更复杂的应用。产品的订货和配送是现实生活中一个非常通用的问题，包括顾客对一种特定的产品或者服务订货；例如，食物、书籍、磁带等，接下来就是把产品交付给顾客并结账。订货模型集中在订单的基本面，没有涉及产品和顾客的详细特征。配送模型描述了订购产品的运送。它们的组合模型的重点是把订单和相应的完成过程结合在一起。产品的详细资料如制造和可用性等留给具体的应用程序或者补充模型来完成。

2. 订单模式

最基本的模式，描述订单本身。

环境（上下文）

在许多实际应用场合中需要订购产品或者服务，例如：饭店订餐、在电子商务公司订购书籍等等。

问题

如何描述订购一种产品或者服务（的过程）。

要求

- 定单必须被准确获取。
- 订单的状态必须一直可知。
- 订单必须与最终的结果——发货或者交付有关联。
- 顾客需要的往往是一类产品，而不是某一个产品。

解决方案

图1给出了需求信息的类模型，包括描述订货过程的相关类，提交订单的顾客类和产品类。顾客类（Customer）和订单类（Order）的关联表明：每一个订单都是由一个特定的顾客给出，但是每个顾客可以下许多订单。一个订单包括由许多条目的集合组成，每一个条目表示一类特定的产品和数量。图2表示订单状态随时间的变化。订单触发其它事件产生状态变化；完成后产生一个发货对象，关闭订单并把它的信息写入一个日志文件。描述动作的顺序图由图8给出。

结论

- 这个模式只描述了产品的订货，但是它与发货有关联。
- 订货的对象可以是产品或者服务。
- 顾客可以是一个人或者一个系统；例如：一个商店订单的每一个条目都是要求生产产品的顾客订单。
- 虽然订单中必须包括完整的信息，但是只有几个特征与顾客和产品有关。
- 本例中的产品是指一类产品，不是一个单个的产品。很容易扩展到单个产品。
- 顾客信息和信用检查只在顾客第一次订货时候进行。

已知的应用和相关模式

见第5、6节。

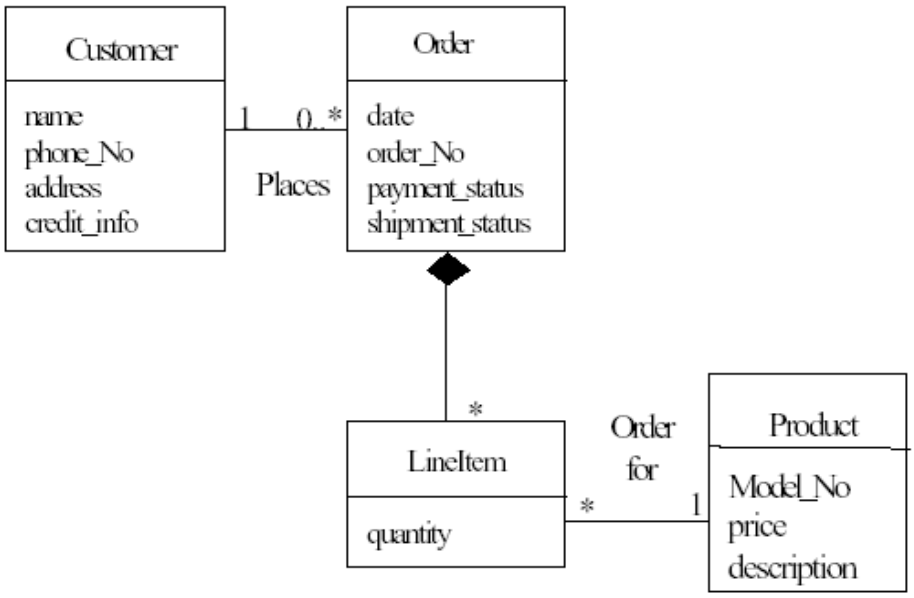


图1. 订单模式的类模型

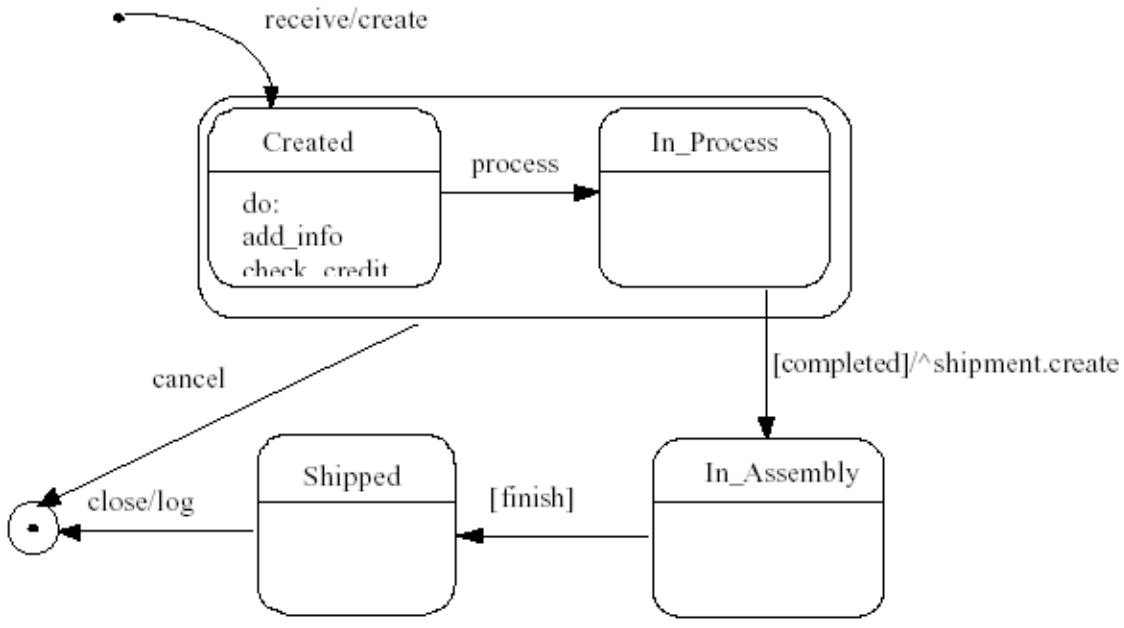


图2. 订单类的状态图

3. 配送模式

环境（上下文）

一个订单已经完成，接着应该将交付给顾客。产品必须按照订单配送。

问题

如何完成一个产品或者一项服务的交付。

要点

- 配送必须按照订单中的描述来完成。
- 配送应该能够描述需要运送的产品和顾客应付的金额（发票）。
- 有相应的顾客作为配送的接收方。
- 产品的运输方式与本模式无关。

解决方案

产品送到顾客手中，他同时收到一张发票作为付款的凭证。图3描述了这些类模型，包括表示这一过程的类：配送、顾客、发票。图4和图5是配送类和发票类的状态图，描述了这些类随事件的变化。当配送完成，生成了发票，关闭配送和发票，写入日志文件，图8是表示这些动作的顺序图。

结论

- 每一个配送必须与对应的订单相关。
- 顾客可以是个人、或者其它系统。
- 送货和结账的细节没有被包括在这个模式里。
- 收货可以与订货不是同一个顾客。
- 发票类描述顾客的应付款金额，对应着一个现实存在的文档凭证。
- 这个模型也可以应用在服务上，配送类代表所要求的服务的交付。

已知的应用和相关的模式

见第5、6节

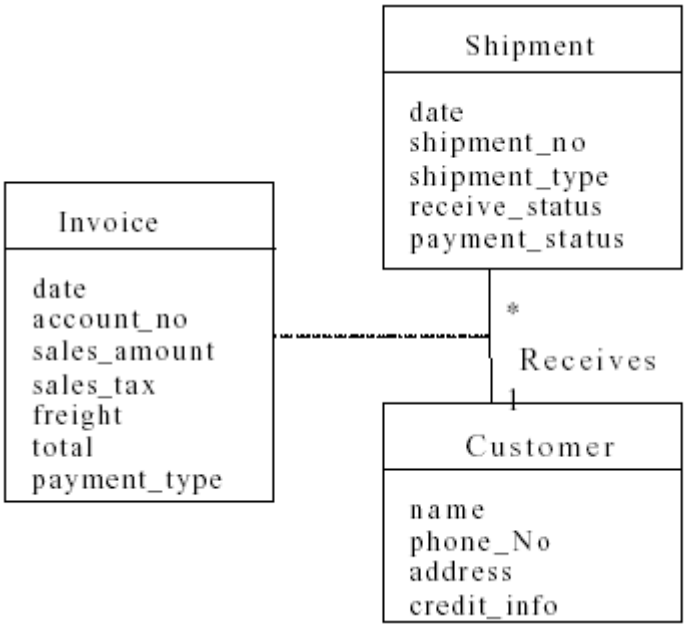


图3. 配送模式的类模型

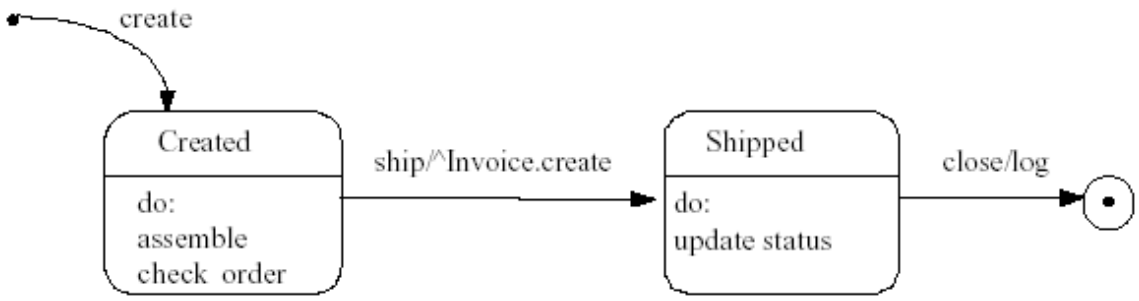


图4. 配送类的状态图

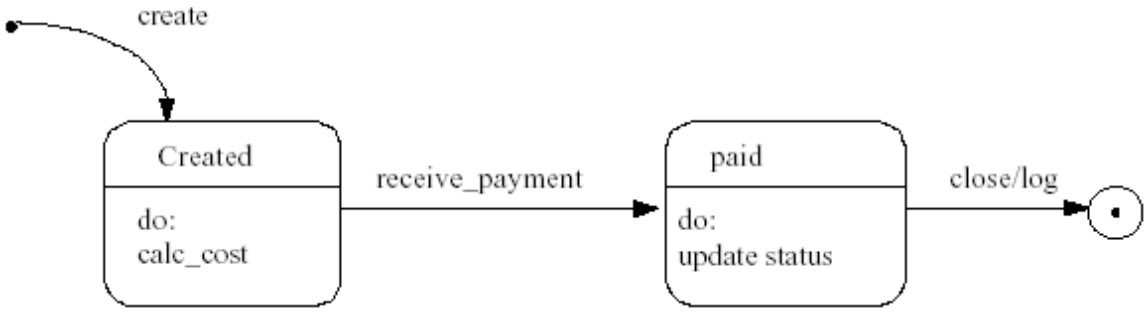


图5. 发票类的状态图

4. 订单/配送模式

目的

这个模式描述产品或者服务的订购以及相应的完成过程。

环境（上下文）

图6给出一个特定的例子，一个顾客从寻呼机生产商那里订购寻呼机。模型中包括的类有：顾客、订单、条目、配送、发票、寻呼机，各个类的含义是显而易见的。配送和订单的关联是：每一个配送都有一个订单相对应，但是一个订单并不一定导致一个配送的产生（例如 订单被取消）。配送产生的发票作为顾客与配送的关联类。

本例是普通订单和配送问题的一个特例，订单和配送问题可以应用于不同的环境，例如：订购某些产品，饭店订餐，预约维修任务。

要求

- 管理机构需要优化订单的完成时间（定性的理解）。
- 管理机构需要跟踪订单的完成情况和顾客的满意度。
- 模型必须包括表示现实生活中的文档凭证，例如：订单、条目和发票。
- 可以用等价产品替换要求的产品。
- 分析模型必须可靠的表示出需求而且不包含具体实现的细节，注意需求可能应用在不同的领域。
- 这个模式必须描述出基本的语意单元。这意味着模式必须足够简化以便可以应用到各种情况。

解决方案

a) 需求

解决方案对应着下列通用用例的实现：

- 接受订单：记录顾客信息（姓名，地址等），检查信用，然后生成订单。
- 取消订单：一个存在的订单被取消，可能用到一些管理机构的策略。
- 订单的交付：按照订单检查配送和服务。生成发票。然后将产品和对应的发票交付到顾客。

这些用例是一般性的，可以对应与不同的应用环境。某些场合其中的一些步骤是隐含的。

b) 类模型

图7是上述用例实现的类图，是图6描述的类图的抽象和扩展，用产品类替代寻呼机类，订单类中的“过程”方法概括了所有产生产品类的必要步骤，同时，配送类中聚集的操作概括了把订单中的不同部分集合在一起的操作。图中还表示出并非所有的产品订单都会最终形成配送或者配送的一些产品可能会与订购的有所不同。

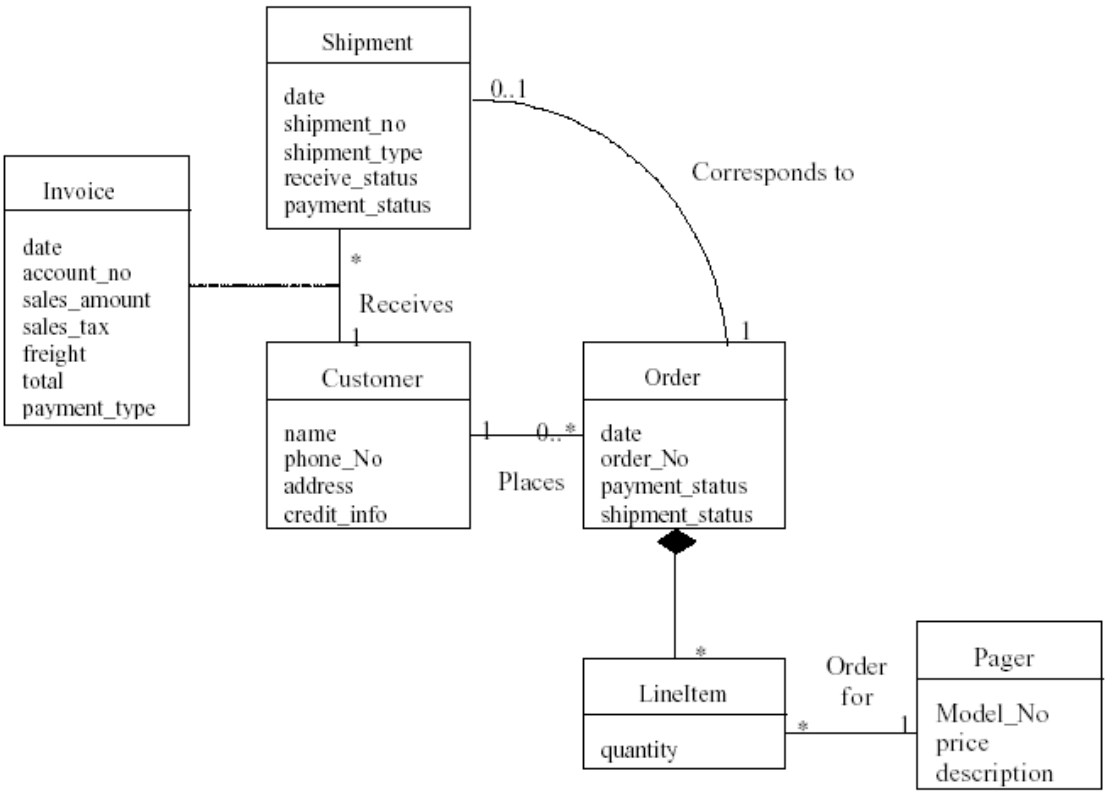


图6. 寻呼机的订货和配送

c) 动态方面

前面已经介绍了订单、配送和发票的状态图。图8的顺序图描述了顾客如何提交产品订单和之后的产品配送过程。图9描述了订货和配送的活动图。注意，在订单的关闭事件中状态图仅显示“关闭”，没有详细说明什么过程关闭订单，而顺序图和活动图显示是结账的事件关闭订单。在图8中，接收产品和结账的顺序可以互换。

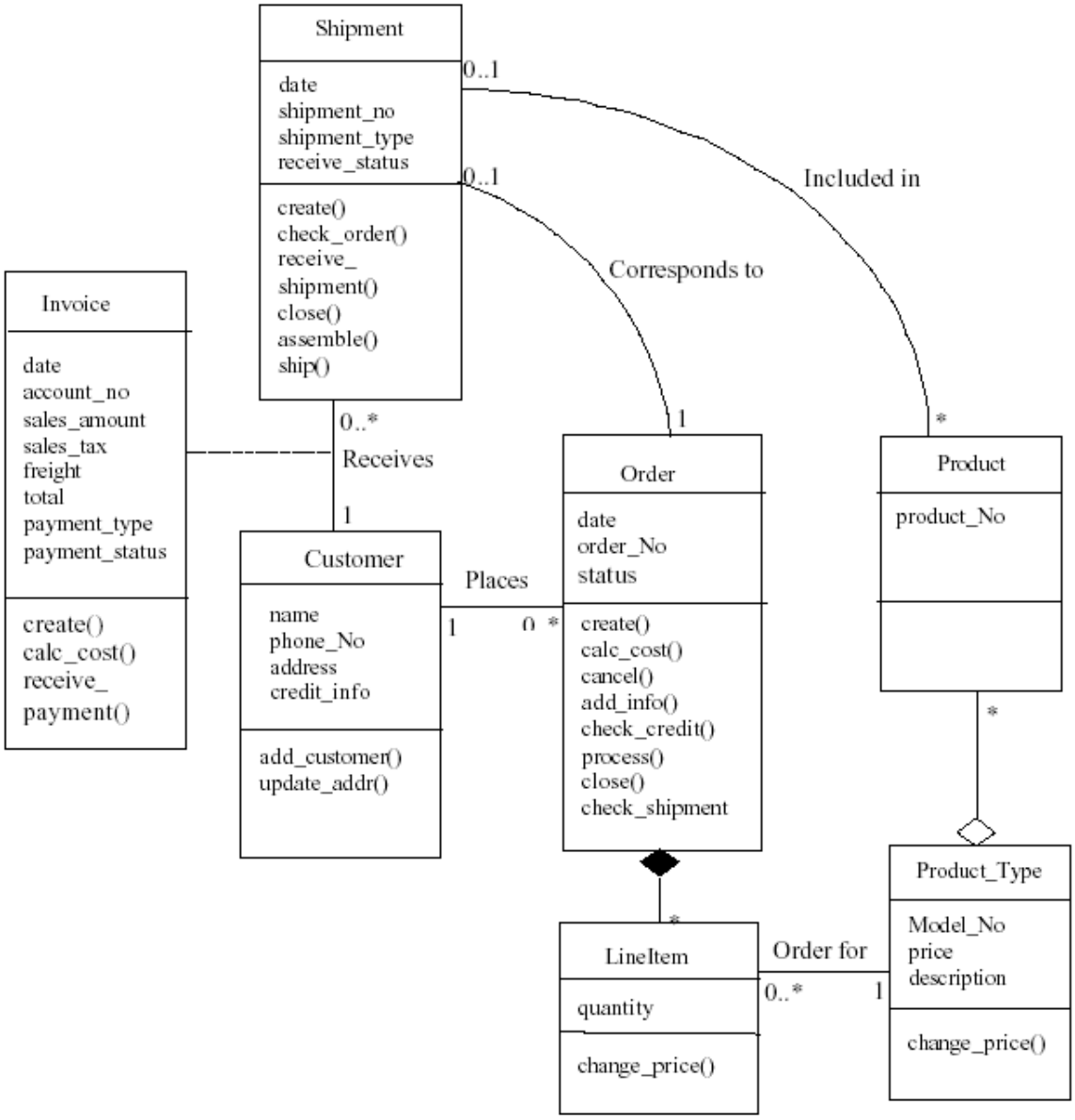


图7. 一个产品的订货和配送的类图

d) 结论

上面的要点对所有这个模型的实现是通用的：

- 一个订单总是按照顾客提出的需求在“系统”中生成。在现代生产工厂或者网上商店，订单几乎肯定是采用数据库中的一条记录的形式。同样的，在历史悠久的快餐企业如Taco Bell，订单是作为计算机系统中的一个虚拟的实体存在的，通过在售货柜员机的一个特殊的按键（图标）即可输入。订单（无疑是虚拟的）可以从顾客的网上请求自动生成。在许多传统的饭店，订单包含账单，由服务生按照顾客的要求手写而成。

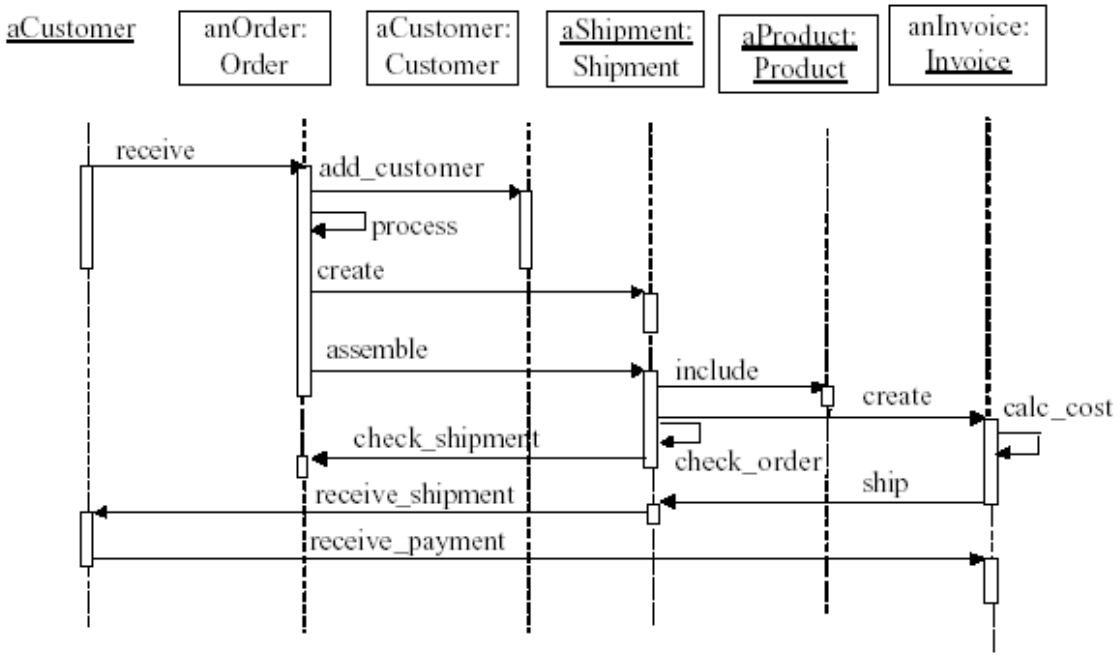


图8. 预定和接收产品的顺序图

- 一个订单总是与一个顾客相关：这意味着，对应的顾客一定能够被识别。在快餐店的滚动流程中，这很简单，可以通过打印的收据上的流水号、饭店账单上的餐桌号码来识别顾客，或者通过更为复杂的系统，使每一个订单都通过一个指针（例如：‘顾客编号’）连接到一个顾客数据库，其中包含信息如姓名、地址、电话、账号状态、折扣等级、购买历史等等。

- 在一条订单的记录上，可以推断下面三种情况之一会发生：

a). 产品有库存，然后订单被执行。

b).产品无库存，商家必须订购或者生产产品。

c).订单在配送之前被取消。

- 总要生成某种文档凭证，一份拷贝作为归档资料，一份拷贝与产品一起送交顾客：

- 这里可以采用现金收据，或者发票，或者两者都要，决定于付款和送货的顺序。凭证包括重要的信息如时间、产品清单和应付（或者已付）价格。

- 订单可以是单个产品，不是一类。

- 产品的接收顾客可以是另外的子系统或系统。

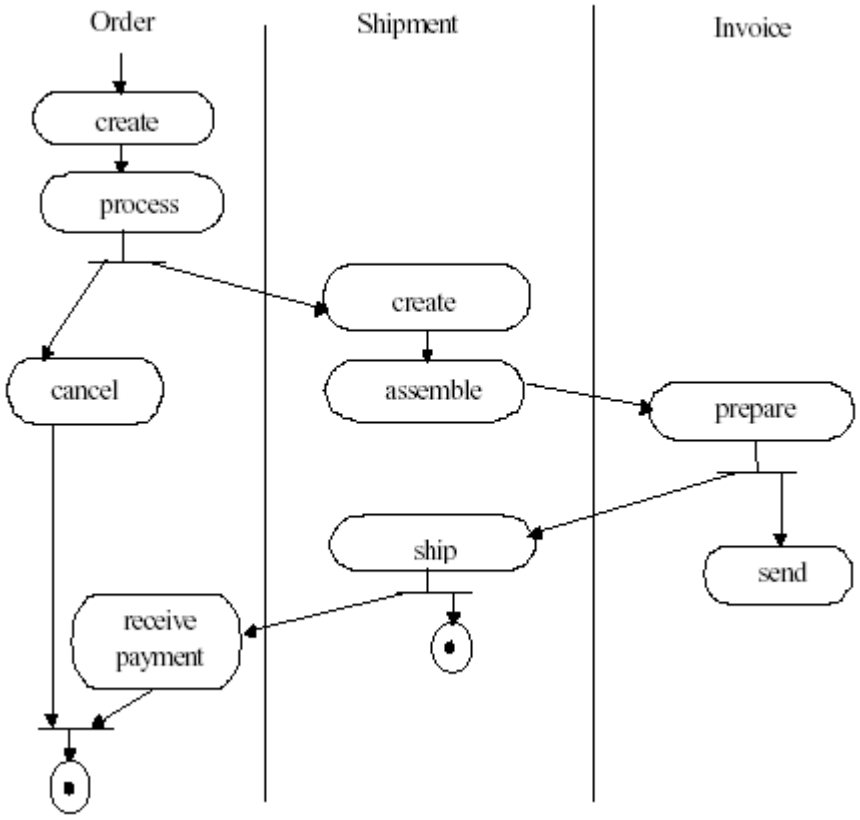


图9. 订单和配送的活动图

这个模式的应用提供了系统的方法来记录订单和它的完成情况，并且能够有助于优化完成周期和满意度。这个模式也应用于服务，这种情况下，配送表现为服务的提供。

类图中应用订单模式到一个通用的产品类，可以通过改变条目（LineItem）和产品类型（ProductType）与产品（Product）的关联来适配特定产品的应用。

并不是说由此模式描述的所有应用情况都是完全类似的：

- 获取和校验用户数据的过程在某些应用场合可能是十分冗长的，而在另一些应用场合可能根本不存在。
- 用户可能直接从商家取回订购的产品，这样模型中的配送过程就被跳过。
- 可能要求在产品送达前付款，也可以送达后结账，决定于顾客的状态和信用等级。

为了一般性，下面许多侧面没有在这个模式中描述：

- 产品的上下文和环境方面。
- 异常，例如：不良信用
- 如何跟踪产品的可用性
- 当产品无货时，如何排队等候。
- 如何处理不同的顾客，例如：个人顾客，集团顾客，优先顾客。
- 订单的更改
- 退货
- 支票付款的策略
- 配送的实现细节，例如：包装。
- 历史记录

所有这些方面需要由其它模式或者特殊的模型来完成。你可以扩展这些模式成为订单的模式语言。

5. 已知的应用

下面是一些应用实例：

- 某种设备的零售/服务提供商，例如：寻呼机，请求在零售店转卖一批设备。
- 顾客从饭店订餐
- 顾客从电子商务公司订购产品，例如：亚马逊（Amazon.com）
- 顾客为她的家订购了一个新的屋顶

文献中有许多这个模型的变种：

- Hay [Hay96] 给出订单和配送模式，但是他没有给出它们之间的关联，也没有考虑动态方面、属性、或操作。

- Fowler [Fow00] 利用订单作为他的UML书中的一个运行实例。我们的活动图就是基于此。
- Berkem [Ber99]描述了更加详细的活动图并把它们与用例连接起来。
- Ambler [Amb97]描述了订单的类图，他没有考虑订单。
- Schneider and Winters [Sch98]枚举了各种应用了订单模式的用例。
- Richter [Ric99]在股票交易环境考虑订单模式。
- K. Brown [Bro96], 对一个订单管理系统应用了设计模式，他没有讨论分析方面的内容。
- [Ope00]一个开放的应用程序组，为订单和配送定义了接口，使之能够在不同系统间协调工作。

6. 相关模式

预定和使用模式[Fer99]，本模式的补充，提供预定正在不断生产的产品的可能性。

股票管理模式[Fer00a]也是一个补充，完成了需求部件的订单并更改了库存内容。

如前所述，本文给出的模式可以作为制造业框架的一部分。类型对象模式[Bau00]在这里作为它的子模式。

顾客在过程可以担当不同的角色，角色对象模式可以用来描述这个过程。

关于货币的详细处理可以在模式 [Hay96] 和 [Fow97] 看到。

依赖请求模式[Hau97]讨论了在生产体系中一个订单可以触发另外的订单。

订单/配送模式可以作为复合模式 [Rie96] 的一个特例来研究。

致谢

感谢我们的带头人和Plop2000写作工作室，它们详细而有深刻见解的评论极大地提高了本文的质量。

参考文献

[Amb97] S. Ambler, “Taking a layered approach”, *Software Development*, July 1997, 68-70.

[Bau00] D. Baumer, D. Riehle, W. Siberski, and M. Wulf, “Role Object”, Chapter 2 in *Pattern Languages of Program Design 4,* Addison-Wesley 2000. http://st-www.cs.uiuc.edu/~plop/plop97/Workshops.html

[Ber99] B. Birkem, “Traceability management from business processes to Use Cases with UML”, *JOOP,* September 1999, 29-34 and 64.

[Bro96] K. Brown, “Experiencing patterns at the design level”. *Object Magazine*, January 1996, 40-48.

[Fer99] E. B. Fernandez and X. Yuan. “An analysis pattern for reservation and use of reusable entities”, *Pattern Languages of Programs Conference*, *PloP99*. http://st-www.cs.uiuc.edu/~plop/plop99

[Fer00] E.B. Fernandez and X. Yuan, “Semantic Analysis patterns”, *Procs. of 19th Int. Conf. on Conceptual Modeling, ER2000*, 183-195.

[Fer00a] E.B. Fernandez, “Stock Manager: An analysis pattern for inventories”, *Procs. of PLoP 2000*.

[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.

[Fow00] M. Fowler, *UML Distilled* (2nd Edition), Addison-Wesley, 2000.

[Hau97] R. Haugen, “Dependent Demand—A business pattern for balancing supply and demand”, *Procs. of Pattern Languages of Programs Conf*., PloP97, 14 http://st-www.cs.uiuc.edu/~plop/plop97/Workshops.html

[Hay96] D.Hay, *Data model patterns-- Conventions of thought*, Dorset House Publ., 1996.

[John98] R. Johnson and B. Woolf, "Type Object", Chapter 4 in *Pattern Languages of Program Design 3,* Addison-Wesley, 1998.

[Ope00] Open Applications Group, *Integration Scenarios,* http://www.opeanapplications.org/oagis

[Ric99] C. Richter, *Designing flexible object-oriented systems with UML*, Macmillan Tec. Publ., 1999.

[Rie96] D. Riehle, “Composite design patterns”, *Procs. of OOPSLA’97*, 218-228.

[Sch98] G. Schneider and J.P. Winters, *Applying Use Cases—A practical guide*, Addison-Wesley, 1998.

网上商店的模式

Eduardo B. Fernandez 著，Huang Yin 译

摘要

网上购物已经变得很普遍，许多网站都为此提供了方便的用户界面。为了支持不同种类的导航视图，网上商店需要良好的基础结构。目录模式和购买流程模式是网上商店基础结构的一部分：目录模式描述了如何组织网上商店的商品信息，购买流程模式描述了在网上购买商品所必需的步骤。我们还将展示在网上商店中如何结合应用这两种模式。

介绍

网上商店之所以变得如此普遍，是因为它们给予商家和客户更多的机会。在这一类应用中可以发现很多模式，我们将其分为三类：

- 用于导航视图的模式
- 用于构筑界面的模式
- 用于基础结构的模式

前两类模式在附录参考书中有介绍[Lya98， Lya99， Pal， Ros00]。好的基础结构可以提高服务器的响应速度，使导航视图和界面具有较好的可扩展性和灵活性。这种类型的模式应该是清晰、灵活、有效、可扩展的，并且独立于另两种类型。

在本文中，我们将介绍建立网上商店所必需的两种基础结构模式：目录模式和购买流程模式。目录模式用一种有效灵活的方式来组织商品信息，并为客户决策提供帮助机制。购买流程模式的目标是简化并提供一个方便有效的购买流程。

这些模式都是语义分析模式（Saps）的范例[Fer00a]，一个简单的语义单位可以应用于不同的情况。它们也是一种合成模式：它们的组件在其中有自己的值。这两种模式的结合可以用于定义一个网上商店的

框架。

目录模式

意图

目录模式组织网上商店所出售的商品信息

环境

网上商店是公司出售各种商品的地方。在服务器上还有用于提供流程结构、权限表、库存以及其他相关功能的应用。

问题

网上商店出售不同的商品，而这些商品有时是根本没有关联的，例如书和食物。一个重要的问题是：如何组织商品的信息，为客户提供一个友好的网上向导，从而提高网站对客户的吸引力？

约束

- 有大量不同种类的商品供销售
- 商品及其描述时常变化
- 如果查询和选择商品不方便，客户将不会再访问网站，基础结构应该支持这些不同的功能
- 应该将相关的商品介绍给客户来吸引其购买
- 目录的制作通常使用特别的样式，并且不能重复使用，这导致了时间和精力的重复和浪费。

解决方案

定义一个目录类作为收集商品的中心点。将不同关注点的特征，例如商品的细节描述，和相关的商品分成单独的类。利用观察者（Observer）模式来发现改变并关注客户的变化。

类图

目录是一类商品的集合。每个商品至少归属于某个目录。**商品**类定义了一种供销售的商品，购买者购买的是一种商品，而不是商品的个体。这个类包括了每个商品的基本属性，特别是代表某种特性的状态属

性，例如，一个新的商品。导航视图可能将特殊的商品放在屏幕上的显著位置，以便与一般商品相区分，从而让客户了解[Lya98]。

ProductInfo 类提供了更多的商品细节信息。其中可能包括同一种商品的不同类别、不同品牌的比较，或是提供一个最好的性价比。例如对电视机，可能就要提供几个不同品牌的比较信息。作为旅游中介，可能就要给客户提供附加的选择，例如让客户能够选择较近的机场，这样可以使飞机票更为便宜。**SimilarProduct** 类则提供相似商品的链接。

通过电子邮件的方式将商品（信息）的修改通知客户，让他们能了解到新的或是感兴趣的商品（信息）。**ProductObserve** 类用于检测变化并通知客户。注意这些类都是观察者（Observer）模式的一种类型。**Notification** 类保证将修改的记录通知客户。

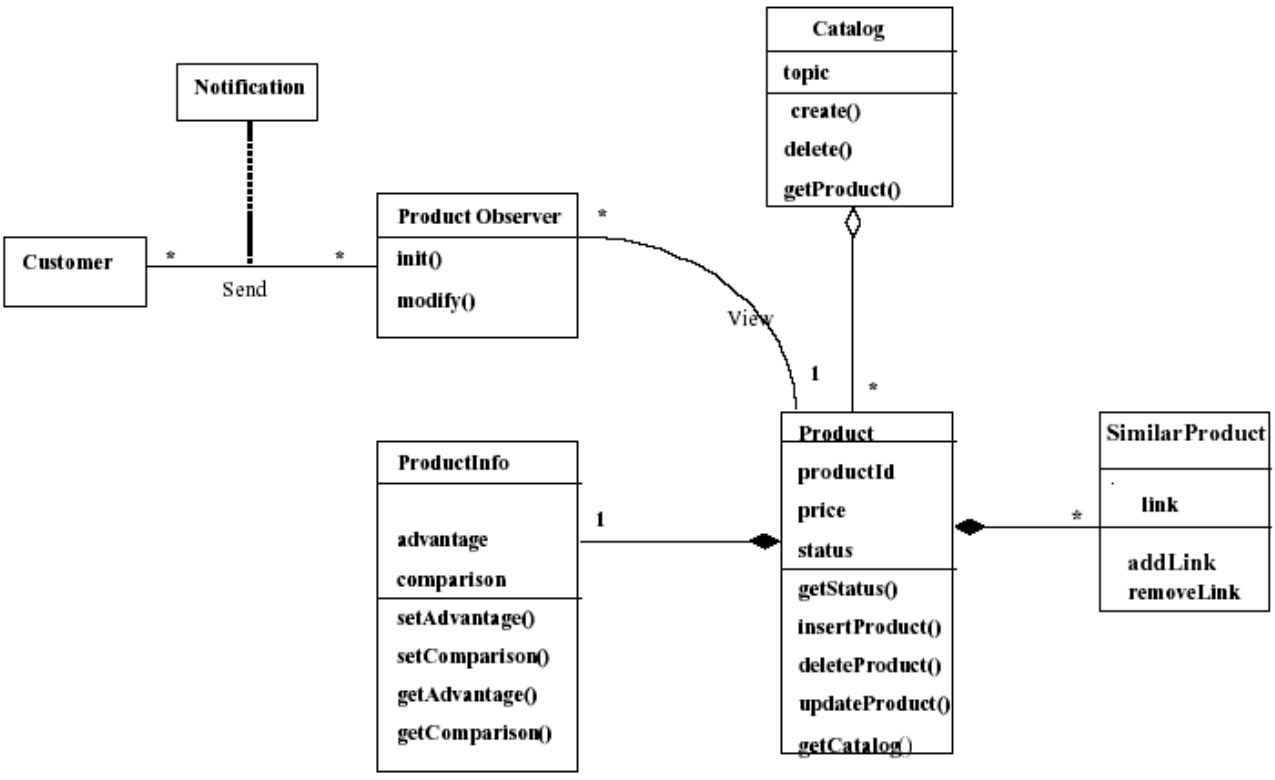


图 1.目录模式的类图

动态描述

图 2 中的序列图显示了当商品信息修改时触发的协同操作。这与标准的观察者（Observer）模式序列相似。

结论

这种模式有以下好处：

- 它提供了一种描述商品方便、有用的基础结构。导航类可以加入其中，利用有吸引力的方式显示商品信息。
- 它可以复用。这种模式适用于各种不同的网上商店，不管是大商场还是个人商店。
- 它可以容易地和其他模式结合使用，例如个人化模式[Pal，Ros01]，我们下面要介绍的购买流程模式，或是库存模式[Fer00b]。

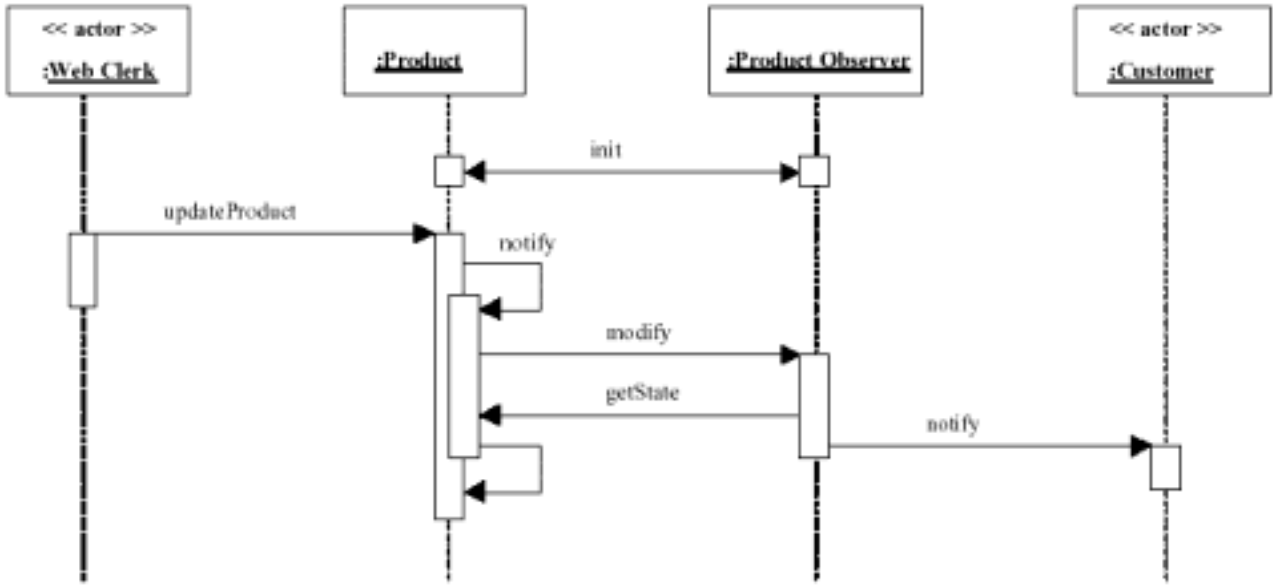


图 2.修改商品（信息）的序列图

其他结论包括：虽然基本模式涉及商品类型（的概念），但某些站点只是出售一些个别的商品，例如旧车网站、拍卖网站。既然 Product 类作为个体商品的集合，那么我们需要一个附加类来指出商品的类型。

已知的用途

该模式可以适用于很多应用，例如：

- 网上书店。每一个网上书店都利用目录来组织书籍，如计算机、医学、历史和文学。商品就是书。商品信息包括每一本书的评论及描述。状态属性指出特殊的交易情况，新书或是畅销书。如果一些客户对某一类的数据感兴趣，不管什么时候，只要新到这类书或是特殊的交易情况

有效，系统都将用电子邮件通知客户。

- 音乐商店，鞋店，酒店…..

很多的 Web 应用服务，例如 IBM 的 Websphere 的商务套餐都体现了目录模式。[ibm]

相关模式

目录模式包括容器/上下文模式[Coa97]。股票经纪通过跟踪大量股票中的商品来实现这个模式[Fer00b]。其他经常和目录模式一起使用的模式是搜索模式[Lya99]。

购买流程模式

意图

为网上商店选择及购买商品的流程建模。

关联

网上商店包括很多商品。客户可以在同一次会话中选择和购买不同的商品。客户寻找自己感兴趣的商品，选择他想买的东西放入购买列表中。他可以随时检查和更改他所购买的商品。有一些顾客还可以在网站上登记并可以得到不同的待遇。网上商店还包括目录，成员表，运输以及其他的应用程序。

问题

购买流程必须有定义良好的步骤，以便告诉客户他现在正处于流程中的那一步。现在问题是：怎样用精确的方式来描述购买流程？

约束

- 客户有可能在购买流程的步骤中感到很迷惑。我们要尽可能地让购买流程对于客户而言清晰而且方便。很明显，这要依靠那些显示给客户的特殊网页。在这里，我们所考虑的是使这项工作更为方便的后台结构。
- 我们需要向客户显示他现在正处于购买流程中的什么位置。
- 因为新的商品或是商业模式而可能需要改变流程的步骤。这可能暗示新的实体或新的步骤。

- 客户希望有不同的支付方式。
- 有些客户必须要区别对待。
- 如果购买（流程）的结构不安全，那么客户将不再光顾。

解决方案

用 " 购物车 " 来实现购买流程，是一种最为普遍的方法。一个客户可能会有几个购物车，每个购物车中都有客户所选择的商品。当客户决定购买购物车中的商品时，产生订单和收据。

类图

ShoppingProcess 类是这个完整流程的单一入口。ShoppingCart 类包括了客户所选择商品的信息，CartItem 类定义了客户购买的商品和数量。客户可以查询、删除其购物车中的商品。Customer 类定义了购物车所对应的客户，这个类中还提供给客户修改个人信息的操作。系统提供了集中支付的方式，如信用卡支付或者电子支票。一旦客户决定对购物车中的商品结账时，产生订单和收据。

※※※

高焕堂

高焕堂，台湾杰出的资深 OO 专家，1995 年创办"物件导向杂志"和 MISOO 物件教室，在台湾普及 OO 方法，育人无数，并著（译）有大量 OO 书籍。交流重点：系统分析，N-tier 架构，UML。

答疑板

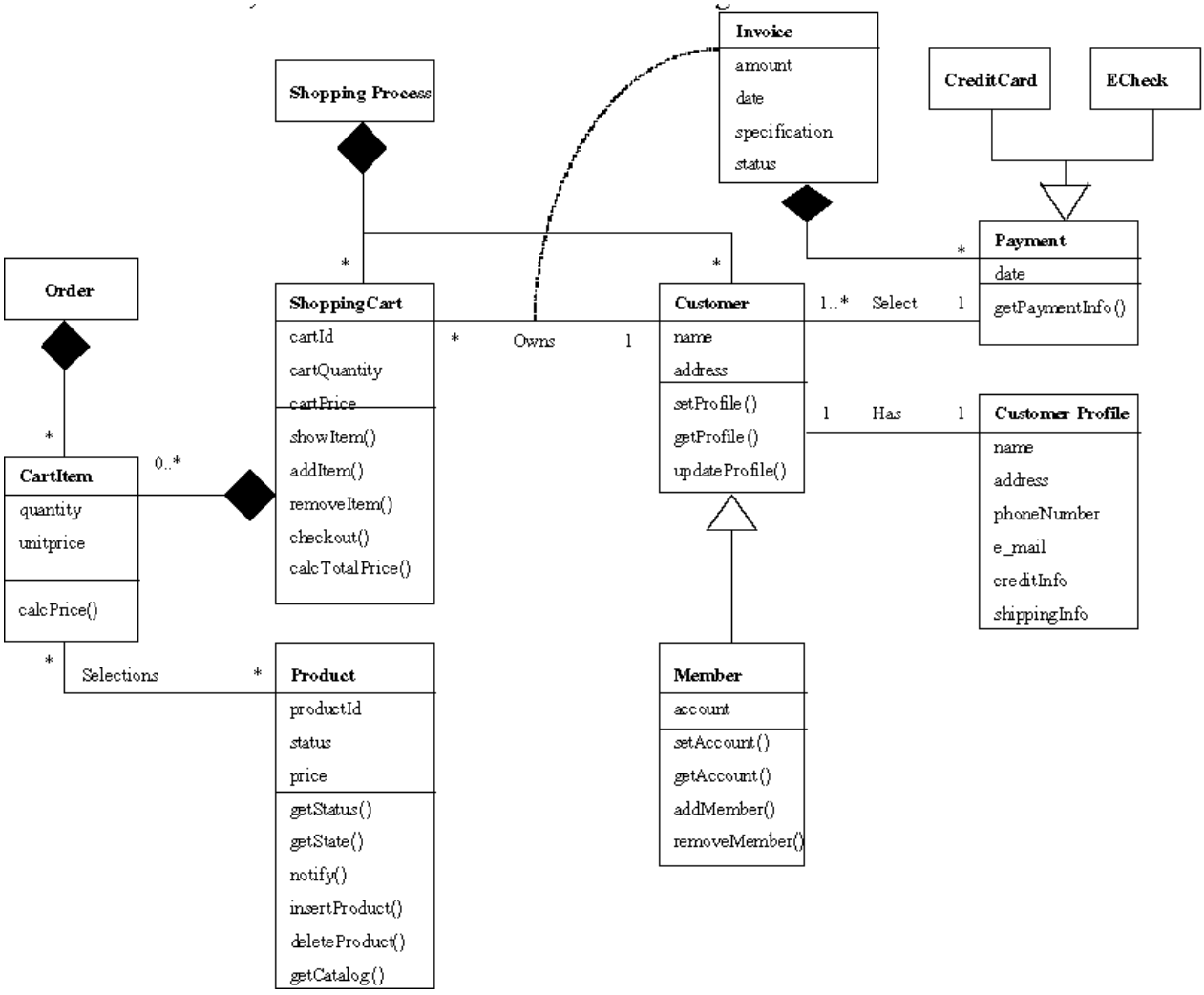


图 3.购买流程的类图

动态描述

这种模式包括了以下交互（图 4）：

- *选择*：当客户对同一种商品进行选择时，产生一个新的购物车项目（cartItem）对象，并将其加入到购物车中。

- 结帐：当客户结帐时，计算所选择商品的金额，用户账单和送货信息被重置，随之产生收据。

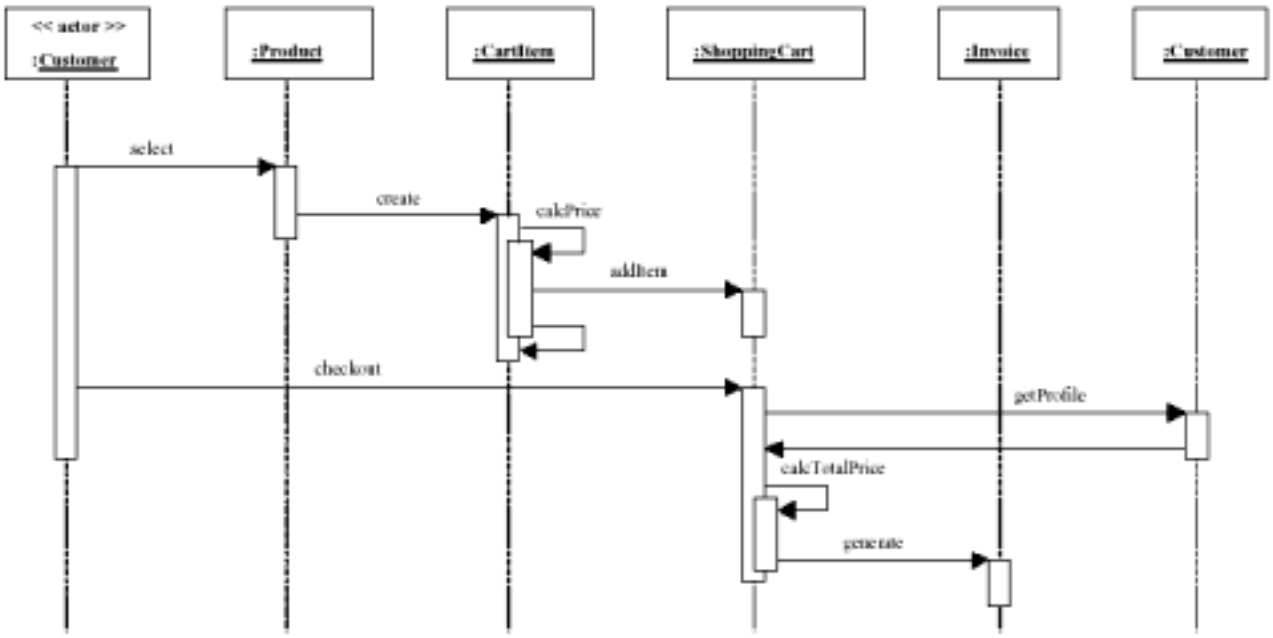


图 4.购买和支付商品的序列图

结论

使用该模式的好处如下：

- 该模式描述了一种抽象的购物流程。它被应用于不同的网上商店，从卖鞋的到卖软件的。
- 该模式为因特网上建立商店提供了公用的元素。
- 该模式可以简化因特网上的购物流程，部分或所有的步骤在向导中显示给用户。
- 对于会员用户可以不需要重新进入。
- 在总体结构中容易加入授权（功能）。例如，仅有流程管理员才有权变更流程。

其他需要注意的地方：

- 该模式并没有提供出错处理的机制，例如对错误的信用卡、错误的付款地址或送货地址的处理等，这些必须要包含在一个特殊的模块中。
- 这是一个通用的网上商店模型，当用于一些特殊的网上商店的典型时，可能需要修订。
- 对有效性而言，这个模式需要低级的基础结构来支持，例如数据库系统。

- 在模式中定义的安全约束必须被低级别的机制所执行，例如数据库授权、文件访问权限、加密和其他一些机制。

已知用途

大多数的网上商店使用购物车的概念，在基础结构上和该模式相似。相似模式的特殊模型和代码可参见附录[Bar96]， [Bol00]和[Kob01]。

相关模式

该模式可以用于支持网络用户的两种模式：

- 动态配置（Dynamic Configuration）模式[Lya98]，该模式帮助用户从不同选项中进行选择并验证其选择。

- 显式流程（Explicit process）模式[Ros00]，该模式帮助用户理解购买流程。

导航模式，如新闻[Lya98]和广告[Ross00]，可以利用这两种模式作为其基础结构的一部分。

订单和送货模式[Fer00]，可以补充本模式中的运送细节。付款的细节处理参见 [Fow97]以及[Hay96]。购物车，客户相关类以及付款类都是各司其职的基本模式。这里所介绍的两种模式的抽象版本的结合是购物框架的基础。

感谢

我们要感谢我们的领导 Gustavo Rossi 有见地的建议，他的建议使这篇文章增色不少。

参考文献

[Bar96] C. Baron and B. Weil, “Implementing a web shopping cart”, *Dr. Dobbs Journal*, September 1996, 64-69 and 83-85.

[Bol00] G. Bollinger and B. Natarajan, “Build an e-commerce shopping cart”, *Java Pro*, June 2000, 38-50.

[Coa97] P.Coad, "*Object models: Strategies, patterns, and applications*" (2nd Edition), Yourdon Press, 1997.

[Con99] J. Conallen, *Building Web Applications with UML*, Addison-Wesley,1999.

[Fer99] E.B.Fernandez and X.Yuan, “An analysis pattern for reservation and use of entities”, *Procs. of Pattern Languages of*

Programs Conf. (PLoP'99), http://jerry.cs.uiuc.edu/~plop/plop99

[Fer99a] E.B.Fernandez, "Coordination of security levels for Internet architectures", *Procs. 10th Intl. Workshop on Database and Expert Systems Applications*, 1999, 837- 841.9

[Fer00] E.B.Fernandez, X.Yuan, and S.Brey, "An analysis pattern for the order and shipment of a product", Procs.of Pattern Languages of Programs Conf. (PLoP'2000), http://jerry.cs.uiuc.edu/~plop/plop2k

[Fer00a] E.B. Fernandez and X. Yuan, "Semantic Analysis patterns", *Procs. Of 19th Int. Conf. on Conceptual Modeling, ER2000*, 183-195.

[Fer00b] E.B. Fernandez, "Stock Manager: An analysis pattern for inventories", *Procs. Of PLoP 2000*, http://jerry.cs.uiuc.edu/~plop/plop2k

[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997

[Hay96] D.Hay, *Data model patterns-- Conventions of thought* , Dorset House Publ., 1996.

[ibm] IBM Corp., Web Sphere Commerce Suite, http://www-4.ibm.com/software/webservers/commerce

[Kob01] C. Kobryn, *Modeling components, patterns, and frameworks with UML*, Notes for tutorial, *Software Development 2001 West*, April 2001.

[Lya98] F. Lyardet, and G. Rossi, "Patterns for dynamic websites", *Procs. PloP'98*, http://jerry.cs.uiuc.edu/~plop/

[Lya99] F. Lyardet, G. Rossi, and D. Schwabe: "Patterns for adding search capabilities to web information systems", *Procedings of EuroPloP'99,* http://www.argo.be/europlop/index.html

[Pal] M. Palmer, "A Personalization design pattern for dynamic websites", http://objectdesign.com

[Ros00] G. Rossi , F. Lyardet, and D. Schwabe, "Patterns for e-commerce applications", *Procs. EuroPLoP'2000*, http://www.coldewey.com/europlop2000/

[Ros01] G. Rossi, D. Schwabe, J. Danculovic, and L. Miaton, "Patterns for personalized web applications", *Procs. EuroPLoP'2001*, http://www.hillside.net/patterns/EuroPLoP/

[Sch98] D. Schwabe, and G.Rossi: "An object-oriented approach to web-based application design", *Theory and Practice of Object Systems (TAPOS*), October 1998.

预订和使用可重用实体的分析模式

Eduardo B. Fernandez，Xiaohong Yuan 著，Shane 译

摘要 本分析模式描述怎样预订然后使用可重复使用的实体。我们通过用例来表达模式的需求，使用类模型、状态图和序列图对模式进行具体的描述。本模式对应一个最小的语义单位。

1．介绍

我们在此提出一个分析模式，描述怎样预订然后使用可重复使用的实体。该模式归于我们称为语义分析模式(semantic analysis patterns)的范畴[Fer98]，因为它们强调的是应用模型的语义侧重面，而不是为了增加设计的灵活性。

我们认为这类模式对于以需求为出发点开始建模过程的设计方法是有用的。例如，识别需求中的一些模式产生一个初步的模型，这个模型可用作后续设计的指南。这些模式也能被用来开发框架和组件。

本文的分析模式强调预订的基本侧重面而不处理它的扩展、异常处理和各种变体；可以在以后添加这些内容来定义预订的模式语言。

2．问题

一个客户（个人或机构）需要预定一个可重用的实体（如旅馆房间、车辆、演出座位）供他随后使用。这些实体数目是有限的，能够按相同种类或类型进行分组。

所有这些情形隐含着这样一个请求：在某个具体日期使用某类实体。如果有一个请求的实体可用，它就被请求它的客户预订。实体在客户使用的时候被分配，同时建立一条使用记录描述该实体正在使用。

被预订的实体是可以重复使用的；那就是说，它们不会被客户永久拥有，客户只在一定的时间段对它们有使用权。这一点意味着通过让用户在使用实体后负责将它们完整归还，使用记录同时具备（明确的或隐含的）合同的功效。

图 1 显示一个客户预订旅馆房间的类图。在 Customer 类和 Room 类之间的关联关系描述一个客户的每个预订情况。Reservation 关联类描述正常情况下一个预订中应该包含的信息。Availability 类描述房间的结构和它们的占用情况。

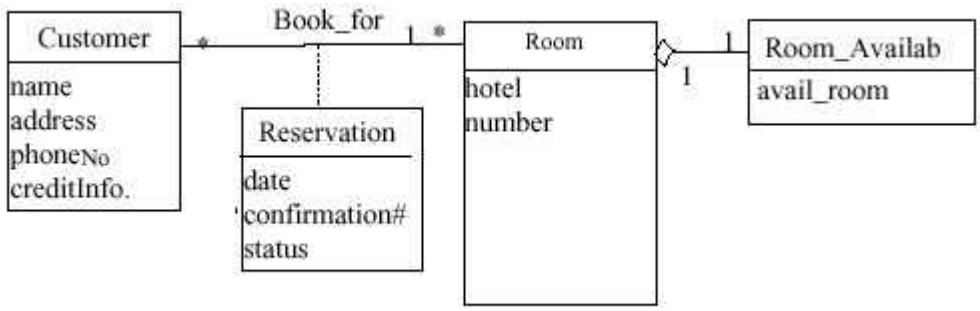


图 1. 预订一个旅馆房间

3. 约束

本模式使用以下约束：

- 正规的分析目标应用于解决方案。一个分析模型必须是对需求的真实可靠的表示，并且必须不能包含任何实现细节。
- 基本语义。模式必须描述一个基本的语义单元。这意味着模式应该是简单的，而且能够描述各种变化的情况。这种方法已在[Ngu98]使用过，在那里被称为一个“最小限度的”(“minimal”)的应用程序。
- 现实文档的模型表示。预订和使用记录通常同时使用文档和它们的软件形式进行记录。

模式语言的约束也应包括方便有效的客户服务（如排队工具）和可靠性（如异常情况的防备措施）

4. 解决方案

4.1 需求

解决方案相应于下列用例要实现的功能。

- 建立预订。主角是客户（个人或机构）。客户请求预订一个一定类型的实体，给出开始使用的日期和使用的时间。如果有适合客户要求的实体，预订成立，该实体被预订。上述相关信息记录在一个预订文档中。

- 使用预定的实体。主角是建立预订的同一客户。在实体被使用前需要建立一个使用记录。使用结束后，实体被归还又可供其它客户预订使用。

- 修改预订。对已建立的预订进行修改。这意味着确定预订的有效性和取消老的预定。

- 取消预订。取消已经建立的预订。一些机构政策可用来定义取消预订的效果。(译者注：原文为“Some institution policies may apply to define the consequences of the cancellation.”)

注意：要求修改预订的功能在建立和取消预订的用例中可以使用。同时，这些用例的异常处理流程未被考虑。

4.2 类模型

图 2 是这些用例实现的一个分析类图。该图对可被预订的实体（类 Entity_type）和使用时被分配的单个实体（类 Entity）进行了区分。UseRecord 类描述实体的使用情况。Availability 类描述检查和跟踪房间占用情况的一般知识。Reservation 类包括正常情况下保存在预订中的信息。Reservation 类和 UseRecord 类之间的关联揭示了这样一个事实：使用记录是建立在预订信息基础上的。

4.3 动态行为

图 3 和图 4 是关于一些最重要的类的状态图。例如，一个预订能够处于已建立（Created）和已确认（Confirmed）状态。图 5 显示一个对实体建立预订和随后使用的序列图。

5. 效果

本模式有以下效果：

- 它能被用于旅馆房间、会议室、飞机和演出的座位、书和录像带等实体的预订。这各种应用表明本模式包含预订问题的基本方面。它也是一个简单的模式，很容易理解。

- 它表示了用例的各个方面，没有包括实现方面的细节。

- 类模型明确包括了预定和使用记录类（Reservation 类和 UseRecord 类），它们是在上述预订系统中使用的基本文档。这简化了跟踪预订历史和正在使用的相关实体的工作。使用记录（UseRecord）作为有效合同的角色也很明确。

然而，本模式对上述所有情形的描述并非完全相同。

- 客户预订飞机和演出的座位时有真实的票据，而在预订车辆或房间时却没有（尽管客户

可以获得一封确认信）。票据代表一个合同。

- 当车辆被客户拿走使用时，车辆的预订明确地转变为一个合同；而旅馆房间的预订是当客户签名入住时隐含地转变为合同。

- 车辆和房间的预订在使用时对具体实体进行分配，而演出和飞机座位的预订是在预订时就对实体进行了分配。

- 有些实体，如车辆和房间，预订一段时间表示为几天，而另外一些实体，如飞机座位，表示为对一个具体的事件进行预订。

- 一个合同不需要预订也可以建立，一个预订也可以不转变成一个使用记录。

此外，还有很多方面的内容没有在本模式中表示：

- 关于实体上下文和环境的描述。例如，一张飞机票应该涉及飞机、机场和起落次数（the number of stops）等。

- 怎样跟踪实体可用性。

- 当实体不可用时怎样对请求进行排队。

- 怎样处理优先客户

- 帐目处理

- 历史

最后，异常流程处理也已被忽略，如无演出和超额预订等情况。

可以考虑将所有这些方面定义为一个模式语言或一个模式家族，这是我们将来要进行的工作。

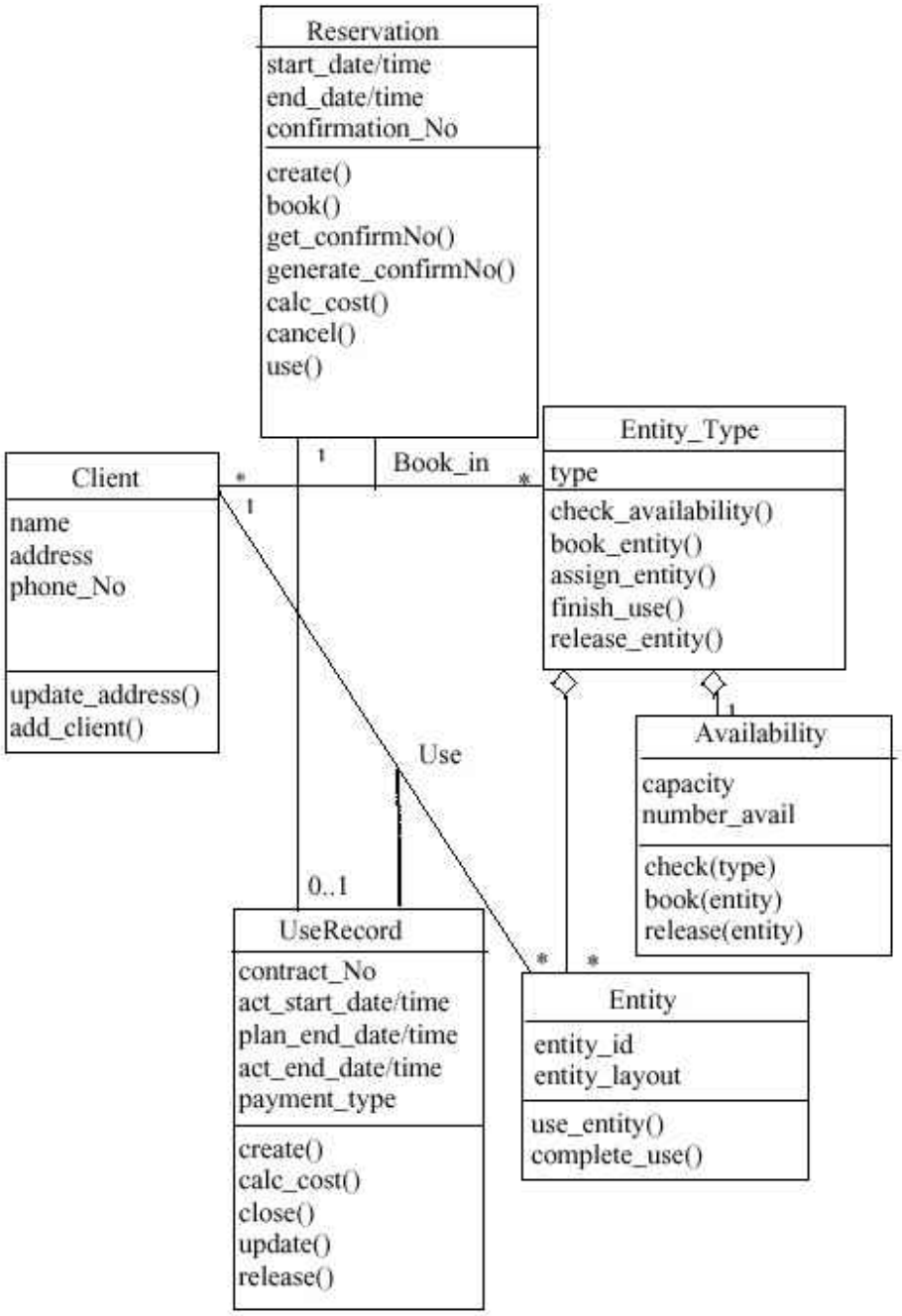


图 2. 预订/使用模式类图

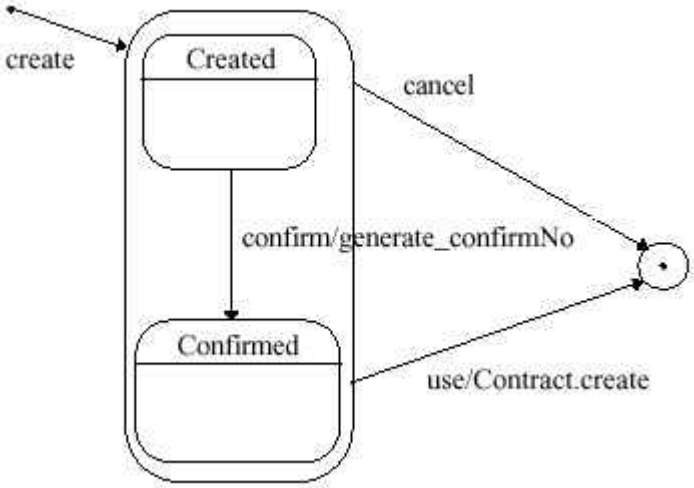


图 3. 类 Reservation 的状态图

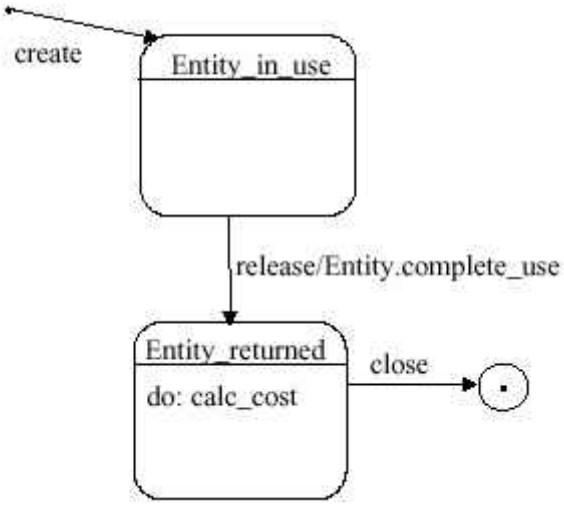


图 4. 类 UseRecord 的状态图

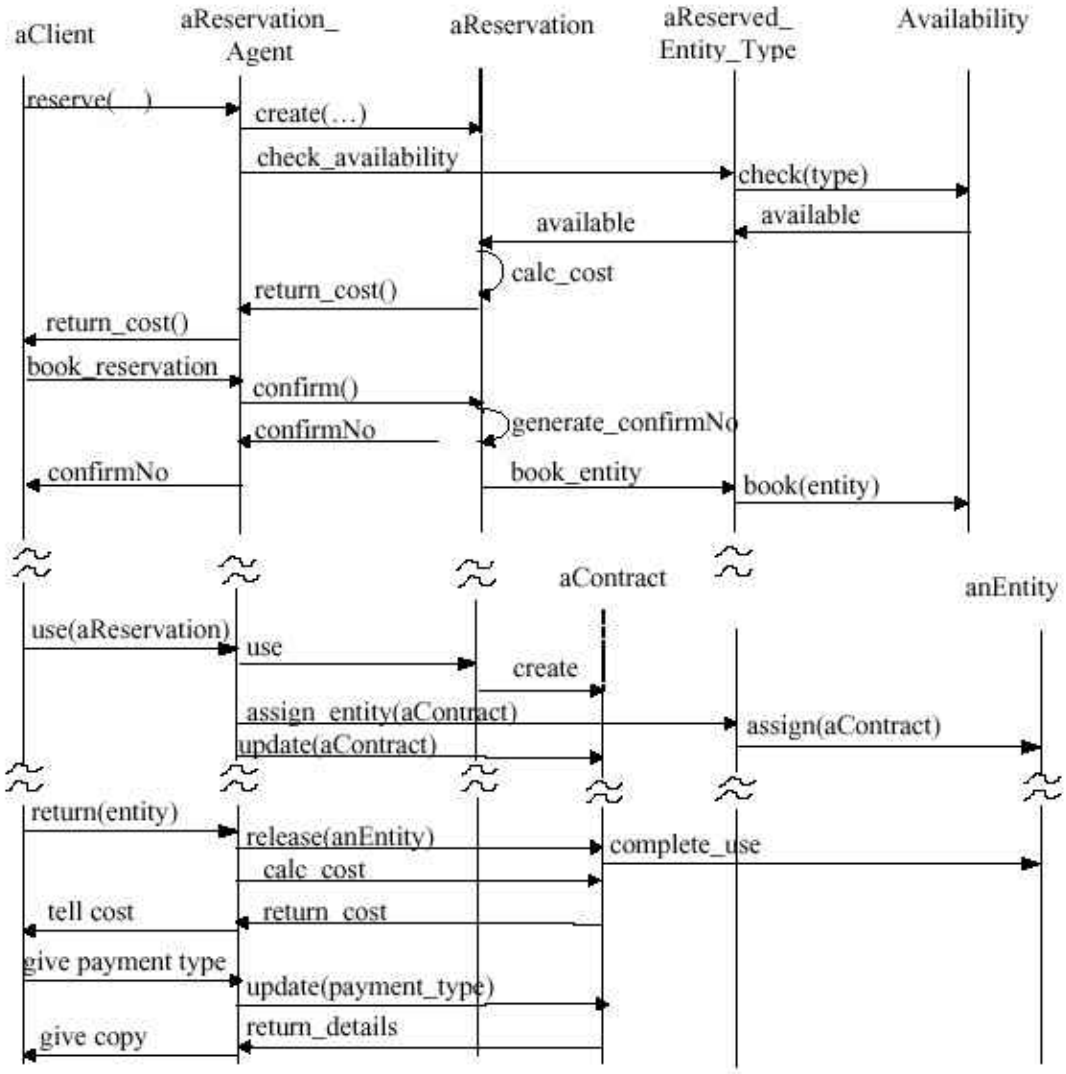


图 5. 预定和使用的序列图

6. 已知应用

根据下列文献和我们的经验，我们发现本模式已被用于以下场合：

- 客户预订演出或飞机的座位[Bak97]
- 客户预订旅馆房间[Fer98a]
- 客户预订车辆[SA98]
- 客户向图书馆预订图书[Fer98a]
- 客户向音像店预订录像带[Fer98a]，[Joh96]

7．相关模式

NATURE 项目[Nat98]中有一些模式，如资源分配和资源使用，从不同观点讲述了某些与本模式相通的内容。Coad[Coa97]提出过一个与本模式在某些方面相通的资源请求模式。Fowler[Fow97]讨论了计划和资源分配。制造业零部件的预订和随后分配方式也与本模式有某些方面的共同点[Fer97]。在所有这些案例中，实体没有被重复使用，因此不需要使用记录。[Bra98]中讨论了书籍和音像制品预订的某些内容，但没有考虑被预订实体的使用情况。

我们的模式作为 Type Object 模式的一个组件使用[Joh96]。事实上，正如[Fer98]中描述的那样，这些复杂的模式通常包括一些本身就具有实际意义的子模式(subpattern)。

参考文献

[Bak97] S. Baker, CORBA: Distributed objects using Orbix, Addison-Wesley 1997.

[Bra98] R.T.V.Braga, F.S.R.Germano, and P.C.Masiero, "A confederation of patterns for resource management", Procs. of PLOP'98, http://jerry.cs.uiuc.edu/~plop/plop98

[Coa97] P. Coad, “Object models: Strategies, patterns, and applications” (2nd Edition), Yourdon Press, 1997.

[Fer97] E. B. Fernandez and Z. W. Peng, “ An object-oriented model for manufacturing inventory control systems”, Tech. Report TR-CSE-97-30, Dept. of Computer Science and Eng., Florida Atlantic University, April 1997.

[Fer98] E. B. Fernandez, “Building systems using analysis patterns”, Procs. 3rd Int. Soft. Architecture Workshop (ISAW3), Orlando, FL, November 1998. 37-40.

[Fer98a] E. B. Fernandez, Class Notes for COP5330, Introduction to Objected-Oriented Software Design, Dept. of Computer Science, Florida Atlantic University, Boca Raton, Florida.

[Fow97] M. Fowler, Analysis patterns – Reusable object models, Addison-Wesley, 1997.

[Gam95] E. Gamma et al. Design Patterns – Elements of reusable object-oriented software, Addison-Wesley 1995.

[Joh98] R. Johnson and B. Woolf, "Type Object", Chapter 4 in Pattern Languages of Program Design 3,

Addison-Wesley, 1998.

[Nat98] NATURE project, http://www.city.ac.uk/~az533/main.html

[Ngu98] K. Nguyen and T. Dillon, "An alternative solution to the observation pattern problem", in PLOP98. http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions

[SA98] Software Architects. Exercise in course SA4012, Testing Object-Oriented Systems, 1998.

仓库管理器：一个库存的分析模式

Eduardo B. Fernandez 著，邓克 译

摘要 库存可以让我们知道一个机构有什么，如：零件，成品，家具，机器等。一个好的库存系统对任何一个现代商务或制造系统来说，都是必要的。本文中，我们列举了一个针对库存系统的分析模式，这个库存系统跟踪仓库中物品的数量和位置，并且能够根据制造或生产的不同阶段(从零部件采购到产品发运)，动态更新有关物品的数量。这是一个通用的模型，是根据对一个实际运行库存系统的抽象而定义出来的，并且可以通过扩展，实现一个有更详细需求的或相类似的应用。这是一个组装模式，由两个原子模式组成。

1 介绍

现代制造系统中，制造过程中所涉及信息的管理已经成为降低产品成本，提高产品质量的一个关键因素。很多公司和机构在这个领域投入了大量的资源，制造资源计划系统（MRP）已经变得重要了[Salv92]。库存是 MRP 系统中最重要的一部分，用来跟踪目标对象的数量和位置。

这里，我们开发了一个库存模式－仓库管理器，它满足对可重用性和可扩展性所期望的要求。这个模型不仅考虑了系统的静态视图，还考虑了它的动态方面。因为一个库存控制系统不能被完全地模型化(除非顾及到整个制造系统中几个其他方面的问题)，所以模型中也包含了制造系统的一些功能性部分（非库存方面的）的作用。这个模型可以作为开发一个完整的制造系统模型的起点或是作为一个语义分析模式（Semantic Analysis Pattern）[Fer00a]的实例,这种实例实际上是一个领域范围内可以使用的最小应用。这个模式也是一个组装模式[Rie96]，我们识别出了两个原子模式：基本库存(Basic Inventory)和物品分布(Item Distribution)。

2 问题

商业活动、制造车间、图书馆怎样跟踪各种不同种类的物品和它们的位置？

3 环境

所有机构，不管是大型的，还是小型的，都使用零部件来构建产品或供应其他商家。他们需要能够跟踪物品的数量。在制造车间里，这些物品是零部件和产品；在图书馆里，它们是书，磁带，或者是档案；在服装店里，它们是服装及附属物，等等。这些机构也需要能够找到所需的物品。

4 约束

- 在一个公司里，物品可能是正在制造或已经完成产品的原材料。货物有损失或遭到破坏的可能。机构必须能够跟踪仓库中物品的实际数目。

- 其他功能单元可以改变物品的数量：例如，物品无论何时、何地迁移或使用时，应该更新其相应的库存数量。

- 解决方案必须描述一个基础的语义单元，这意味者它必须足够简单，以便应用到各种环境中。这是可重用性的基础。

- 解决方案必须包括现实的文档解释。

5 解决方案

5.1 需求

一个库存系统的基本功能或用例（Use Case）可以总结如下：

- 把不同类型的货物分开，如：物料或零部件与成品。

- 跟踪仓库中的每个物品。可能需要几种类型的数量。手头（onHand）数量指示了物品的现存数量。定购（onOrder）数量指示了将来有多少物品可以入库。保留（reserved）数量指示了为订单保留的数量，它暗示了一个员工以后可以使用的数量（可获得数量）是手头数量与保留数量的差额。在更复杂的系统中，可能还会用到其他类型的数量。

◆ 跟踪物品的位置。库存应该记录物品在特定位置上的分布。

由这些功能的本质所决定，几乎所有其他子系统都会对库存中记录的数量有影响。一些对库存系统有直接影响并更新库存记录数据的功能是：采购、接收、物料的分发、审计、报废、发运、制造。

5.2 原子模式

我们从定义一个跟踪物品的库存模式—基本库存模式开始。

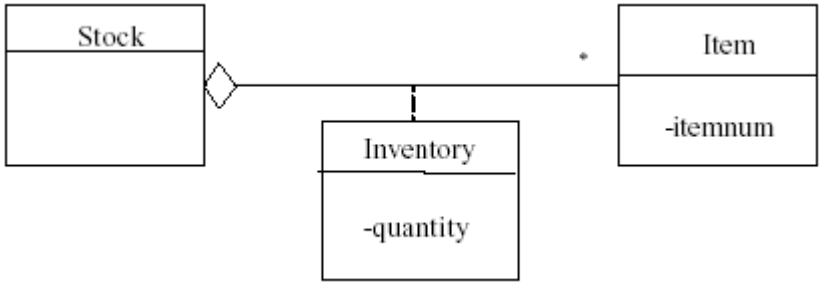


图 1.基本库存模式的类图

库存系统的基本模式如图 1 所示。物品是任何我们想知道它的存在位置和数量的东西。每一个物品属于一个唯一的类型，这个类型提供了一个标识，如：物品号（物品编码，型号）。库存(Inventory)类持有每个物品我们所关注的各种数量。这可以被认为是一个原子模式，如语义分析模式（SAP），它既可以单独存在，又可以作为一个更大模型的一部分。它的动态方面在组装模式中显示。

图 2 显示了物品分布模式的类模型。这是一个描述一系列物品的位置以及在那些位置上的分布的模式。它是另一个原子模式，将和基本库存模式一起组装成仓库管理器模式。为了指示分布状态，我们需要把库存数量分解成多个位置数量。（一个库存物品通常存放在几个不同的位置上）。这要求在库存(Inventory)类和位置(Location)类之间定义多—多的关联关系，并且使用一个关联类－分布(Distribution)来指示物品在位置上的分布。

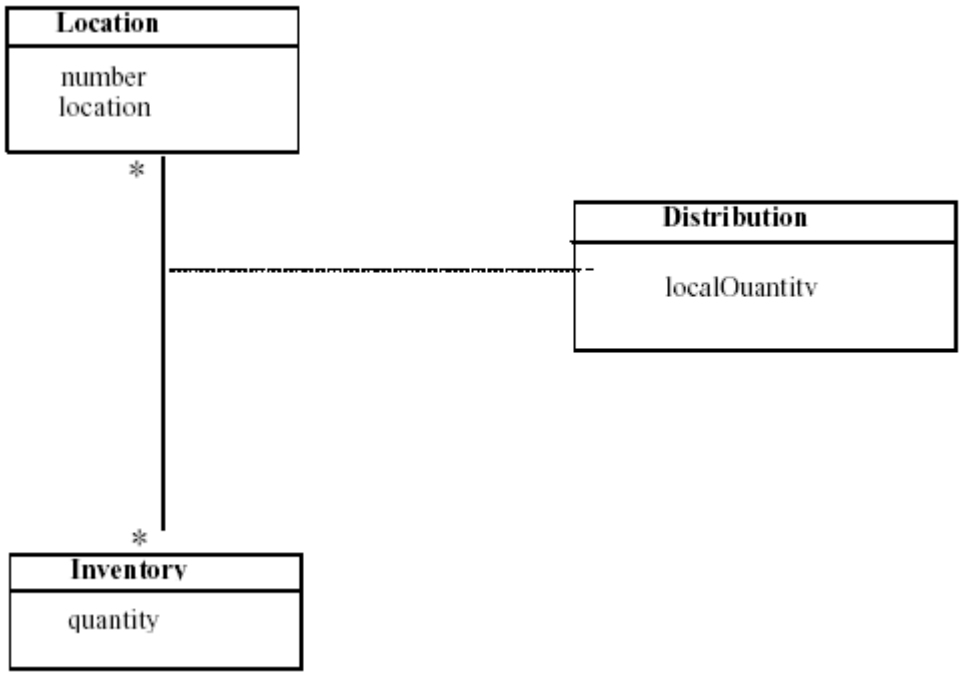


图 2 物品分布模式的类图

5.3 货物管理器的类模型

如下图所示：

图 3 显示了满足需求 5.1 的库存模型。仓库(Stock)和零部件/产品(Component/Product)通过一个聚合关联发生关系。[1] 库存(Inventory)类中的数量(quantities)属性是仓库(Stock)与零部件/产品(Component/Product)类的结合点，不同的链接，有不同的值。这个模型允许设计者定义不同类型的仓库，作为单独的物品集合；例如，零部件仓库，产品仓库。库存的不同类型可以归纳为一个库存(Inventory)类。

[1] 这是一个松散的聚合类型，两个类都可以独立存在。[Booc99]

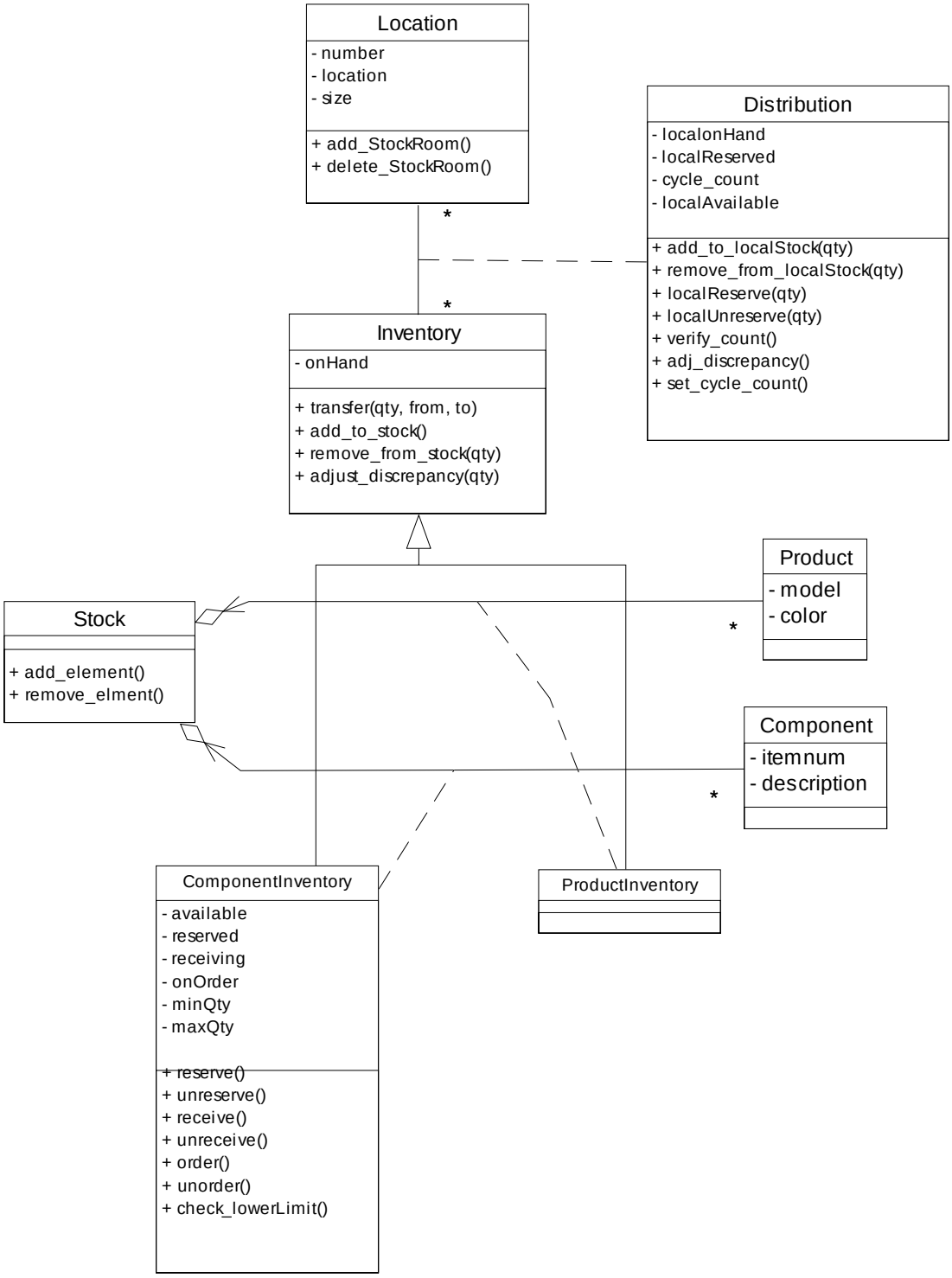


图 3.仓库管理器分析模式的类图

【译者注：原图中关联类 ComponentInventory 和 ProductInventory 的关联目标搞反了。这幅图是译者自己用 Rational Rose 2000 作的。另外，原图中的'/'符号，在 UML 中没有，应该是作者的笔误。】

物品被划分成两个不同的类型：成品和用于制造产品的零部件。其他的划分方法也是可能的。产品(Product)类中的“型号(model)”属性，用来作为一个唯一的标识符，其他属性描述了顾客选择时可能用到的特征，如颜色等。零部件(Component)类具有物品序号，描述，类型等属性，物品序号用来做唯一的标识符。产品(Product)和零部件(Component)通常是多-多的聚合关系（一个型号的产品使用几种类型的零部件，一种类型的零部件可以被用在几种型号的产品中），不过这与库存模型无关，因此在图中没有标明。

物品(Item)类被划分成产品(Product) 和零部件(Component)子类的原因是：在这两个实体的管理中，存在着很多不同。例如，产品是由一些零部件制成的，在制造过程中，库存系统需要跟踪零部件数量的变化。换句话说，零部件库存比产品库存更复杂。另一方面，两者又在一些方面有相同之处，例如，都需要跟踪手头（onHand）数量（合计的总数量）和实际的存放位置。这里可以应用泛化关系，定义库存(Inventory)类为一个父类，含有共同的特征，而产品库存(ProductInventory)和零部件库存(ComponentInventory)类作为子类，含有每一种不同库存类型的独特特征。图 3 的模型中也包含了从动态分析中引申出的操作，这将在下一部分描述。

5.4 动态分析

图 3 中的类模型提供了库存系统的静态视图。这是不够的，我们也需要动态模型来显示库存怎样随时间而变化。在动态模型中，我们定义了为完成通用的操作而应该具有的一般特征。具体的库存系统的其他特征可以从该模型的基本配置中派生。图 4 是动态模型的序列图。

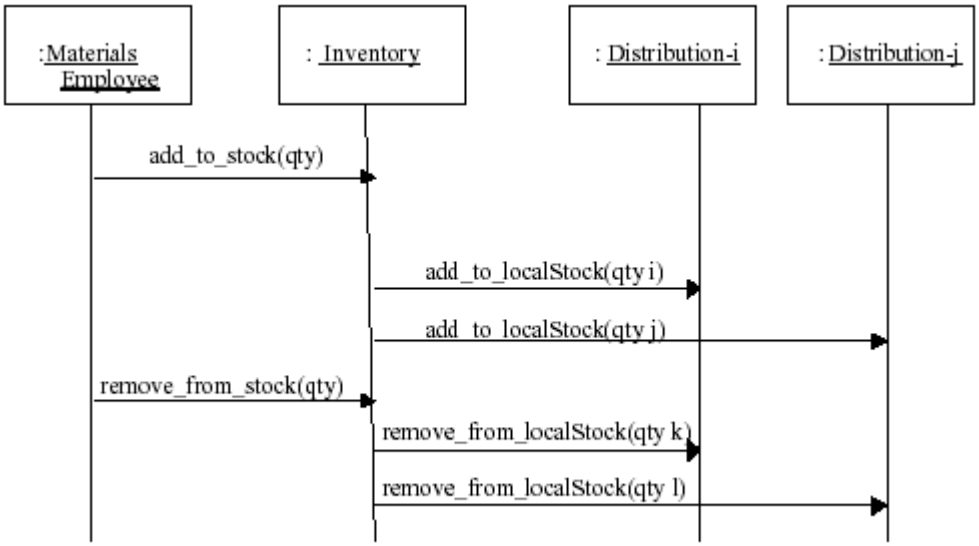


图 4.物品入库、出库的序列图

当零部件或产品入库时，通过调用 add_to_stock 操作，它们的手头数量（onHand）增加。当零部件或产品出库时，通过调用 remove_from_stock，它们的手头（onHand）数量减少。而 add_to_stock 操作根据预定的规则，决定物品放在哪儿。当零部件或产品被放到某个特定的仓库区域时，操作 add_to_localStock 增加局部的手头数量（localonHand）。相似地，remove_to_localStock 减少局部手头数量。

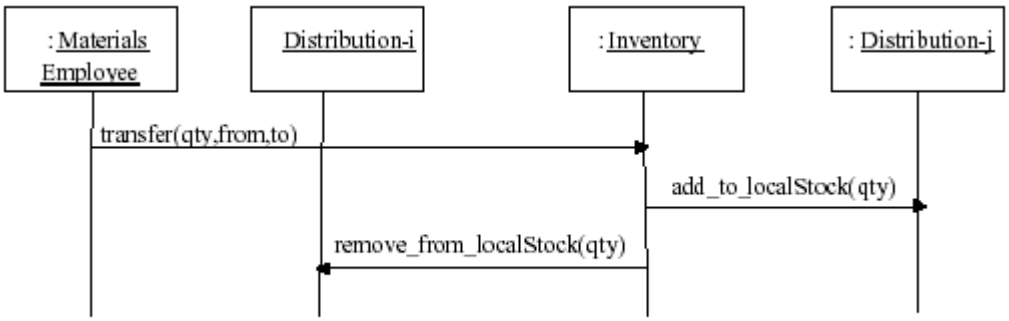


图 5.物品迁移序列图

零部件或产品可以从一个仓库区域迁移到另一个仓库区域，操作 transfer 执行这个动作。因为所有的移动操作既可以应用到零部件（由 Component 类表示），又可以应用到产品（由 Product 类表示）上，所以我们把这个操作放在父类 Inventory 中。当零部件或产品移出某个仓库区域时，remove_from_localStock 操作被用来更新局部手头数量（localonHand）。相反情况下，使用 add_to_localStock 操作。图 5 显示了物品迁移的序列图。

最复杂的库存变化发生在制造过程中（见图 6）。我们首先假设顾客的订单已经被处理，并形成了一张表单。表单中详细指示了在制造某种类型的产品时所需的零部件--什么型号的，以及每种型号的数量；这张表单就是生产指令单。当物料员根据生产单备料时，根据生产指令单上指示的数量，所需零部件保留数量的值增加。当零部件从仓库区域实际被领出时，保留数量和手头数量的值减少了。当制造过程结束时，产品库存中的手头数量增加了。通常，生产指令单从处理到结束，需经历几天时间。备料、领料到制造等阶段使人们知道生产指令单正处于什么状态。图 6 显示了制造过程中的库存变化。

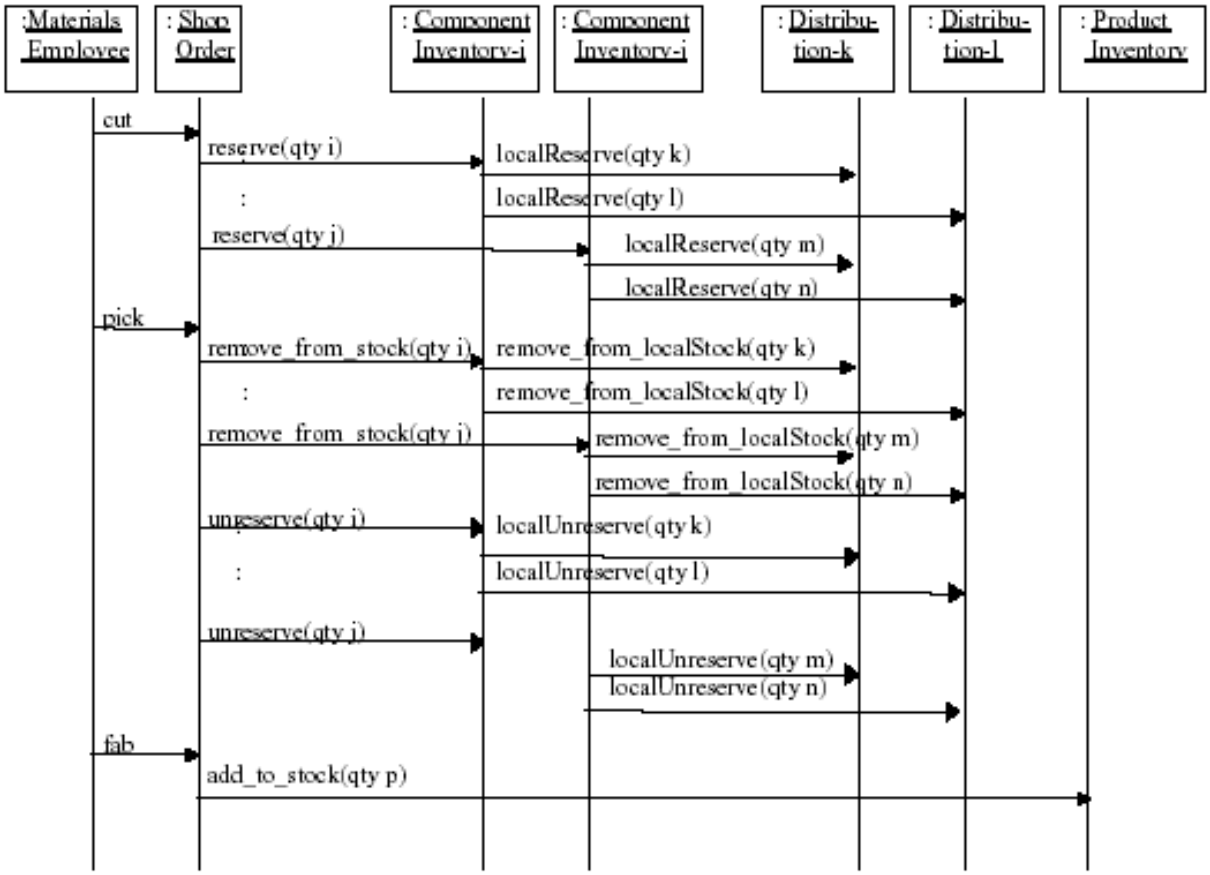


图 6.按照顾客的订单来制造的序列图

6　在图书馆库存管理中应用此模式

为了展示本模式在不同领域的使用，我们把它应用到一个图书馆库存中。如图 7 所示：

现在，作品仓库(Works Stock)类定义图书馆各种材料的仓库。分布(Distribution)类用来指示某个位置上的数量。这里的位置(Location)类可以表示借书处、图书存放架、缩微胶片中心、公共阅读区，等等。这里，我们显示了一个作品仓库；然而，这个仓库或许可以分成几个独立的仓库：书库，视频磁带库，CD 库，等等，因为这些种类的资料的处理方式是不同的。这个系统的用例(Use Case)是诸如借/还书的动作。这些用例(Use Case)的有些序列图与前面给出的序列图相似，如书的位置的迁移；而有些则不同，如顾客订单的处理。

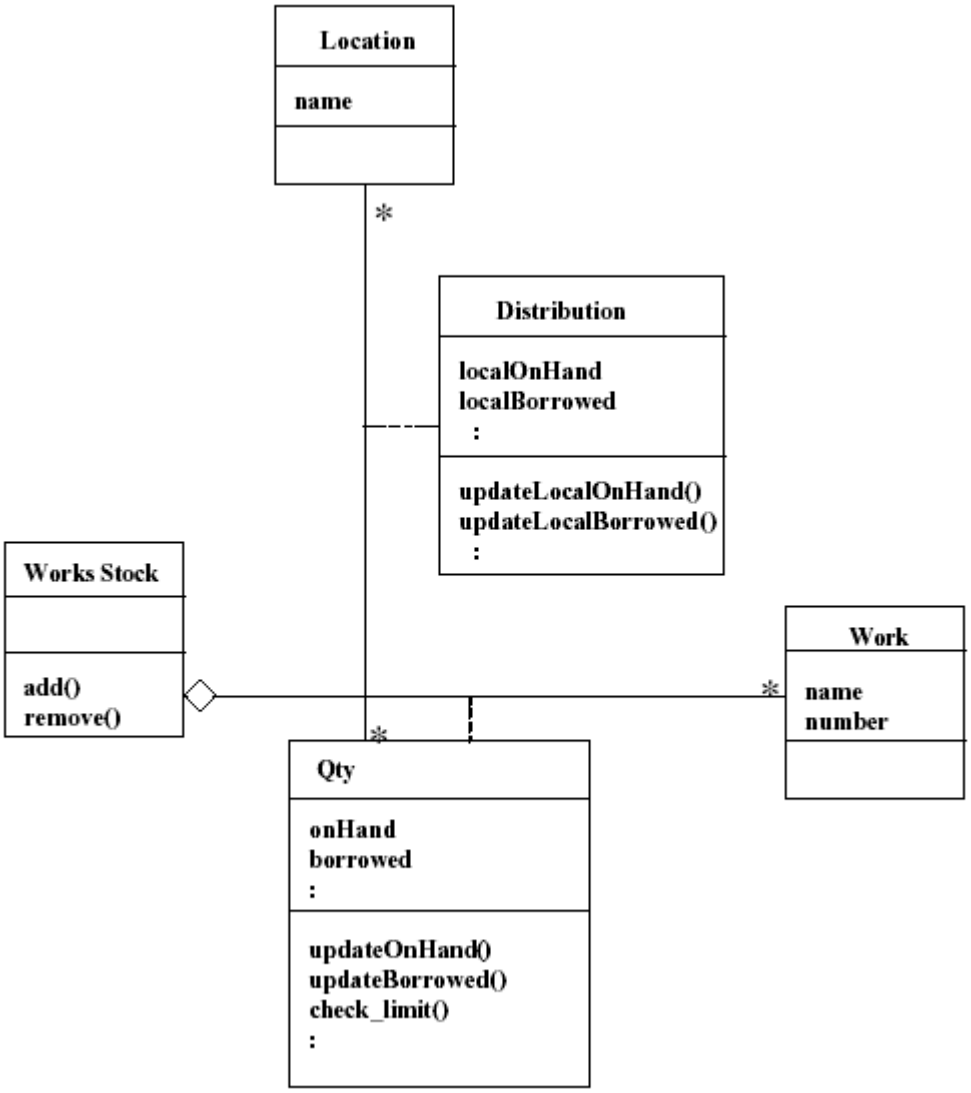


图 7.图书馆库存系统

7 结论

产生的模型满足了以下的约束：

- 该模式可以用于跟踪不同类型仓库中目标物品的数量和它们的分布位置。
- 其他动作产生的影响可以通过调用适当的操作反映出来。如图书馆的例子中指出的那样，对库存产生影响的动作在不同的应用中可以是不同的。
- 尽管该模式是用制造业中的术语描述的，但它仍可以用来表示诸如图书馆、商务活动、或者其他

相似的地方的库存。

- 文档，如顾客订单，和其他一些东西，被认为是与库存系统相互作用的外部系统的组成部分，因此没有在模式中表示出来。

为了使该模式能够在多个领域里应用，我们省略了：

- 物品的细节，如结构和描述。
- 存放位置的细节。
- 描述内部动作的文档，例如，在不同位置间货物的迁移。
- 异常处理，例如，保留数量比实际能获得的数量大，订单取消等。
- 提示低库存水平的报警。
- 历史信息
- 资金流动
- 当库存物品为液体形态时的处理。（度量液体时应该使用不同的方法）
- 这些方面应该通过其他模式来增加，或者是开发一个包含它们的完整框架。

8 已知的使用

一些库存模型的例子是：

- 在为位于美国佛罗里达 Boynton Beach 的摩托罗拉公司寻呼机产品分部的信息技术部开发一个库存原形时，应用了该模型。[Fern97]
- 在 Hay 的库存模型中[Hay96]，资产是些有实际物质形态的东西，(目前)必须存放在一定的场所，如仓库中的某一位置，制造工作中心。更确切的说，一项资产是一个场所中某种资产类型的物质形态。考虑了两种性质的资产：离散的物品和库存。离散的物品可以被单独的信息条所标识（如

序列号，公司标签号【译者注：原文中的 serial member 为笔误，应该为 serial number.】)。库存只用来跟踪大批量的资产。资产的类型又可以分为产品类型、物料类型、和零件/设备类型，这是因为以作者(Hay)的观点，物理形态不同的产品、零件、物料是按不同的方式来跟踪的。总的来说，贺氏模型中包含了我们忽略的一些方面。然而，它的模型没有包含动态的方面，属性，或操作；也没有把仓库（stock）与库存(Inventory)分开。

- Fowler 的文献[Fowl97]中有一章（第六章）写的是关于库存和会计的。他强调会计中的货物和资金的流动。他的模型中的另一个重要的方面是元数据（处于知识层中）和操作的分离。

- Hellenack[Hell97]讨论了作为其他业务模式一部分的库存。她分离了成品库存和原材料库存（与我们的产品[Product]和零部件[Component]类等价），也包含了一些资金流动。她的模型中只包含了静态的方面并且没有物品分布的管理。

9 相关的模式

保留和实体使用模式(Reservation and Use of Entity Pattern)[Fern97]，订单和发运模式(Order and Shipment Pattern)[Fern00b]，在几个应用中与本模式相辅相成，如制造业。[Brag98,Brag99]中的模式应用到了资源控制领域，并且看来会在很多应用中与本模式合作。一个可以使用“仓库管理器”模式的模式是依赖性需求模式(Dependent Demand Pattern)[Hau97]。因为产品和零部件是有类型的，所以类型对象模式(Type Object Pattern)也是相关的。最后，仓库/产品、仓库/零部件的聚合关系，是容器/环境模式(Container/Context Pattern)的一个实例。

10 致谢

Motorola 公司库存系统的原型是由 Michael Lynch, Chin- Shu Lee, Zhiwei Peng,and Mahbub Anwar 构建的， 他们都为本模式中的这些构思做出了贡献。Rosana T.V. Braga，我们的主管，以她富有洞察力的意见，对改进这篇论文的质量起到十分重要的作用。笔者在 PLoP 2000 的工作室也为改进这篇论文提供了有价值的建议。

参考文献

[Booc99] G.Booch, J.Rumbaugh, and I.Jacobson, "*统一标记语言用户指南*", Addison-Wesley 1999.

[Brag98] R.T.V. Braga, F.S.R.Germano, and P.C. Masiero, "资源管理模式的联合使用", *Procs. of PLoP'98,*http://jerry.cs.uiuc.edu/~plop/plop98

[Brag99] R.T.V. Braga, F.S.R.Germano, and P.C. Masiero, "业务资源管理的模式语言", *Procs. of PloP'99,*http://st-www.cs.uiuc.edu/~plop/plop99

[Coa97] P.Coad, "对象模型：策略，模式和应用" (第二版),Yourdon Press, 1997.

[Fern97] E.B. Fernandez , M. Lynch, and C.S. Lee, "库存控制系统的面向对象原型的开发", Report TR-CSE-97-32, Dept. Of Computer Science and Engineering, Florida Atlantic University, April 1997.

[Fern99] E.B.Fernandez and X.Yuan, "保留及实体使用的分析模式", *Procs. of Pattern Languages of Programs Conf.* (PloP'99),http://st-www.cs.uiuc.edu/~plop/plop99

[Fern00a] E.B. Fernandez and X. Yuan, "语义分析模式", *Procs. of the 19th Int. Conf. on Conceptual Modeling (ER2000),* 183-195.

[Fern00b] E. B. Fernandez, X. Yuan, and S. Brey, "产品订单和发运的分析模式", *Procs. of PLoP 2000.*

[Fowl97] M. Fowler, *分析模式 – 可重用的对象模型*, Addison- Wesley,1997.

[Hau97] R. Haugen, "依赖性需求—供应和需求平衡的业务模式", *Procs. of Pattern Languages of Programs Conf.*, PloP97, http://st-www.cs.uiuc.edu/~plop/plop97/Workshops.html

[Hay96] D.C.Hay, 数据模型模式—*思考的常规*, Dorset House Publishing, New York, 1996.

[Hell97] L. J. Hellenack, "面向对象的业务模式", *Object Magazine*, January 1997, pp. 23-30 and 70.

[John98] R. Johnson and B. Woolf, "类型对象", 程序设计的模式语言*3* 中的第四章, Addison-Wesley, 1998.

[Rie96] D. Riehle, "组装设计模式", Procs. of OOPSLA'97, 218-228.

[Salv92] G. Salvendy, 工业工程手册, Wiley- Intersience Pubs. ,1992.