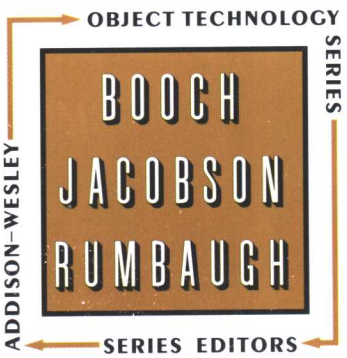
软件工程实践丛书

PEARSON
Addison
Wesley

UML对象、组件和框架
——Catalysis方法

Objects, Components, and Frameworks with UML
The Catalysis℠ Approach

（美）Desmond Francis D'Souza
Alan Cameron Wills 著
王 慧 施平安 徐 海 译



清华大学出版社

软件工程实践丛书

UML 对象、组件和框架

——Catalysis 方法

（美） Desmond Francis D' Souza
Alan Cameron Wills 著

王 慧 施平安 徐 海 译

清华大学出版社
北 京

内 容 简 介

本书介绍了如何使用对象、框架和 UML 表示法来设计、建立和重用基于组件的软件。Catalysis 是一种新兴的、发展势头强劲的、基于 UML 的对象和组件开发方法。Catalysis 提供了 UML 表示法的明确含义和系统的使用方法，并开辟了通过修改和组合通用的和特定领域的建模框架来快速建立模型的途径。

本书可作为计算机专业的教材，也可作技术人员参考之用。

图书在版编目(CIP)数据

UML 对象、组件和框架——Catalysis 方法/(美)德苏热(D'Souza, D. F.)，(美)威尔斯(Wills, A.C.)著；王慧，施平安，徐海译.—北京：清华大学出版社，2004.10
(软件工程实践丛书)
书名原文：Objects, Components, and Frameworks with UML: The Catalysis SM Approach
ISBN 7-302-09640-6
Ⅰ.U… Ⅱ.①德… ②威…③王…④施…⑤徐… Ⅲ. 面向对象语言，UML－程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2004）第 098315 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社总机：(010) 6277 0175　　**客户服务**：(010) 6277 6969
文稿编辑：汤涌涛
封面设计：立日新设计公司
印 刷 者：北京牛山世兴印刷厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印 张**：35.5　**字 数**：905 千字
版　　次：2004 年 10 月第 1 版　2004 年 10 月第1 次印刷
书　　号：ISBN 7-302-09640-6/TP・6684
印　　数：1～4000
定　　价：59.00 元

前　言

企业及其运营环境总是处于不断变化之中，几乎所有的企业都在工作中使用软件，而且许多企业还把软件嵌入到它们的产品中。我们开发的软件系统必须满足这些业务需求，必须能够正常工作、由团队有效开发并能够灵活应变。

软件的第一需求：完整性

前两项需求并非什么新闻，自 1968 年以来，软件界就一直在研究有助于理解和满足业务需求的方法。一般性桌面软件总体上可以正常地工作，偶尔也许会出现 bug，但这总比一直等到开发者开发出完美的软件以后再使用要好一些。另一方面，现在越来越多的软件被嵌入到各种各样的产品中，从汽车刹车、飞机和智能卡到烤面包机和补牙机，无处不在。有时，我们更应关注软件的完整性。学习本书所介绍的技术将有助于获得正确的需求，以及正确地实现它们。

软件的第二需求：团队开发

为了使开发任务能够在团队间进行分配，必须根据明确的依赖关系划分工作单元，构架约定和规则必须清楚，同时还必须明确无误地规范接口。组装组件的并不一定就是开发者，并且有可能在建立组件很长一段时间后才进行组装。必须能用系统的方法对实现、接口规范以及最终的用户需求之间的关系进行测试。

本书介绍的技术将有助于你构建具有以上这些属性的工作包和组件。

软件的第三需求：灵活性

为了保持竞争力，企业必须不断地推出新的产品和服务，因此，业务运营必须同步进行调整。例如，银行常常推出新业务，创造竞争优势以吸引客户。现在，它们通过电话和 Internet提供的新服务比通过其支行提供的新服务更多。灵活的软件必须随业务的变化而变化，这样才能从根本上保持竞争力。

灵活性不仅意味着能够迅速做出变化，而且还意味着能够同时提供多种变体。银行可能全局上部署了相同的基本业务系统，但是它必须能够修改该系统，以适应许多地方性规则和惯例。软件产品供应商不会把相同的方案强加于所有的客户，他们也不会每次都从头开始开发一个方案。相反，软件开发者更喜欢有一个可配置的软件产品族。

本书介绍的技术将有助于你用系统的方法划分和解耦软件部件。

灵活的软件

若要在短期内制作出大量软件产品，关键是能使一项软件开发服务于多种产品。重用并非剪切和粘贴代码：简单的剪切和粘贴操作会引起代码激增，以及无数次局部编辑，这样会使维护费用昂贵到难以承受的地步。

有效的策略是进行通用的设计，创建这些设计以在不同的软件产品中使用。这种可重用的资源不仅包括代码，而且还包括模型、设计模式、规范甚至项目计划等。

下面是建立一个可重用部件集合的两条重要规则。

- 可重用部件不能被使用它们的设计者所修改。你可能只想维护每个部件的一个版本，它必须有充分的适应性来满足许多需要，这可能通过定制来实现，而不是通过修改来实现。
- 各个可重用部件应当形成聚合的工具箱。与在汽车库中找到不同材料进行组装相比，建立像 Lego 那样具有个人喜好拼装结构的玩具要更加简单快捷。虽然在汽车库中找到的材料也可能是部件，但它们并非是按整体搭配的思想设计的。

那些能够用来配置但不能修改的可重用部件就是所谓的组件，它们的范围从没有程序源的编译代码到模型和设计的部件。

组件工具箱产品族

多年来硬件设计者们一直致力于研究标准化组件。他们不是设计一辆新的汽车，而是设计一系列汽车。通过组合基本的组件集以形成不同的配置，就可以设计出各种各样的汽车。只有极少数组件是专门为某个产品而制造的。一些组件是为了现有的产品系列而制造的，另一些组件是为了与以往的产品系列共用而制造的，还有一些组件是由第三方制造的，可用于其他品牌的汽车。

对于软件也可以这样做，但是我们不仅需要设计它们的方法，而且还需要创建和组装它们的技术。

组件技术

对于一个通用的组件，我们必须提供方法让客户的设计者能够根据他们的需要自定义组件。这些技术包括调用函数时传递的参数、组件读取的表、组件上的配置或者部署选项、插入点——组件中可以插入其他各种组件的位置，以及像工作流系统那样的框架，可以将各种组件插入其中。

面向对象(OO)编程强调可插性，或者说多态性的重要性，也就是使一种软件能够与许多其他软件耦合的艺术和技术。人们总是把程序分割成模块，而其最初的原因是为了在一个团队中分配工作，并减少重新编译。有了可插式软件后，这一思想就发生了变化：可以如同硬件设计者用一个包括芯片和底板的工具箱制成许许多多产品那样，以不同的方式组合组件来制成不同的软件产品，并可以延迟绑定时间(如图 0.1 所示)。

面向对象编程一再强调的另一个重要思想是责任分割。这一思想指每个对象或者组件(即可重用部件)应当有各自的责任，并且它的设计应当尽量不依赖于其他组件的设计，甚至

与其他组件是否存在无关。

不管是否使用面向对象编程语言，也不管是在程序设计中，还是在分布式系统中，或是在一个企业的部门中讨论对象，上述两种思想都在起作用。

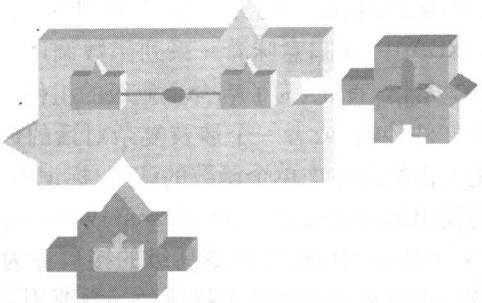


图 0.1　基于组件的组装，任意绑定时间

组件存在于哪里

从较小规模上看，可插式用户接口小配件形成组件。几种这样的工具箱可以在可视化编程工具中看到，如 Visual Basic 等，它们有助于把组件组合在一起。这样的工具箱还可以扩展到用户接口域外部的小部件（例如，VisualAge 和 JavaBeans）。所有的这些组件都可以在一个可执行程序中运行。

从较大规模上看，自含式应用程序可以互相驱动；对象链接和嵌入/组件对象模型（Object Linking and Embedding/Component Object Model，OLE/COM）、UNIX 管道和信号以及 Apple 事件就允许这种情况发生。通信组件可以用不同的语言来编写，并且每个组件可以在各自的空间中执行。

从更大的规模上看，组件可以分布于不同的机器之间。分布式组件对象模型（Distributed COM，DCOM）和公共对象请求代理构架（Common Object Request Broker Architecture，CORBA）是最新的技术；它们底部的各层（如 TCP/IP）提供了更加基本的连接。当部署这种规模的组件时，必须考虑各种新的分布式故障以及对象位置的经济性。工作流、复制以及客户机-服务器构架，或者说 n 层构架，提供了适合这种规模的组件的框架。此外，还有用于建立这种系统的工具和语言。Enterprise JavaBeans 和 COM+是更新的技术，它们解决了组件开发者在用大粒度服务器端共享组件进行工作时所遇到的许多问题。

从较大的规模上看，组件通常支持个人或者特定的职能部门所扮演的业务角色。企业越来越讲究开放式联盟构架，其中分布式组件的结构反映业务的组织结构。当重新组织一个业务时，我们需要能够以相同的方式重新编制软件。

基于组件的开发所面临的挑战

基于组件的系统的技术已经相当成熟，而用于开发它们的方法却不然。为了成功完成大型的企业级开发项目，需要明确的、可重复的开发规程与技术，良好定义的标准构架，以及同事间能够互相交流设计的明确标记。

建立组件工具箱的关键技术必须是明确定义组件间的接口。这就使我们回忆起前面提到过的完整性。如果打算把互不认识的设计者设计的部件插在一起，我们就必须明确连接中的协议：每一方应为其他各方提供什么功能，以及期望其他各方提供什么功能。

在组件技术中，诸如 COM，CORBA 和 JavaBeans 等，它们的重点在于定义接口（这种思想的发展历史可一直追溯到 20 世纪 70 年代出现的试验语言，例如 CLU）。其实，任何技术的重点都在于定义接口，如 UNIX 管道、工作流、RPC、通用数据库访问等。每当一个部件可以插入许多其他部件时，必须定义连接工作机制以及期望被插入组件具有的功能。

然而，在 Java 或者 CORBA 中，接口意味着一系列函数调用。由于以下两方面的原因，只定义接口还称不上良好的设计。首先，为了耦合企业级的组件，我们需要从更大规模上来表达连接：连接可能是一个文件传输，或者一个涉及复杂对话的数据库事务。因此，我们需要一种设计标记，这种标记无需始终考虑单个函数的调用；标记不仅应当能够表达流入组件的信息，而且还应当能表达流出组件的信息。JavaBeans（或 Enterprise JavaBeans）已经朝着这个方向有所发展了。在 Catalysis 中，我们把更高级的接口称为连接器（connector），以区别于基本的函数调用。其次，仅仅通过参数签名描述的函数调用不能充分说明期望的行为。编程语言没有提供这种功能，因为它们不打算表达设计，但我们需要编写准确的接口说明。需要的准确性特别重要，因为每个组件可能与其他未知的组件接口。在模块化编程时期，耦合模块的设计者能够在咖啡机旁解决问题；而在一个基于组件的设计中，不同的组件可能由两人分别设计，再由第三个人单独组装起来。

为了开发组件的聚合工具箱，首先我们必须定义一系列通用的连接器，以及组件间互相交流的通用模型。例如，在银行系统中，除非所有组件（在它们的连接器中，如果不是在组件内部）对客户、账户和钱等基本概念都使用相同的定义，否则不可能使组件可重新配置。

一旦定义了通用的接口集和构架框架，许多设计者就可以把他们设计的组件提供给工具箱了，然后人们就可以用组件来组装成产品（如图 0.2 所示）。

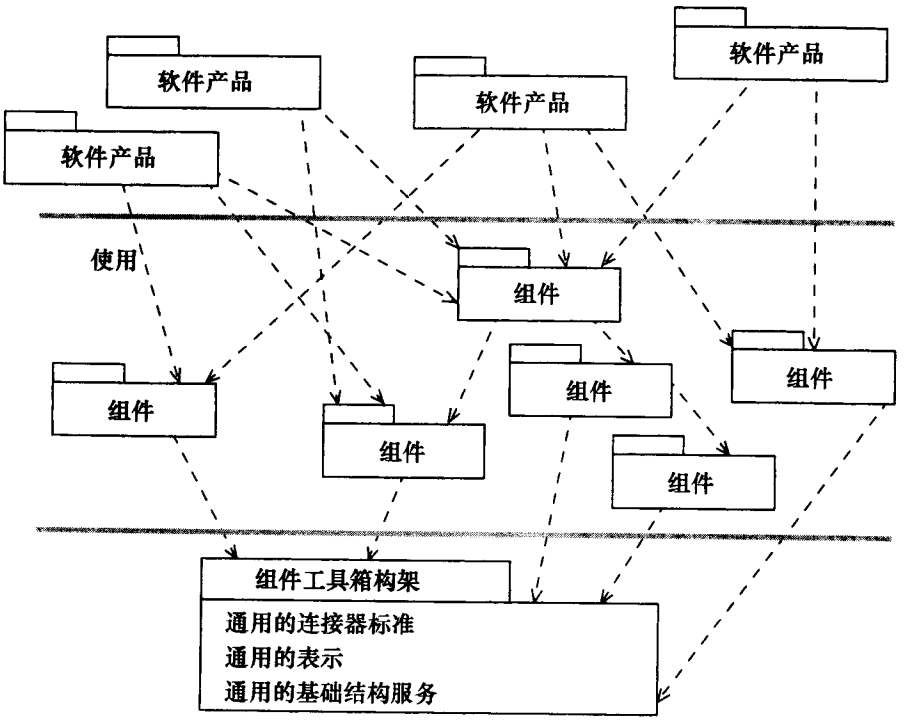


图 0.2　多种来源的组件组装成产品

Catalysis 提供的功能

如果说这种基于组件的场景看上去很不自然，请想一想 Babbage 的分析引擎(Analytical Engine)的命运。Babbage 不能使它正常工作，因为它有很多部件，并且没有提供把各部件很好地组合在一起的加工技术。随着软件行业不断提高匹配组件方面的技能和一致性，我们也将能够利用组件制成软件产品。

本书把一些组件技术集中在一起，我们认为它们是建立相干的组件工具箱所必需的。为了使基于组件的开发发挥作用，我们需要软件设计者的最佳技能，并且需要重新组织制作软件的方法。

Catalysis 中的技术和方法提供了如下功能。

- 对于基于组件的开发：如何在独立于实现的情况下精确地定义接口，如何构造组件工具箱构架和组件连接器，以及如何确保组件符合连接器。
- 对于高度完整性的设计：精确的抽象规范，以及从业务目标到程序代码的明确的可跟踪性。
- 对于面向对象的设计：明确的用例驱动技术，用于将业务模型转换为面向对象代码，具有以接口为中心的方法和高质量的保证。
- 对于二次工程：理解已有软件并且通过已有软件设计新软件的技术。

Catalysis 和标准

Catalysis 使用基于行业标准的统一建模语言(Unified Modeling Language，UML)的标记，现在对象建模组织(Object Modeling Group，OMG)已经标准化了 UML，本书的两位作者都参与了对象建模的 OMG 标准的提交任务，Desmond 的公司帮助定义并共同提交了 UML 1.0 和 UML 1.1。

Catalysis 已经成为 Texas Instruments 和 Microsoft 定义的组件规范标准、TI/Sterling 的 CBD-96 标准以及 Platinum Technology 的服务和产品的核心，已经有几家公司采纳它，把它作为基于 UML 开发的标准方法。它适应 Java，JavaBeans，COM+和 CORBA 开发的需要，并且支持 RM-ODP 方法。它还支持基于用例的系统开发。

Catalysis 的由来

Catalysis 基于并帮助形成了对象建模环境中的标准。它是作者在开发、咨询和培训过程中的工作总结，并且以金融界、电信业、航空和航天、GIS、政府部门以及许多其他领域的客户经验为基础。

Catalysis 中的许多思想是从其他地方借鉴而来的。参考文献部分列出了许多特定的参考资料。我们可以标识出 Catalysis 主要特征的资料来源，并对它们的作者表示衷心的感谢。

- 我们开始应用严谨的方法进行 OMT 对象分析[Rumbaugh91]。集成快照、事务、状态模型，从设计类中分别处理系统操作和分析模型，以及源于 Desmond 工作时间间隔的基本思想。
- 在以往一些面向对象开发方法中看到的严谨方面(规范、细化以及 VDM 和 Z 的影

响):聚合[Coleman93]、合成[Cook94]和 Bon[Meyer88]。我们把严谨的方法(例如 VDM 和 Z)应用于对象的兴趣可以追溯到 Alan 的博士论文[Wills91]。

- Helm,Holland 和 Gangopadhyay 的合同第一次介绍了把协作作为第一类设计单元,并在 Trygve Reenskaug 的[Reenskaug95]方法和 OORAM 工具中得到了发展。
- 抽象的联合动作源于 Disco[Kurki-Suonio90]、OBJ 传统[Goguen90]和数据库事务,以及 Objectory 用例的通用标记。
- 近年来,在各种模式中都提到了组件连接器。它们可以追溯到 Wong 的即插即用编程工作[Wong90]、以往关于代码框架的工作(大部分在 Smalltalk 领域)以及关于组件和连接器的构架工作[Shaw96b]。
- 过程模式是模式编程语言(Pattern Language of Programming)会议的几个投稿者的工作讹误。

在开发 Catalysis 期间,我们还从很多客户和同行顾问、教师以及研究人员那里获得了大量反馈意见。

如何阅读本书

不要一口气读完本书。如果你认为这个前言太长,可以等到看完本书其余部分后再来阅读。你将需要哪些背景知识呢?一些关于 UML,OMT,Booch 或者 Fusion 建模方面的基础知识对阅读本书会有所帮助;Martin Fowler 的简明 UML 综述非常易于阅读[Fowler98]。如果你已经对 UML 有所了解,请先看一看附录 B。

本书从第 1 章开始,带领你浏览设计工作的本质。在整个浏览过程中,我们介绍了所有主要的 Catalysis 技术,最后总结了我们的方法及其优点。接下来阅读随后各部分(I~V)的简介,以了解本书的结构。后面的大部分章节是这样设计的:你可以阅读每章的第一部分和末尾的小结,然后跳到下一章。当你按这种方法浏览完全书以后,就可以返回来深入阅读你感兴趣的内容。

我们在本书的某些地方讨论了关于建模方面的死角,学习时可以跳过这些内容;我们还在某些地方使用 Java 语言说明实现,如果你对这种语言不熟悉,也可以跳过这些章节。

第 2~4 章是基本原理,告诉你如何建立行为模型,以及它们表示什么和不表示什么。第 5 章至关重要,讲述如何把设计归档。第 6 章介绍了如何构造业务模型和程序代码间的精确关系。第 7~9 章介绍了把模型分解成可重用部件,并组合它们以形成规范和设计。第 10~12 章讲述了用可重用的组件建立企业级的软件。第 13~16 章介绍了应用 Catalysis 的过程,并非常详细地讨论了一个案例研究。根据不同的阅读对象,建议按如下方式进行阅读。

- 分析人员:如果你习惯于结构化方法,那么对主流的面向对象分析会感到比较困难。我们的方法在一些方面比较简单:研究系统级的场景,描述系统操作,获取在系统的静态模型中使用的术语,然后使用这个模型形式化操作。在其他方面,我们的方法更难一些,我们不喜欢模糊的和不明确的分析文档,因此我们推荐的一些精度与早期的需求活动不太密切。建议阅读第 1~7 章、第 9 章和第 13~15 章。
- 设计者:面向对象设计和面向对象分析一样新颖。同样,在某些方面我们的方法比较

简单。以非常明确的所需行为的描述开始，并且有可以遵循的到基本的面向对象设计的默认路径(参见模式 16.8)。对于基于组件的设计，除了设计组件层次外，需要使用分析人员的技术。

如果你是一位面向对象设计人员，请做好转向不同重心的准备。首先，你要理解作为一个独立实体的大粒度对象(系统、组件)的行为；然后，你要为它的状态建立一个独立于实现的模型，然后设计它的内部部件以及它们的交互方式。要严格地区分类型/接口与类，并始终用其他接口编写实现类。建议阅读第 1～6 章(跳过讲述规范细节部分的内容)、第 7 章、第 9 章、第 10～12 章和第 16 章。

- 实现人员：当满足功能需求的任务进入设计阶段时，面向对象实现应该容易一些。实现决策就可以集中于利用所选配置和语言的特征，它们是实现所有其余的需求所需要的。
- 测试人员：测试指通过运行测试数据和观察响应，设法证明一个实现不满足它的规范。规范描述的范围从应当调用哪一个函数到系统必须支持哪些用户任务；导出测试的方法也会相应地发生变化。请阅读第 1～6 章。此外，还要阅读关于 QA 的内容(参见 13.1 节和 13.2 节)，并且坚持在测试之前遵循它。
- 项目经理：仔细考虑使用组件或者对象的目标，以及建立灵活的可插式部件的理由(第 10 章)。密切注意项目风险，通常以需求和基础结构为中心(第 13 章)。与设计师一起，设计并遵循包结构的演变(第 7 章)和把组件插入包中的机制；如果有诸如开发构架之类的东西，也是如此。认识到团队间共享准确的术语的重要性(第 2～3 章)。阅读第 1～5 章，并有选择地阅读第 6～7 章、第 12～13 章。考虑从“Catalysis lite”(*www.catalysis.org*)开始。
- 工具制造者：Catalysis 在建模、一致性检查、可跟踪性、基于模式的重用和项目管理方面为自动的工具支持提供了机会。请阅读全书。
- 方法和过程专家：我们介绍的一些内容比较新颖；各部分紧密结合，而核心部分是很小的，因此要仔细阅读。请阅读全书。
- 学生和教师：本书包括了几个学期的课程和几个研究项目的资料，甚至可以作为特定课程的教材。几乎没有课程基于软件工程中严格的基于模型的方法。我们已经成功地将本书中的材料在几个为期一周的课程和专题讨论会中进行了试用。如果你希望在你的讲座中使用本书中的某些插图，你需要得到我们的认可。
- 其他：本书中的活动和技术既可以应用于大型项目，也可以应用于小型项目，但是它们具有不同的侧重点和明确的可交付项；还可以用于业务建模、软件项目投标、委外的服务和直接的软件开发，尽管当前描述的严格程度会令人感到畏惧。请访问 *www.catalysis.org* 站点。

何处可以获得更多信息

当你完成了本书的学习并希望进一步深入研究时，请访问 *www.catalysis.org* 网站，它提供了一些额外信息和共享资源，具体如下：

- 模型、规范、文档和框架示例；

- 本书中还没有完全解决的问题讨论：并发性、分发、业务过程模型等；
- 基于 Web 的论坛，用户、教师、顾问、研究人员、批评家和不怀好意的人共享经验和资源的邮件列表；
- 支持 Catalysis 开发和建模技术的免费工具和商务工具；
- 本书的联机版本和开发过程模式；
- 通用的建模练习和解答；
- 帮助他人使用 Catalysis 和改进 Catalysis 的资源，包括教育同行建模人员、设计人员和管理人员的简短幻灯片，可以分发的关于 Catalysis 的白皮书，等等。

此外，还可以查看作者所在公司的 Web 站点。每个站点都包含大量有用的资料，并且这些内容会不断地更新：

- *http://www.iconcomp.com/catalysis*——ICON Computing，Platinum Technology 公司(*www.platinum.com*)。
- *http://www.trireme.com/catalysis*——TriReme 国际有限公司。

Desmond Francis D'Souza
Alan Cameron Wills

目　录

第Ⅰ部分　概　述

第Ⅱ部分　对 象 建 模

第Ⅲ部分 分解模型和设计

第Ⅳ部分 组装实现

第Ⅴ部分　如何应用 Catalysis