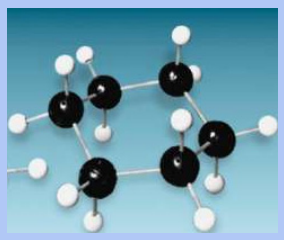
# 第 32 期（2003 年 12 月）





模型

X-Programmer 非程序员

软 件 以 用 为 本

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

http://www.umlchina.com/

# INTELLIUN 广邀专家一起探讨 EXECUTABLE UML 的未来

[2003/12/18]

Plano, TX (PRWEB)—Intelliun 今天宣布开始 Virtual Enterprise (VE)的下一阶段，所有用户都可以立即下载 VE 4.0 SE ，开始使用这种新技术，然后为即将到来的新特性投票，以驱动这个革命性新平台的发展。

VE/Designer 4.0 VE/Server

VE 是一种快速开发、集成和配置 web 应用和服务的平台。VE/Designer 是一个构造完全可执行 UML 模型的 IDE。VE/Server 是一个执行引擎，位于 IBM WebSphere, BEA WebLogic, Sun ONE Application Server, JBOSS Application Server 和 Apache Tomcat 等 J2EE 应用服务器的顶端，提供即时执行 UML 模型的服务。

Intelliun 正在寻求软件专家和面向对象专家的帮助，以改进 VE4.0 的特性。“我们正在开创新领域，把软件开发的抽象水平提升了一个层次”，Intelliun 的创始人和 CEOIyad Jabri 说，“我们希望得到指导，了解在众多的特性之中，那些特性是最突出的，这样我们增加特性时就不会危及 VE 4.0 的简单性”。

这项技术使开发人员的精力专注于从用例和分析对象模型中得到的应用逻辑，并自动产生表现层和持久层（即用户界面和数据库）。开发人员可以在技术和架构中立的情况下，快速和迭代地构造和验证应用的行为。

（自 emediawire，不得转载用于商业用途）

征稿

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

# 微软 Whitehorse modeler 的背后

[2003/12/16]

在领头人 Bill Gates 的宣告之后，Microsoft 开始通过称为“Whitehorse”的软件更新它的建模武器库。“Whitehorse”的目标是“design for operations”，允许用户尽早陈述逻辑基本结构需求，并把应用程序设置和逻辑基础结构相印证。

“现在的情况是，在基础结构团队和架构团队之间很少沟通”，Microsoft 开发人员部门的产品经理 Prashant Sridharan 强调。他描绘了这样的场景：逻辑基础结构（logical infrastructure）设计师和服务设计师共同使用 Whitehorse。“秘诀在于如何让这两种人一起工作”，他说。这个软件将缩短软件开发各个阶段之间的鸿沟。Whitehorse 将和下一个 Visual Studio .NET 版本－－Whidbey 在 2004 年末发布。

当问及 Whitehorse 中的 UML 支持时，Sridharan 说 Whitehorse 并不在本体内支持某一种建模语言，或者说可以支持很多种建模语言，包括 UML 这种面向对象的建模语言是其中一种。“我们尝试提供一种建模引擎和建模框架，这种框架是公共的”。

Sridharan 对这个产品寄予了很高的期望，“我们要迎来建模的全面兴盛时期，我们信仰模型驱动开发，虽然我们也信仰传统的代码开发”

他指出，目前一些微软开发人员已经使用了 Rose 这种 UML 工具，但 Whitehorse 将不同于 Rose。“UML 涉及的是面向对象领域，但面向服务的架构越来越常见，系统变得高度分布，象响应能力、安全性等因素必须考虑，面向对象方法未必是完成这些工作的最好方法。”

合作伙伴们得到 Whitehorse 的消息了吗？IBM Rational 总经理 Mike Devlin 说，“是的。Microsoft 在干一件漂亮的活，对于建模将向何处去，他们有自己的创见。我们和他们之间的区别是：我们是平台无关的，他们是平台相关的”

微软在推进建模技术－－仅仅是这一事实就值得关注。“他们进入建模领域的事实真令人兴奋”，Grady Booch 说，微软在行动，也证实了 IBM 和 Rational 自 90 年代中期以来的努力是对的。

（摘译自 adtmag，不得转载用于商业用途）

# 业界领先的美国电信设备制造商选择 Telelogic

[2003/12/12]

MALMO，瑞典-- Telelogic (股票号码:TLOG)，高端系统和软件开发解决方案全球提供商中的佼佼者，今天宣布一家业界领先的美国电信设备制造商已经升级了他们原有的 Telelogic TAU 产品，交易额为 120 万美元。这是这家公司在 2003 年耗资 116 万美元购买的 license 和服务的升级。


最新的协议包括了 Telelogic 的 TAU 第二代和 TAU 第一代工具集中的一些产品。其中包括 TAU 第二代工具集中的 TAU/Developer(R)和 TAU/Architect(R)，以及 TAU 第一代工具集中的 SDL 套件，UML 套件和 TTCN 套件。这些自动的开发和测试工具被部署在全球 25 个以上的地点。

“和一家世界领先的电信设备制造商达成协议，这是对我们产品性能的认可。”Anders Lidbeck，Telelogic 的总裁和 CEO，如是说，“应用 TAU/Developer 的自动代码生成能力的良好基础是达成这次交易成功的关键因素。”

基于UML(TM)2标准的TAU第二代允许公司部署集成的多用户可视化工具解决方案, 支持并加速应用开发的整个生命周期，从系统分析、规约到软件设计、实现和测试都在同一个开发环境之中。和劳动力密集型的手工编程方式相比，TAU 第二代工具应用模型来可视化开发过程，大大提高了开发效率。

在 2003 年 4 月的软件开发杂志进行的产品评测中，TAU/Developer 2.0 得到了四星级的好成绩，满分为五星级。四星级相当于是“Great”和“可以购置”的评价。软件开发是面向软件开发管理者的颇具影响力的杂志，其订户数目达到了 100840。

（自 primezone，袁峰 摘译，不得转载用于商业用途）

# IBM 如何对待 Rational 工具

[2003/12/11]

LEXINGTON, Mass. －IBM 正在把自身的开发工具与先前收购的 Rational 公司的工具结合，并称之为 IBM 软件开发平台。


周三在此举行的的一次媒体发布会上，IBM Rational 总经理 Mike Devlin 说，IBM 开发工具套件，WebSphere Studio，现在开始将处于 Rational 的支持下。

“从产品来看，这次集成超出了我的期望”，Devlin 说，“我很想知道客户们是如何反应，但以往他们是普遍支持的。”IBM 于 2003 年 2 月完成对 Rational 的收购，但一直遭受分析员、专家、顾客的担心，担心当两方的软件工具发生重叠时，IBM 应该如何处理。

在集成工作进行的同时，IBM 也在开展一项更大的计划，重新调整它的 5 个软件品牌－－WebSphere, Tivoli, Rational, DB2 和 Lotus，使它们“构件化”，从一个品牌出来的产品能够和其它品牌的产品一起无缝地工作。为了这个愿景能够实现，IBM 必须集成关键的 Rational 工具，使他们能和其它部门的产品一起工作。Rational 的产品包括允许开发人员按需选择过程构件的 Rational 统一过程（RUP）平台，以及 Rose、ClearCase、ClearQuest。

Devlin 说，Rational 已经可以在 Websphere 中间件产品上工作，提供 DB2 数据建模的支持。Rational 的一个目标是，继续努力使 Rational 和 Tivoli 结合得更紧密。

Devlin 也再次回答了 Rational 对关于 IBM 会怎样对待 Rational XDE 的问题，Rational XDE 是一个集成开发环境工具，能帮助 Microsoft .NET 和 Java 程序员更好地结合设计和编码环境。现在 IBM 合并了 Rational，工业界的一些人猜测 IBM 可能会迫使 Rational 抛弃对.NET 的支持，更多偏向 Java。毕竟，IBM 已经在 Eclipse 下了很多功夫，Eclipse 是一个开源的、基于 Java 的平台，开发人员群体和微软开发群体针锋相对。Devlin 称最终的目标是所有的 IBM 软件都基于 Eclipse 框架。他也指出，IBM 和 Microsoft 一起合作完成一些 Web services 标准，等这些合作完成后，他们还会在软件开发的许多领域经常合作。

“我们没有办法说服人们相信我们不会抛弃.NET，但行动胜过言辞”，Devlin 说。上一次在 5 月份的产品升级，支持.NET 的新特性要比 Java 多。IBM130 亿的软件业务中，有 20 亿来自面向微软的产品。

Grady Booch 在软件开发向何处去的讨论中指出，他相信 Rational 将帮助开发组织花更少的成本，有更高的产出。Booch 现在的工作是探索软件产业的未来，他认为开发人员和象 IBM 这样的公司需要“提升构造软件的经济”，因为现在软件开发逐渐变成人力密集型，抽干了人力资源的能力。

“我们生活在一个软件不断进化的年代，永不休止”，Booch 说，“我们过去常常构造许多单立的应用…但现在我们构造的软件不再是孤岛”，他说趋势已经转向分布式系统－－“一台你所不知道的计算机会使你的计算机崩溃。”

Booch 称其中的一个目标就是使 UML 能够更好地可视化描述复杂软件，更富有表现力，并通过使涉众能进行可视化测试来提升模型驱动开发，这是使工具加速进入市场的门径。

（自 internetnews，不得转载用于商业用途）

# Wizdom Systems 发布 ProcessWorks! V.6 支持 UML

[2003/12/1]

NAPERVILLE, IL (PRWEB)--Wizdom Systems，业务流程重组（BPR）17 年的领头羊，今天宣布发布 ProcessWorks! V.6。ProcessWorks!是一个 BPR 软件工具无缝套件，它能创建、分析和编辑业务流。新版本大大增强了用户友好方面和提供了支持 IDEF 和 UML 方法的能力。这意味着客户可以通过最新的业务建模技术从改进流程直接获益。

Wizdom™.com Wizdom Systems, Inc - *the process people*

“ProcessWorks! V.6 完全满足了我们的流程建模需要。” 成功的咨询公司 LogConGroup 的 CEO，Paul Ricciuti 说，“我们的客户需要严谨的建模工具，以前没有任何 BPR 工具和课程能满足我们公司及公司客户的需要。通过 ProcessWorks!，特别是新特性如 HTML 输出、从 IDEF 到 UML 的即时转换，我们的客户实现了立竿见影的价值和利润回报”

ProcessWorks!的优势

*提供处理业务的杰出方法

*揭露冗余和无价值活动

*建立业务分析的结构

*标出改进的要点

ProcessWorks! V.6 最新特性

1. 从 IDEF 模型直接得到用例模型，只需点一下按钮！

2. 文档管理器：链接遗留的业务支持文档。

3. 把模型以 HTML 页面方式输出

4. 还有很多…

详情参见 http://www.wizdom.com

（摘译自 emediawire，不得转载用于商业用途）

# Scott W. Ambler 谈敏捷的真相

[2003/12/1]

一些 IT 人在畏惧敏捷，但管理阶层应该去调研敏捷的真相。

最近我和一个朋友聊天时吃惊地了解到他所在公司 IT 部分正在考虑的一些新变化。我吃惊地发现他们选择了更加烦琐的规程，尽管这些规程看上去根本无法生效。我问他们为什么不考虑使用敏捷的方法，答案是：高层领导认为“敏捷”这个词太过激动。

问题的根本在于管理层没有真正地理解敏捷软件过程。他们听到了对之理解得不甚了了以及清楚从而很恐惧敏捷的属下的说法。敏捷计算的主要好处之一就在于他使得官僚主义和对组织没有贡献或贡献甚少的人难以容身。在过去复杂的传统的开发过程：重型和讲究文档的过程中，这些人能够很好地藏身其中，而敏捷方法却让他们没有藏身之地，或者意识到自己必须快速地提高自己的技能。

这种情况中，害怕敏捷的人会很狡猾地对管理层进行误导。而非常不幸的是，领导们往往想不起来请假一下了解故事的另一面的（像我这样的）专家。我曾经使用过诸如面向对象软件过程和企业统一过程（enterprise unified process）等传统的软件过程，也曾经使用过诸如敏捷建模和敏捷数据等敏捷过程（agile data）。

这家公司的领导听到关于敏捷的各种谎言，他们被告知，敏捷开发人员根本不建模，也不写文档，但事实是，敏捷开发者在每天都在进行分析和设计，而且要书写高价值的文档。他们被告知，敏捷开发者尽开发一些低质量的系统，这一点和事实恰恰相反。他们还被告知，敏捷就是黑客式开发的一个新名字，但只需要上 www.agilealliance.org 看看就知道事实并非如此。你真的认为业界的领袖们：Martin Fowler、Bob Martin、Alistair Cockburn、Barry Boehm 和 Jim Highsmith 都在倡导黑客式的开发方式？

敏捷是大家在一起高效率地工作，清楚所有沟通上的障碍，关注于增值的活动，从而使得开发更加成功。敏捷是大家肩并肩地工作，而不是什么都通过文档。敏捷是管理者积极地参加到项目管理中而不是成天去写状况报告，用那个来监督到底发生了什么事情。敏捷是开发人员和涉众（stakeholders，译者注：项目开发中涉及的从需求到最后交付的各种人员）在一起制定实际的计划，而不是用复杂的微软 Project 去制定一些几乎没人看的进度表。

公平地说，很难设想敏捷术能如何发挥作用，尤其是在一些软件开发本身都问题重重的传统组织中，被误导过的敏捷更是难以帮上什么忙。

（自 itbusiness，袁峰 摘译，不得转载用于商业用途）

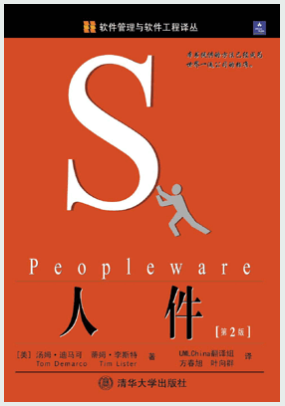
PH PTR

# 人件集

## ——人性化的软件开发

# The Peopleware Papers

NOTES ON THE HUMAN SIDE OF SOFTWARE




续篇

Larry L. Constantine 著

谢 超 刘 颖 谢卓凡 李 虎 译

人民邮电出版社


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

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

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

# 敏捷方法争论

Ryan Chen 译

查看评论

两位软件工程界大师关于敏捷方法实现细节的辩论

Tom DeMarco，大西洋系统协会

Barry Boehm，南加州大学

在2002年1月出版的《Computer》杂志中，Barry Boehm对敏捷方法提出了一些新的看法（《准备适应灵活的软件开发方法》(Get Ready for Agile Methods, with Care，2002年1月，64－69页)。 这篇来自业界大师的赞成性评论，促成了最近Boehm和业内权威Tom DeMarco之间的一些电子邮件交流，Tom DeMarco强烈提倡软件组织开始走向敏捷方法。－－ Michael Lutz，地区编辑

## 辩题

***Tom DeMarco:*** Barry，我很高兴看到你加入到关于敏捷方法的争论中。在你来之前，事情好像被认为是要对这样一个决议进行辩论，即敏捷方法是好的并且应该代替目前的重型过程（fixed processes）。

正方－Jim Highsmith 和其他的赞同者，认为当前的开发过程是不好的，应该被替换掉。而反方－例如 Steven Rakitin，认为敏捷方法只不过是编码的另外一个名字，我们不应该放弃严格的开发过程，既然我们已经努力使其正确。

我觉得它们两个阵营就象托洛茨基派和沙皇派一样。第一个阵营赞成革命，而第二个阵营则坚守CMM 3.621级的阵地，决不后退“1毫米”。你的文章找到了一个有意义的中间地带，辨认和保留了一些有价值的东西，而一些无价值的东西则被取代。

***Barry Boehm:*** 嗯，Tom，我很高兴把你对“敏捷方法争论”的清晰描述用克劳斯威茨的对位法（Clausewitz’s counterpoint）来做个比喻，就如军事行动中的装甲和机动性。不幸的， 我们在软件开发和军事行动中所看到的都是这么一种趋势：钟摆总是在两个极端之间摆动。然而在大多数情况下，我们需要在装甲和纪律、机动和活泼之间获得一个平衡。事实上，我要说敏捷和重型方法这两个阵营，它们的领导人的观点，其实都在各种各样的可靠的中间地带，只是那些狂热的追随者们过度地分割了“纪律”和“活泼”。

***DeMarco:*** 我很同意。幸运的是我们领域的实践者比学术的提倡者更懂得寻找有意义的中间地带。

既然你是业内第一个拥护风险管理（risk management）的，我希望你已经用风险的概念对敏捷方法相对于传统过程进行了比较。在速度和风险之间，敏捷方法提供了一种折衷的方式。所以它们并不是天生就是好的或者坏的，而是在好东西“速度”和另外一个坏东西“风险”里交替变化的。 在选择敏捷方法的过程中，管理者说：“我会牺牲一些可靠性来提高快速成功完成的几率。”如果速度真的非常重要，而且每个人都理解了这种增加的风险，那么这种考虑很有意义。

***Boehm:*** 我试着在我的文章里使用风险这个词，来帮助人们在给定的情形下，决定“多少的计划，多少的敏捷是合适的”。但是我很同意你的观点，即速度和风险的权衡是另一个需要考虑的重要方面。用内隐的人际间的知识（tacit knowledge）代替外显的文档化的知识（explicit knowledge），是敏捷方法在提高速度的同时限制风险的主要方式。

## 普通人还是超人?

***DeMarco:*** 在你使用“premium people”的时候，我感到很奇怪。这个词听起来很不像你写的，那个发明这个词的人是不是不是你，而是你的邪恶的孪生兄弟？

Barry，什么是“premium people”？他们是尼采的超人吗？他们是Aldous Huxley在Brave New World(译注：《美丽新世界》，很值得看的一本小说)里写的Alphas（小说里的α阶层）吗?

***Boehm:*** 我不打算更多解释这个术语，我们来讨论你已经发现的关于软件天才的限制供应。相对于那些天才比例高的组织，这些天才比例低的组织将得到一个更危险的速度与风险的权衡曲线。

***DeMarco:*** 如果你把你的文章中所有的“premium people”替换成“良好训练过的人（superbly trained people)”,我会觉得到更加合适。它们之间的不同是显而易见的:premium意味着先天的条件,而superbly trained意味着后天的或者说可以获得的条件.

举个例子，我曾经很奢侈的花了一个星期时间来参加Kent Beck的XP沉浸式强化训练，离开时，我被人们在这个体验的过程中的成长深深地打动了。他以一种技能练习的方式教人们一条到关键实现步骤的途径——规格说明（specification），版本控制（versioning）， design partitioning，测试（testing），等等。当他们完成那一周的练习，课程的参与者们具有了强大的新能力，创作出来的设计，比他们单独创作的更好。

***Boehm:*** Kent的XP训练非常好。但是看看XP和其他严格的新方法（比如个人软件过程，小组软件过程）早先报告的成功率，我想我们仍然看到了它们是怎么在以前过程的基础上工作的。尽管贤明的导师们想去帮助两者，我们仍然必须去决定如何让这些方法对大众和落后者起作用。

另一方面，我将会说XP训练是必要的，但它不是多数应用的成功的充分条件。大多数的应用项目需要至少一个人，他同时具有很强的软件能力和很强的应用领域的技能，就象Bill Curtis的“千里眼的守护者（keeper of the holy vision）”，而这种应用领域的能力，不是一个星期就可以学到的。

最近，我和Alistair Cockburn在我们的南加州大学软件工程中心（USC-CSE）的敏捷方法研讨会里的一次讨论中，解决了这个问题。“如果你只有一个同时具有很强的软件能力和很强的应用领域能力专家，但是你有5个项目需要她，你会怎么办？”让她进入一个项目，这个项目将变得敏捷，而另外四个变得危险。让她同时在5个项目工作，则将需要以文档的方式把知识明显（explicit）表达出来。 我就经常碰到这种情况。

***DeMarco:*** 这个例子漏了真正的重点。开发过程中困扰了我们20年的一个部分，就是我们试图要把过程应用到组织 级别，而不是个人级别。我们花了很多钱来教育组织如何去构造系统。但是敏捷意味着多投入到个人技能的构造，而不是组织的规则设定。我接触过的那些大多数没有足够的“超级训练过的人”的公司，它们之所以没有这些员工，是因为它们压根就没有试过。一旦它们开始把注意力放在帮助员工构造个人技能方面，我们将看到你和Alistair讨论的问题只是暂时的。

***Boehm:*** 我同意，在敏捷方法和个人软件过程中针对个人技能增强的注意，是迟到的。但是我仍然认为当个人没有足够所需要的业务领域知识时，如果勇敢坚持下面这些敏捷方法的原则风险会很高，比如：满足客户，在软件上集中注意力和信赖有动力的个人。举例来说，我们曾经有些团队努力去满足一个客户的对自然语言处理能力的需求。他们关注于已有的自然语言处理软件（NLP software）的原型版本，而没有意识到这些都不是健壮的系统。没有一些外在的明确的文档，没有一个可以检查的框架，就很难判断这些项目什么时候偏离正轨。

## 计划还是超级计划？

***DeMarco：*** 一开始，你对SEI-CMM的描述，以及类似的“计划驱动的开发”，让我很着迷。但是然后我开始感觉到它们都是错误的。极限编程比大多数CMM组织所做的包含了更多计划。每天都有具有意义的、很重要的、复杂的计划步骤来说明下个将要发布什么版本，包含什么东西，将要作什么测试，如何模式化设计。因为开发者会反复的回顾很多这样的决定，这就是重构所全部要做的事情。思考规划能力，在XP里比在其他任何固定过程驱动的方式里，会得到更多的练习。

在目前这个流行潮流中，把CMM叫做“模板和计划驱动的”的过程更合适。

***Boehm：*** CMM非常容易被官僚化、模板化地执行，这点我同意。我已经高兴的看到CMM已经被CMMI替换，CMMI强调风险管理和团队集成。你可以使用一种风险驱动的文档方法来避开那些即使没有写风险也较小的文档，也避免那些你试着写它们时会引起很高风险的文档，如用户界面文档。这同样适用于风险驱动的活动中的相互检查和独立的质量担保。CMM 和CMMI的领袖们对敏捷方法的态度是友好的，其中例子就是Mark Paulk的文章 “CMM观点中的极限编程（Extreme Programming from a CMM Perspective）”（*IEEE Software*, Nov.-Dec. 2001, pp. 19-26），这是一个值得鼓励的倾向。

我还可以找到很多应用，在这些应用中，需求是相对稳定的，一个预先计划的结构可以成功地适应以后增加的需求。在这些例子中，相信“你将不需要它”，并且一次一次的重复检查决定变成了一种不必要的返工和一种对你的客户的凌辱。然而，我看到一个明显的倾向，就是越来越多的应用里有动态的和不确定的需求，这些应用就需要使用敏捷方法。

***DeMarco：*** 在经过20年的整个产业性的关于过程的困惑之后，我在客户公司里遇到的经典的开发过程，开始变得太过于以文档为中心，结果就是文档膨胀在我们领域里成了一种职业病。开发过程的任何一个新的变化，我们都是倾向于添加：“呀，我们在x项目中发现了这种问题，所以我们要在我们的开发过程中加入一个新的部分，并把它强加到其他所有的项目。”任何的变化都增加了文档，从来不会去减少。对固定过程的这种被广泛谅解的缺陷，敏捷方法是一种反冲。如果不去考虑文档 膨胀，你就忽视了敏捷方法所能帮助的主要方式之一。

***Boehm：*** 我同意，文档膨胀是一个大问题，总体来说敏捷方法可以帮助去对付它。就象CMM/CMMI 第5级所做的，它们提倡我们要正视我们作的过火的事，比如文档化， 然后纠正回来。不幸地，太多的实施了CMM的组织停留在了第3级，直接的结果就是文档膨胀，这是政府自身引起一个问题，在众多的选择中它仅仅要求第3级。

Agile软件开发丛书

PEARSON Addison Wesley

# 有效用例模式

# Patterns for Effective Use Cases

Foreword by Craig Larman

[美] Steve Adolph
Paul Bramble 著

车立红 译

UMLChina 审

UMLChina 指定教材

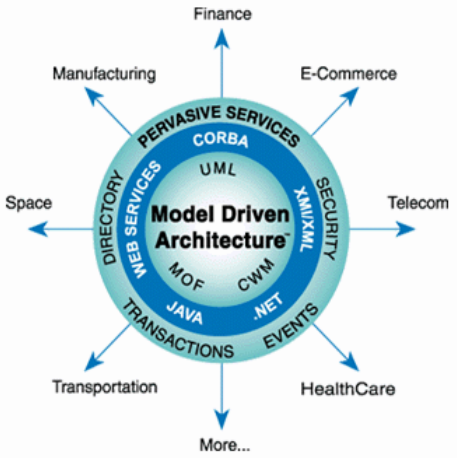
清华大学出版社

# 模型驱动架构 MDA 介绍

袁峰 著

查看评论

模型驱动架构（MDA，Model Driven Architecture）是国际对象管理集团 OMG 目前力推的软件开发的新框架，它在模型定义、软件开发过程等领域必将产生深远的影响。本文从其定义、思想的来源、发展历史等方面对其进行了较为深入的介绍。




## MDA 的定义

关于 MDA，OMG 并没有给出一个明确的定义。在【Joaq03】中提到：

“It is an approach to using models in software development.”

在【Davi03】中提到，MDA 是一种基于 UML 以及其他工业标准的框架，支持软件设计和模型的可视化、存储和交换。以独立于实现的技术开发，以标准化的方式储存机器可读和高度抽象的模型，并进行数据交换等操作。

简单地说，MDA 是 OMG 在模型可执行上的努力，其目的是为了“把建模语言当编程语言来用，而不只是设计语言” 【Davi03】。模型可执行（Executable Model）是 MDA 的终极目的，为了实现这个目的，OMG 制定了模型的精确形式化表示、模型存储以及模型交换方面的各种规约如 UML2、MOF（Meta Object Facility，元对象设施）、OCL（Object Constraint Language，对象约束语言）、QVT（Query/View/ Transformations）、XMI（XML Meta-data Interchange，XML 元数据交换标准）等等。

MDA 不是某一种具体的技术，也不是一种具体的方法论1，它是包含了诸多规约的一个集合，是 OMG 提出的在模型驱动开发方面的一个总的架构。模型驱动工程 MDE（Model Driven Engineering）中包含了这些规约以及应用规约的过程：MDA+过程=MDE。

## MDA 的思想本源

从 MDA 的发展历史，能够更清晰地理解 MDA 的思想本源，本文从两个角度来介绍 MDA 的发展历史。

1） MDA 提供了统一的元数据管理框架。

2） MDA 是 OMG 在模型可执行上的努力。

### MDA 提供了统一的元数据管理框架

由于商业利益及其它一些原因，软件行业的发展中经常出现各厂商之间产品不兼容的情况。这种情况的存在对于软件的集成和发展都造成了障碍，直接造成了软件成本的提高和应用及开发的复杂度。

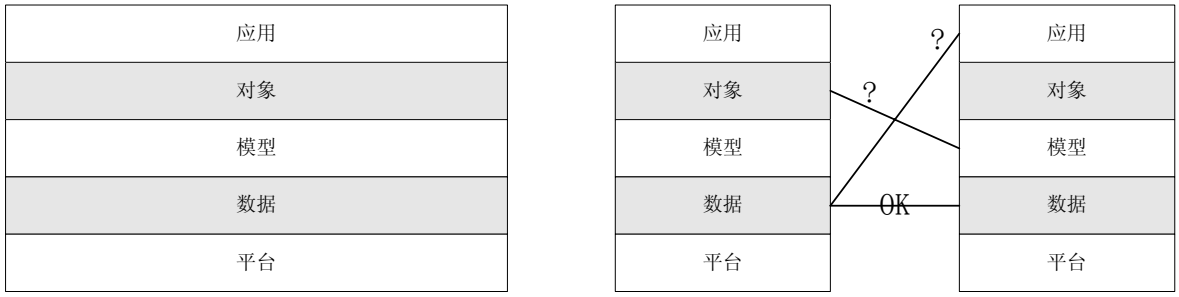
国际标准化组织 OMG（Object Management Group）是独立于各厂商的非盈利性组织，其宗旨是要统一不同的商业产品和标准之间的数据交换及互操作，从而改善各厂商的软件产品之间不兼容的情况。

CORBA（Common Object Request Broker Architecture，公共对象请求代理结构）是 OMG 以往的一个重要工作成果，CORBA 致力于提供一个跨平台和跨语言的对象互操作框架，在中间件层次上统一不兼容的平台和不同语言开发的对象。设想是很好的，但目前来看，OMG 的这个努力已经被证明没有达到预期的效果。在中间件级别涌现出来的 EJB 和.NET 造成了新层次上的不兼容。从这个意义上，MDA 是在比 CORBA 更高的抽象层次上进行统一的努力，是 OMG 一贯思想的自然延续。

但是，假设在中间件层次上 CORBA 成为唯一的标准，是否一切产品不兼容的问题就得到解决呢？

回答这个问题之前，首先看一下在统一软件产品不兼容上的另一种重要努力：W3C 组织提出的 XML 语言。XML 在数据的层次上进行统一，XML 及其相关的各种标准如 DTD、Schema、XSL、SOAP 等，已经被证明是非常成功的。但是，如图 1 所示意，我们将不兼容的情况分为纵向和横向两个维度，XMI 可以解决不同数据之间的不兼容，CORBA 可以在一定程度上解决不同对象之间的不兼容。但不同的软件产品之间的交互，不仅限于数据与数据之间、对象与对象之间、或者模型与模型之间，不同层次之间的交互需求同样非常多，尤其在下文将提到的模型可执行的实现上，模型和源代码的交互（由模型生成代码），模型和应用的交互，非常重要。以往的解决思

路都没有考虑到不同层次间不兼容问题的解决，MDA 则在这方面做了重要的工作。




（ a ）　　（ b ）

图 1 不同层次上的不兼容

不同层次之间的不兼容，实质上是元数据的不兼容。举例来说，目前对于同一个组件，可以使用 Java、XML、CORBA IDL、UML 等不同的语言来描述。所有这些描述信息都包含描述关于组件的元数据。目前，这些元数据的描述语言都是各自不同，它们之间的结合目前也没有一个统一的标准，例如 JAVA 和 XML 之间的交互是 JAXB 等，UML 和 XML 的交互是 XMI 等，但 JAXB 和 XMI 之间并没有遵循什么一致的标准。

MDA 的一个重要贡献正在于提供了这样一个管理不同元数据的统一框架。和以往进行数据格式统一管理的思路不同，MDA 允许使用不同格式（不同元数据描述）的数据存在，并为描述这些数据的语言（其元数据）提供了统一的定义语言，即 MOF。

MDA 通过基于 MOF 的四层元模型架构对元数据进行统一管理，不但能解决图 1 中不同抽象层次上的不兼容的问题，还可以在更广泛范围元数据的管理上发挥作用。举例来说，考虑一个软件开发过程，对于相同的组件，在需求阶段，使用需求模型来描述，在开发阶段，使用 UML 模型来描述、在测试阶段，使用某特定的测试模型来描述……，由于侧重点的不同，不同阶段对这同一组件的描述使用了不同的元模型，对这些元模型描述得到的模型，它们之间存在交互和转换等需求，MDA 统一的元数据管理框架，能够有效地支持满足要求。如图 2 所示，MDA 框架（这里起到重要作用的是 MOF）在这里起到了一个公共语义总线的作用。

图 2 中上方是使用各种不同的元模型表示的模型，下方是 MDA 框架为元模型的一致存储、交互和转换等提供的支持，包括了一系列相关的标准，这些包含在 MDA 框架内。

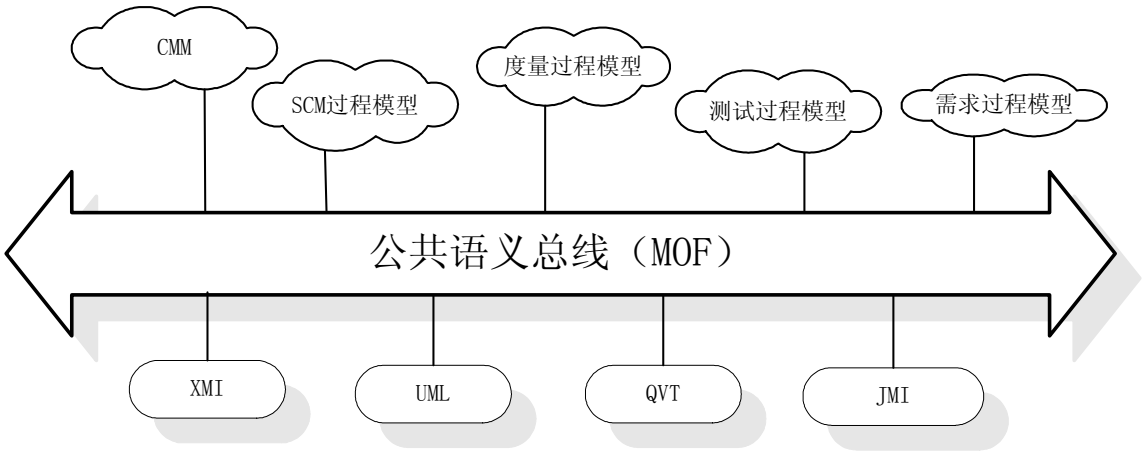



图 2 公共语义总线 MOF

### MDA 在模型可执行上的努力

在第 1 节中已经提到，MDA 的一个重要目标是加强模型在开发中的作用，简单地说，就是实现模型可执行。这实际上也是软件行业发展的必然之道。从机器语言、汇编发展到现在的高级语言，软件开发的抽象层次越来越高。这也就意味着，开发人员越来越多地关注问题本身。

MDA 将抽象层次提高到模型的高度，其理想情况是：开发人员只需要考虑和业务逻辑相关的模型设计，模型到应用的实现，是和业务无关的、可重复的、低层次的工作，交由机器来实现。目前，MDA 要做的是将业务逻辑从不同的中间件平台（JAVA、.NET、CORBA、Web Service）中剥离出来，开发人员只需要考虑业务逻辑的建模，得到的是和具体中间件技术细节无关的 PIM（平台无关模型），在 MDA 提供的支持下，将得到的 PIM 转换得到对应的加入了技术细节的不同平台上的 PSM（平台相关模型），它的表现形式可能是一个可运行的 JAVA 程序，或者 window 系统上部署的一个 Web Service，这样，MDA 的模型可执行得到满足。

从 2.1 的角度来看，开发人员设计的 PIM 和最后的可运行 JAVA 程序对应的是同一个组件。但使用了不同的元数据来进行表示。前者的元模型可能是 UML 元模型，而 JAVA 程序的元模型则是 JAVA 的语义规范。从模型到 JAVA 程序，模型的可执行，实际上还是对应着图 1 中的元模型的转换。

## MDA 的发展历史

简单地了解 MDA 的发展历史，有助于理解 MDA 的思想、应用，目前存在的问题以及将来的发展方向。

前面已经提到，OMG 的过去的主要工作 COBRA 并没有达到预期的效果。但其引入的另外一个标准 UML 却获得了空前的成功。作为一个应用于面向对象建模领域的建模语言，UML 目前已经超出了其最初的设想，俨然成为“软件工程的语言”。

UML 广泛而成功的应用是 OMG 提出 MDA 的一个重要基础，但另一方面，MDA 却来源于对 UML 的审思：UML 不是万能的，它是一种元模型，但并不是唯一。在其他领域，SPEM、CWM 等元模型以及 OMG 最近推出的 ODM（本体定义元模型）和 UML 同处于 MDA 四层元模型框架中的 M2 层：元模型层。其最高一层，M1 层，正是 MDA 的核心 MOF，它是 MDA 整合元模型管理的基础。

这里介绍一下 MDA 中非常重要的四层元模型架构。为了保证元数据管理框架的可扩展性。OMG 采取了 ISO 和 CDIF 等标准化组织所使用的经典的四层元模型标准架构。如图 3 所示：

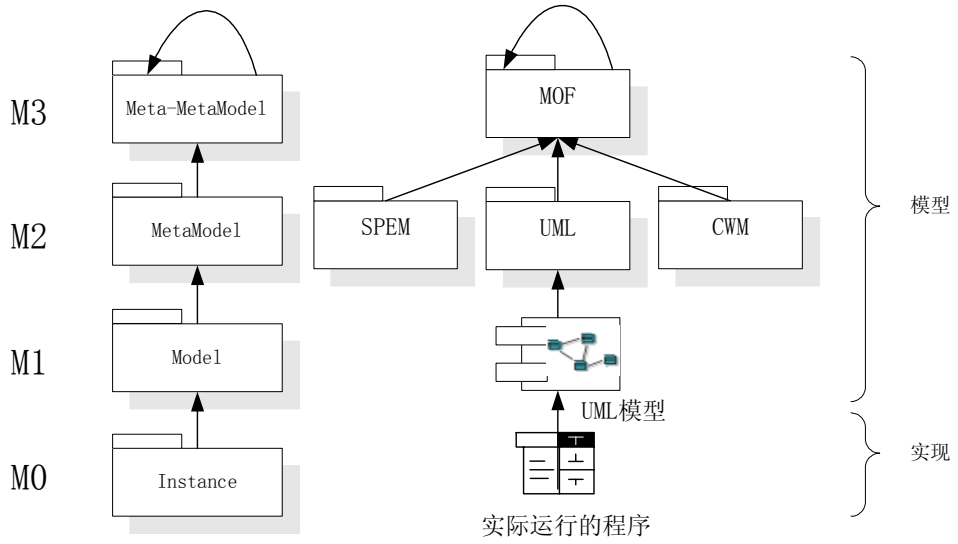



图 3 四层元模型结构

其中：

- M3 层：元元模型层，在 MDA 中，这一层上只有 MOF，MOF 是可以自描述（self-describing）的，也称为是反射（reflective）的。
- M2 层：元模型层，这一层上是应用于各不同领域的元模型，例如，UML、CWM、SPEM 等。
- M1 层：模型层，这一层上是利用 M2 层提供的元模型进行实际建模工作得到的产品，例如，使用 UML 对网上售货系统创建的模型。

● M0 层：实现层，是 M1 层中模型的实例化。例如，一个实际可运行的网上售货系统。

模型中各层之间的关系是“instance of”，每一层都是上一层的一个实例，例如：实际原型的程序是 UML 模型的实例，而 UML 模型是 MOF 在面向对象软件分析与设计领域的一个实例。对于定义需要利用工具进行可靠存储、共享、处理和交换的复杂模型而言，四层元模型结构已经被证明是非常有效的一种模式【Cris99】。除此之外，【OMG02】中还列举了使用这种结构的各种好处。

在理想的 MDA 架构中，最高一级（M3 层）的元元模型（元模型的元模型）只有 MOF，所有 M2 层的元模型都是基于 MOF 定义的。也即，UML 是基于 MOF 定义的，SPEM、CWM 等也都一样。

但 MOF、UML 等的实际发展却和理想情况相差甚远。

OMG 在 1997 年提出 MOF 的时候，就指出【OMG97】:

*MOF is the core of OMG's meta object framework specification for integrating industry standard meta models such as the UML.*

这个定义直到目前依然正确，但 OMG 当时还不足够坚定，同在【OMG97】中，又有如下描述：

*MOF 集成了元数据的自描述、UML 的建模概念以及 CORBA 的架构来提供一个在 CORBA 分布式环境中描述和管理元对象的统一架构。*

可以看到，当时 CORBA 在 OMG 的地位还相当高，MOF 和 CORBA 的关系非常暧昧，它提供管理诸如 CORBA、COM 接口、UML 设计、数据库定义等元对象提供的对象仓库架构。甚至 MOF 的早期版本中，基本数据类型都与 CORBA 一样。

**但 MOF 并非是基于 CORBA 的**，随着版本的升级，这一点也越来越明显，MOF 和 CORBA 不在一个级别上，MOF 是描述 CCM 的语言，在 MOF 白皮书【OMG02】中，只有“MOF to IDL Mapping”这一节中涉及到了 CORBA。

上面我们理清了 MOF 和 CORBA 的关系，下面来分析 MOF 和 UML 的关系。按照 MDA 的理想架构，MOF 是 UML 的定义语言。但实际上，UML 已经发展为了一定的成熟版本（1.1），被 OMG 接纳为标准的时候，才有 MOF 的诞生。并且，在 MOF 的定义中借用了 UML 的一些概念和建模结构，用作描述建模结构的抽象语言，因此，MOF 元模型看上去就象 UML 类模型，甚至可以用 UML 类建模工具来创建。

曾经有学者提出直接使用 UML 来作为 M3 层的元模型【Jean03】。其理由是：UML 的表示能力已经足够强大，可以满足定义元模型的需要，可以满足自描述的需要。而且，使用 UML 作为元模型，可以直接使用已有的 UML

CASE 工具来进行元模型的定义。

不过，要达到自描述的要求，UML 中很小的一部分就已经足够，像用例图、状态、顺序图等很多部分都是 UML 专为描述面向对象的分析和设计而设计的。在描述 CWM 和 SPEM 等其它元模型时，保留这些部分并不明智。因此，MOF 中提取了 UML 中定义元模型所需要的最小集，用它来定义 UML 以及其它的元模型。

但是，由于前面提到这些发展中的原因，UML1.x 和 MOF1.x 中存在很多不一致的情况。UML 和 MOF 的抽象语法应该共享同一个核心，但实际情况却并非如此，有一些 UML 类建模元素如关联类、修饰符以及 N 元关联、依赖等关系都不为 MOF 所支持。

这些不一致给 UML 和 MOF 的结合使用带来了很多障碍。使得 UML 模型之间通过 MOF 的转换和交互难以实现。为了改善这一点，UML2.0 和 MOF2.0 都在朝着一致性的方向做出努力，UML2.0 在 UML1.X 的基础上进行了重大的结构调整。新的结构包括 inftastructure 和 superStructure 两部分。其中 inftastructure Library 在设计的时候就考虑到 MOF2 的重用需要，分为 CORE 和 Profile 两个包，在 CORE 包中提取了过去版本中的抽象语法和语义定义部分，改善了与 MOF 不兼容的情况。这部分在 MOF 和 CWM 都可以进行完全的重用。如图 4 所示。

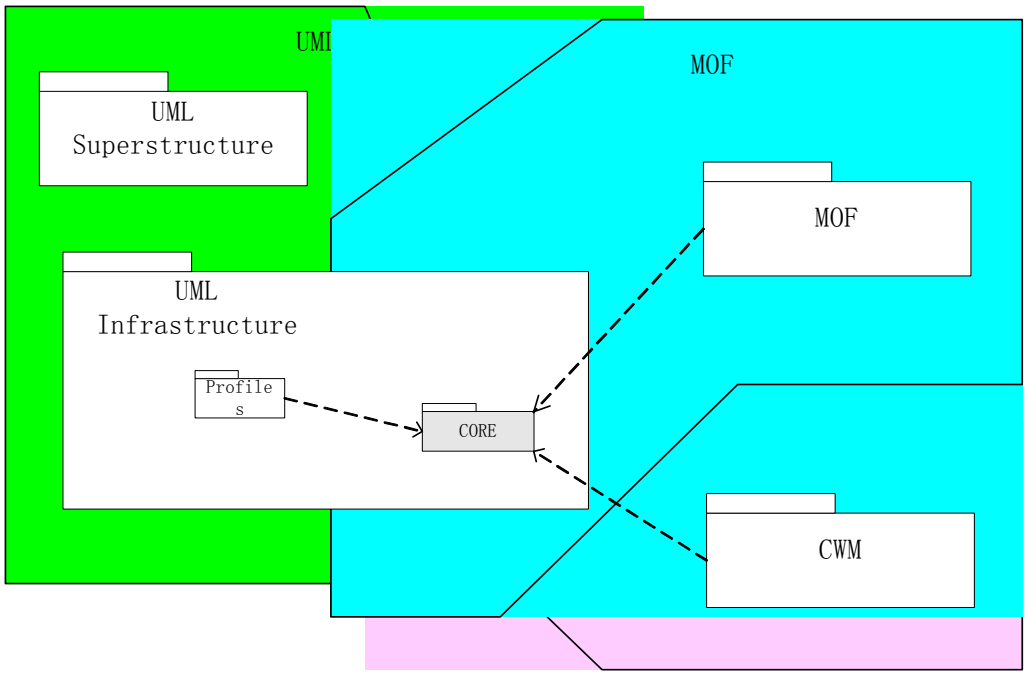



图 4 UML、MOF、CWM 的关系

CORE 包在设计时被赋予了高度的可复用性，可以将其看做是一个完整的元模型，它实际上是整个 MDA 框架的核心所在。MOF2 CORE 的工作小组（MOF 2.0 Core submission team）已经声明 MOF2 将与其保持一致。【OMG03】

使用了同样的 CORE，UML2 和 MOF 2 的一致性将得到很大改善。这将带来如下好处：

1. 简化了元数据的定义，定义人员只需要了解 UML 类建模的子集即可，不需要了解其它无关的内容；
2. 基于 MOF 的各种技术（如 XMI、JMI）等，可以应用于 UML 模型，包括各种使用各种 UML profile 定义的模型上。
3. 可以直接使用各种已有的 UML 建模工具进行元数据的建模。

在【OMG02】的“UML Profile for MOF”一节中，专门列出了 UML 模型子集和 MOF 元模型之间一对一的映射关系。从长远来看，OMG 有把这两者融合在一起的趋势，这也是符合 MOF 的定义初衷的。

## 参考文献

【Joaq03】《MDA Guide Version 1.0》，Joaquin Miller，Jishnu Mukerji，2003.5

【Davi03】《Model Driven Architecture：Applying MDA To Enterprise Computing》，David S. Frankel，Wiley Publishing，2003

【OMG97】http://www.omg.org/news/pr97/umlpr.html

【OMG02】《Meta Object Facility (MOF) Specification》，Version 1.4，April 2002

【Cris99】《UML2001：A Standardization Odyssey》Cris Kobyn，1999-10

【Jean03】《New Trends in Applied Model Engineering》，Jean B ezivin，Olivier Gerbe，2003，http://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/OFTA/new.trends.pdf

【OMG03】《3rd revised submission to OMG RFP ad/00-09-01:Unified Modeling Language: Infrastructure》，version 2.0，2003-03-01

The Addison-Wesley Signature Series

# Patterns of Enterprise Application Architecture

中译本即将上市

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

UMLChina 负责中译本最后审校，并指定为训练用教材

# 使用模型驱动架构方法在 J2EE 平台上进行模型驱动开发

MIDDLEWARE 著，陈龙 译

查看评论

## 1 执行摘要

最近，对象管理组织(Object Management Group, OMG)发布了一种新的服务器端软件开发的思维模式，即模型驱动架构(MDA)。MDA 不同于以往传统开发方法之处在于使用 MDA 你首先要用 UML 构建对象模型，然后通过代码生成工具和模式库(pattern repository)从对象模型生成代码。OMG 认为 MDA 有许多优点，其中最令人兴奋的就是开发人员生产率的提高。

这一案例研究的目的就是验证或反驳关于基于 MDA 的开发工具可以提高软件开发生产效率的声明。两个团队开发同样的 J2EE PetStore 应用，一个团队使用了基于 MDA 的开发工具，而另一个团队使用传统的企业级集成开发环境(IDE)以代码为中心(code-centric)的开发方式开发。

研究的结果是，MDA 团队的开发速度比传统的团队快 35%。MDA 团队在 330 小时内就完成了开发任务，而相比之下传统的 IDE 团队用了 507.5 小时。作为这一案例研究的结论，The Middleware 公司推荐那些致力于提高开发效率的项目组可以在他们的项目中尝试使用基于 MDA 的开发工具。

## 2 介绍

通过这一案例研究，你将了解到 MDA 开发方式的概况，并且能够看到一个接受这种开发方式的团队获得了什么样的生产率。如果你正在寻求如何使项目中的开发人员的生产率最大化，这份案例研究是你必读的。

### 2.1 什么是模型驱动架构(MDA)?

模型驱动架构是一种旨在使业务逻辑和应用逻辑与技术进展相分离的软件开发思维模式。MDA 可以帮助你快速构建出与中间件无关的，具有优良架构，坚固的，可维护的代码。

MDA 开发方式的关键点在于以下这几步开发过程：

1. 确定应用程序的业务需求。

2. 画出业务模型的 UML 图，这些图与任何具体实现它的开发技术(J2EE,.Net,CORBA 等)都无关。这个 UML 模型代表了核心的业务服务和组件，例如可以有定价引擎，购物车，订单处理等模块。因为它完全和技术无关，所以这个 UML 模型叫做平台无关模型(Platform-Independent Model, PIM)。也就是说，不管你使用 J2EE 还是.Net，这个 UML 模型都是一样的。你可以用一个 MDA 专用建模工具来建造这种 UML 模型。

3. 画出应用程序的 UML 图，这些图指定与某种技术(比如说 J2EE)相关。这个 UML 模型包含特定技术的元素，比如说特殊的 J2EE 设计模式。这个 UML 模型叫做平台相关模型(Platform-Specific Model, PSM)。你可以手工建模，也可以用一个 MDA 工具生成它的大部分，然后手工调试需要特制的部分。

4. 最后，用一个 MDA 工具生成应用程序代码。也就是说，不是基于 UML 模型手工编写程序，而是从 UML 图***生成*** 程序的大部分。如果是 J2EE,MDA 工具可以生成绝大部分 servlet, JSP 和 EJB。你剩下的工作就是填补上 UML 无法建模的细节，比如业务逻辑。

### 2.2 谁支持 MDA?

对象管理组织(OMG)创造了模型驱动架构。OMG 是一个业界标准组织，有几百个组织成员，包括 IT 用户和开发商。OMG 是几种被广泛使用的标准的管家，包括统一建模语言(UML)和通用对象请求代理架构(CORBA)。因为用 OMG 的开放过程制定，所以 MDA 是一种中立于开发商的方法，任何厂商都可以创建一个用于辅助 MDA 开发过程的 MDA 工具。关于 MDA 的更多信息请参考 OMG 关于这一主题的白皮书，位于 http://www.omg.org。

### 2.3 MDA 与传统的开发方法有何不同?

我们已经多次提到，MDA 让你从 UML 模型生成代码。这是可以做到的，而且在 MDA 之前就可以做到。比如说 Rational Rose 就可以从 UML 模型生成 Java 类文件。然而 MDA 的关键进步之处在于它可以指导你从高层的与平台无关的设计一直到相当完整的具体平台的代码的整个开发过程。值得注意的几点是：

*MDA 从比其他设计方法更高的抽象层次开始。顶层的模型(PIM)非常抽象，仅仅是实体和服务。

*PSM 用元数据方式对应用程序的完整描述。在这一层次，你可以用特定技术的特性(比如 EJB 实体 bean 中的定制查找方法)来补充设计，而不接触 Java 代码。

*从 PSM 生成的代码已经接近完工了。有很多工具(比如，Middlegen 或 XDoclet)可以从某种模型生成代码，但是这些工具生成的代码只是应用程序的一些片断。因为它们没有从整体模型出发，所以他们不是全面的。

*从 PIM 生成 PSM 和从 PSM 生成代码的算法可以由架构师配置。

### 2.4 使用 MDA 有什么好处?

一起联合打造 MDA 的组织相信它会带来如下好处：

**更快的开发速度。**生成代码而不是手写每个文件可以使你节省一次次的重写同一个文件的时间。比如说 J2EE 开发，有时为了一个 EJB 组件需要写六个或更多文件。而这些工作的绝大部分可以由一个聪明的代码生成工具自动完成。

**架构的优势。**使用 MDA 时，用 UML 为你的系统建模，不仅仅是给 Java 类建模而是给高层次的问题域实体建模。这个过程强迫你切实地考虑支持系统的架构和对象模型而不仅仅是投入到编码中。软件工程的原则已经证实在软件开发生命周期中先作系统设计可以减少在后面的开发过程中把架构性缺陷引入系统的可能。

**改善了代码的一致性和可维护性。**许多组织在他们的项目中都存在着保持应用程序的架构和代码一致性的问题。有些开发人员使用了广为接受的设计模式，而其他人不用。用 MDA 工具生成代码而不是手写代码可以使代码按照协调一致的算法生成，也就给所有的开发人员优先使用设计模式的机会。从易于维护的观点来看，这是一个显著的优点。例如，在一个开发组织中接受 MDA 开发方法的开发人员能够很容易地理解其他人开发的代码，因为他们使用同样的设计模式和语言。

**增加了在各种中间件之间的移植性。**如果你需要在不同的中间件平台上迁移(例如，在 J2EE，.Net 或 CORBA 之间迁移)，平台无关模型(PIM)是可重用的。从 PIM 可以生成新的目标平台的代码。虽然不是所有的代码都可以自动生成，但是生成一个应用的大部分的能力肯定可以比从无到有手写代码节省时间。(注：The Middleware 公司认为软件组织可能不会中立地使用 CORBA，J2EE 和.NET。然而，这一点对于一个独立软件提供商来说是有用的，独立软件提供商销售可以支持各种 J2EE 应用服务器的产品，这样他们就可以把软件卖给任何一个 J2EE 用户)。

在这次案例研究中我们关注的是评估*更快的开发速度*。我们计划在将来的案例研究中再评估其他的优势，比如易于维护和改善质量。

# 3 研究说明

为了考察使用 MDA 可以具有更快的开发速度的承诺，Compuware 公司委托 The Middleware 公司执行这次案例研究——两个团队开发同一个应用，一个使用 MDA，另一个不用。The Middleware 公司被选中作这次案例研究主要是因为该公司是厂商中立的，并且对研究的结果没有任何偏见。The Middleware 公司向你保证以公正和公平的方式执行这次研究。

在这次研究中我们不会提及团队成员使用的开发工具的名字。为了保证以代码为中心的方法的正常表现，我们可以证实这个团队使用了一种市场上领先的 IDE。我们想让这次研究在 MDA 方法和传统方法的对比评估上具有教育意义。我们不想让这一研究转变为一场开发商之间的混战。

## 3.1 关于规范

每当执行一次类似这样的案例研究，都存在因团队间的差异，建造出的应用程序无法对比的危险。为了避免这一问题，我们为这个 J2EE PetStore 写了一篇 46 页的规范。规范描述了应用从数据库到网页用户界面的详细需求。你可以从 http://www.middlewate-company.com/casestudy 网站下载这个规范。

这次案例研究使用的功能规范叫 *The Middleware 公司应用程序服务器平台基线规范*。这个规范是 The Middleware 公司在一个卓越的专家团的帮助下历时两三个月独立完成的。这个专家团包括图书作者，开源项目的参与者，Engineering 的 CTO 和 VP，代表了最顶尖的三家美国 IT 研究公司中的两家，还有以前对 The Middleware 公司做的案例研究的批评者。专家组成员包括：

Assaf Arkin (Chief Architect, Intalio), Clinton Begin (Author, Open Source JPetStore), Rob Castaneda (Author, CEO, CustomWare Asia Pacific), Salil Deshpande (The Middleware company), William Edwards (The Middleware Company), Marc-Antoine Garrigue (OCTO, lecturer ENSTA), John Goodson (VP, DataDirect), Erik Hatcher (Author, Java Development with Ant), Rod Johnson (Author,Expert 1-on-1: J2EE Design & Development), Anne Thomas Manes (Analyst, CEO, Bowlight), Vince Massol (Author, JUnit in Action), John Meyer (J2EE/.NET Analyst, Giga Information Group, now Forrester Research), Tom Murphy (J2EE/.NET Analyst, META Group), Cameron Purdy (CEO,Tangosol), Roger Sessions (.NET Guru, Founder, ObjectWatch), Vivek Singhal (CTO & VP Engr,Persistence Software), Bruce Tate (Author, Bitter Java), Bruno Vincent (OCTO), Andrew Watson (Vice President & Technical Director, Object Management Group), Wayne Williams (CTO, Embarcadero Software), Joe Zuffoletto (Author, BEA WebLogic Server Bible)

规范被设计成不仅能做效率研究之用，还能够用在其他的研究中：架构和设计，性能，互操作性，或其他。

### 3.2 应用程序的选择

我们决定为这次对比研究选择大家熟悉的 J2EE PetStore 应用程序。虽然我们知道 PetStore 有它的有利和不利之处，我们选择它主要是因为它是一个大家都熟悉的简单的应用程序。对于我们来说这种熟悉度是很重要的，因为我们想让大家都容易地理解这个案例研究。

对于你们当中可能不熟悉 PetStore 的人来说，它是一个简单的基于 Web 的 J2EE 电子商务系统。PetStore 有如下功能：

**用户管理和安全。**用户可以登录系统，也可以管理他们的账户。

**产品目录。**用户可以在网站上浏览宠物的目录(比如鸟，鱼或爬行动物)。

**购物车功能。**用户可以把宠物加入购物车，也可以管理购物车。

**订单功能。**用户可以为他们购物车里的货物下订单。

**Web 服务。**用户可通过 Web 服务查询订单。我们引入这项技术来扩展 PetStore 是因为 Web 服务是让人感兴趣的新兴领域。

从技术的角度来看，PetStore 包括如下功能：

*一个 HTML 用户界面的瘦客户端层

* JSP 在服务器端生成 HTML

*基于 JDBC 的 SQL 数据访问

*EJB 中间层组件

*Ad-hoc 的数据库查询

*数据库事务

*数据缓存

*User/Web 会话管理

*Web 服务

*基于表单的身份验证

值得注意的是"PetStore"触发了很多人的复杂情感，因为 SUN 微系统公司从来没有打算把 PetStore 样例程序用作一个案例研究的基础。毕竟，PetStore 起初只是一个 J2EE 的样例程序，不是一个成熟的规范。此外，起初的 PetStore 并不代表一个架构优良的应用程序。

我们相信我们通过以下方法解决了这些问题：

*我们现在有了一个 PetStore 的规范，而不仅仅是一个实现。

*符合我们的规范的“现代”版的 PetStore 实现已经完全不同于 SUN 的原始的实现了。实际上该规范和 SUN 原始的 PetStore 的最大相同点只是应用领域都与购买宠物有关。

*规范没有要求使用任何特定的架构性方法去设计，而是给了两个团队架构自己应用的自由。

### 3.3 基本规则

规范详细描述了构建应用程序的规则。比如规范描述了什么数据可以被缓存，数据库排他规则(database exclusivity rules)，基于表单的身份验证的需求和会话状态规则。规范也非常详细地描述了应用程序必要的使用体验。

我们给每个团队同样的HTML和图片。此外，要求每个团队使用同样的数据库表结构。我们决定这样做是因为我们想测量J2EE编码的效率而不是创建数据库或图片制作的效率。我们也想让用户界面尽量保持一致。

我们要求两个团队必须使用EJB，除此之外没有要求具体的实现细节，包括选择J2EE实现模式等细节都留给团队自己决定。对于我们来说，两个团队的最终成果要类似才是要紧的。

另外，每个团队都可以选择一种J2EE兼容的应用服务器。安排这样做是想让每个团队使用自己最熟悉的应用服务器，最大程度的发挥他们的生产率。毕竟这是一个生产率的案例研究。

我们不会提及具体使用的应用服务器的产品名称。因为没有邀请他们参与这次案例研究，所以提及这些应用服务器的名称对于厂商来说是不公平的。

每个团队也可以选择自己的版本控制工具和其他一些辅助的工具，比如日志工具或文档工具。我们给每个团队最大程度的自由以发挥他们的生产效率。两个团队选择如下的工具：

| 工具 | 用途 | 注释 |
| --- | --- | --- |
| StarTeam | 版本控制 | StarTeam被认为是领先的版本控制系统。在我们The Middleware公司内部就被广泛使用。 |
| Apache Axis | Web服务 | Apache Axis和SUN的Java API for XML Parsing (JAXP)兼容。 |
| Apache Log4J | 日志和审核 | Apache Log4J正快速地成为J2EE项目中标准的日志包。 |

请注意，我们严格要求MDA团队选择一种完全符合MDA的工具。对于传统的开发团队，我们要求他们使用一个领先的J2EE IDE。我们这样做是因为这次研究的要点是比较不同的开发方式，因此每个团队就应该使用符合这种方式的工具。

### 3.4 测试过程概况

每个团队在提交他们的最终代码库之前都进行了单元测试。另外为了确保最终的应用程序运行的一样，我们让每个应用程序都通过了严格的测试过程。整个测试过程包括37个测试用例，这些测试用例我们在每个应用上都人工测试过。测试用例把应用程序的运行方式和原始规范描述的运行方式进行了对比。实际上这些测试用例都是功能性的，测试用例没有执行代码的单元测试而是描述了一个测试者通过Web界面和应用交互的过程。例如，其中一个测试过程是登录系统然后编辑账户信息。两个团队的应用都通过了测试。

我们还用了一种包结构分析工具(可以在http://javacentral.compuware.com/pasta下载)测试了双方的代码的非循环依赖性(acyclic dependencies)。两个团队在非循环依赖性测试中都得了90%的高分。

最后，因为我们知道很多代码生成工具能够产生“坏代码”，所以我们检查了双方生成的代码。根据我们的检查，我们相信生成的代码质量很好。

### 3.5 团队概况

我们竭尽全力确保两个团队具有大体相近的技术集(skill-set)，不会让一个团队比另一个具有无法比拟的优势。每一个团队成员都具有在多种应用服务器上进行J2EE开发的丰富经验。此外，为了平衡技术集之间的差异，我们花了很多时间把开发人员按照他们愿意使用的工具和技术做了调整。两个团队还解决了一些小的编队问题。这些时间不会被计算在最终的生产率数据中。

每个团队包含3名成员：

**一名高级J2EE架构师。**这个人掌握自己开发环境的技术细节。他的职责是开发，做架构，合理地分配任务，最终实现规范。

**两名经验丰富的J2EE程序员。**这些人都具备至少3年的J2EE应用程序开发经验。

### 3.6 项目计划和管理概况

为了准确地记录每个团队工作日志，我们分别和每个团队每周召开例行电话会议。在电话里我们对两个团队每周的经历作了详尽的记录。两个团队都必须回答下列问题：

*你的团队这周作了什么？

*从效率的观点看，这周好的方面有哪些？

*从效率的观点看，这周哪些方面存在问题？

这些记录的摘要会在这份案例研究的下一节作介绍。

在项目开始之前我们举行了一个项目开工仪式，仪式上每个团队都估算了他们多长时间能够完成项目。另外，每个团队每周都要提交上周的详细时间表。时间表包括估算时间和实际使用时间的对比。

## 4 研究结果

在这一部分我们将论述案例研究的结果。结果被分为如下几个部分：

**在架构分析部分**你将了解每个团队使用的架构和J2EE模式。我们在文档后面创建了一个*专用设计模式术语表*来简要地解释每种模式。对于有抱负的J2EE架构师来说这个术语表将是一份有趣的阅读材料。如果你不熟悉J2EE模式的话，你最好在阅读架构分析部分之前先阅读这个术语表。

**在定性分析部分**你将了解到每个团队对他们选择的开发方法的感性的思考。你将了解每个团队遇到的问题和他们是怎样克服问题的。

**在定量分析部分**你将会看到每个团队最终的生产率数字。

## 4.1 架构分析

### 4.1.1 UML和代码生成

两个团队都创建了对象模型的UML图。而其实两个团队创建的的对象模型非常相似。每个团队都作了如下抽象：

*用户账号

*用户资料信息

*产品

*产品目录

*供应商

*购物车

*订单

*排列项

传统团队用了一种开源工具为他们的对象模型创建UML图。他们确实是为了设计和沟通的目的画的UML图，而不是为了从UML自动生成J2EE代码。不过，他们确实用了他们IDE的向导功能生成了JavaBean的accessor/mutator方法(也就是geter/seter方法)，EJB组件，struts代码，异常处理代码和加速实现接口的桩代码(stub-code)。

MDA团队用他们的MDA工具不仅创建了平台无关模型还创建了平台相关模型。他们用MDA工具的UML代码生成功能自动生成了比传统开发团队多的代码。生成的代码有JSP，EJB组件，Struts代码，异常处理代码和接口，还有许多设计模式和构建应用程序时用到的框架代码。不仅仅是框架代码，MDA工具还在代码生成模式(code-generation patterns)的基础上利用UML对象模型生成了一个可以运行的应用程序。

### 4.1.2 Web层

两个团队在Web层都使用了servlet，Java Server Page，和JSP标记库(taglibs)的组合。两个团队都用了*Apache Jakarta Struts*作为他们Web层开发框架。Struts是一个流行的建造基于J2EE的Web应用的开源框架。它鼓励使用广为接受的Model-View-Controller(MVC)设计模式。两个团队都调整了生成的代码使其更符合MVC思维模式。

#### 4.1.3 EJB层

从架构性的观点来看，两个团队的EJB层代码的具体不同之处在于他们选择的模式。

| 模式 | 传统团队 | MDA团队 |
|---|---|---|
| Session-entity wrapper | Yes | Yes |
| Primary Key generation in EJB component | Yes | Yes |
| Business delegate | Yes | Yes |
| Data Transfer Objects(DTOs) | Yes | Yes |
| Custom DTOs | Yes | Yes |
| DTO Factory | Yes | No |
| Service Locator | Yes | No |
| JDBC for Reading via Data Access Objects(DAOs) | Yes | No |
| Business interface | Yes | No |
| Model driven architecture | No | Yes |

从上表可以看出，两个团队都用了很多J2EE设计模式。虽然这样做在开发PetStore这样的应用时显得大材小用，但是我们还是想使用在实际项目开发中用到的模式，这样就更能够测试出MDA生产率的真实水平。

另外还请注意，传统开发团队使用了比MDA团队更多的模式。当我们一开始做好这个比较表时就注意到了这一点。传统开发团队使用了额外的模式是否会使这次案例研究失效呢？毕竟，传统的团队使用更多的设计模式会让人认为他们的开发过程会自然地被拉长。

为了回答这个问题，我们仔细分析了被使用的每个模式。我们也会见了两个团队的成员。完成这次研究后，我们关于这一问题的一致意见是虽然传统团队使用了更多的模式，但是比较结果还是非常正确的。为了帮你理解为什么是这样，让我们看看每个传统团队使用了而MDA团队没有使用的模式吧：

**The service locator 模式** 没有给传统团队添加额外工作。实际上这个模式还节省了他们的时间，因为它简化了EJB层的客户端代码编写，也包括EJB组件调用其它EJB组件的情况。由于是使用一个类中的静态方法实现的，所以它减少了必要的代码行数。所有对EJB的访问都是由这些方法完成的。

**The JDBC for Reading via DAOs** 确实需要传统团队花时间去实现；然而，如果假使他们不用这个模式的话，他们就不得不写实体Bean来替代这些数据访问对象(DAO)了。还有，MDA团队需要手工编写代码来排序数据，而这些工作传统团队可以通过JDBC for Reading模式完成。所以说，这种模式实质上没有给传统团队增加任何工作量。

**The business interface pattern** 没有给传统团队带来任何工作量。据他们讲，每个接口大概需要10分钟。因为要使用MDA推荐的平台无关模型创建业务对象模型，所以MDA团队也要创建业务接口，只不过层次更高而已。

**The DTO Factory pattern**花了传统团队一点实现和测试的时间。但是由于这种模式减少了应用程序中的代码量，所以实际上还是节省了他们的时间。

同样值得注意的是传统团队拥有世界顶级的J2EE模式专家(Owen Taylor,我们J2EE模式教程的作者)。这种专业的模式知识不是普通的IT开发组织所能拥有的。所以传统的团队使用模式比MDA团队多就可以理解了。正是因为传统的开发团队选择的模式没有拖延他们开发的时间，所以我们得出的结论是没有明显的迹象表明这次生产率比较是无效的。

#### 4.1.4 安全

两个团队开发的应用程序大部分都是相近的。不过两个团队在安全方面却产生了差异。

传统团队使用了一个身份验证过滤器(authentication filter)来解决安全问题。这个过滤器察看当前请求的页面是否是受限制的，如果是的话就要看用户是否已经登录了(登录信息保存在HTTP会话里)。这个过滤器完全是这个团队自己写的。过滤器实现javax.servlet.Filter接口。

相比之下，MDA团队使用了MDA工具辅助开发安全系统。他们的MDA工具提供了一个安全框架可以使他们从不同的身份验证方法中选择，这些方法其中包括简单验证，基于摘要的，基于表单的，和编程实现的验证。该组选择了编程的方法。这一框架还提供了登录和退出的JSP页面，该组定制了这些页面把他们用在了PetStore中。

## 4.2 定性分析结果

在这部分，我们将评论两个团队感性的思考，这些是从他们每周提交的资料中总结出的。

### 4.2.1 传统团队

#### 4.2.1.1 传统团队第一周

传统团队采用迭代式的原型法开发。在第一周里，他们搭建了一个简单的原型，这样就可以架构他们项目的设计模式基础。对于他们来说建立一个得心应手的开发环境是一件相当棘手的问题。在让StarTeam版本控制和他们的IDE有效结合的过程中就遇到了一些问题。在第二周他们才下决心用替代方法绕过这一问题。第一周他们还花了很多时间在架构对象模型上，比如确定包结构等和开发环境有关的问题。

从生产率的观点来看，第一周他们印象最深的就是IDE的能力。他们发现 IDE和被选择的应用服务器集成得很好，而且也具有很好的代码生成能力。这都得益于每个团队成员都有以前使用这种IDE的经验。

这周他们生产率存在的问题和文件共享，沟通交流，执行代码复查，维护包结构的稳定性有关。

#### 4.2.1.2 传统团队第二周

第二周，传统团队开始分配不同的开发角色。一名队员负责model(EJB)层，另一个队员负责View(web)层，第三个队员提供两个层之间的辅助类和接口。

该队决定以用例的方式构建他们的系统，一次只关注一个用例。这一周他们在开发用户账户维护用例，例如登录，退出，用户创建和用户偏好维护。

从生产率的观点来看，这周他们发现Struts框架发挥了作用。他们发现编写Struts代码的效率很高。另外他们还决定使用一种让web层和业务层解耦的策略。这种策略让实现“桩”的代码能够植入没有完全建好的代码中。他们通过属性文件和附录中的业务代理(business delegate)设计模式在“桩”和“真实”代码之间切换。

他们这周的生产率问题都是一些开发团队在开始会遇到的典型问题，例如坚持使用源代码控制。在IDE的使用上也遇到了困难。情况是，如果他们试图从IDE生成J2EE组件然后修改的话，这些修改过的组件就很难再逆向到IDE中了。

#### 4.2.1.3 传统团队第三周

这周，传统团队继续开发他们的产品目录浏览和购物车等功能。

从生产率的观点来看，这周他们解决了如何使用版本控制协作开发的问题。他们还构建了队员用到的接口库，这样就减少了构建时(build-time)的依赖性。在开发的这一阶段，每个队员的效率都非常高，而且由于项目模块分解的非常合理，每个队员都有自己的工作领域。还有的就是他们的IDE在代码生成上也起了很大的作用。

他们这周的生产率的唯一问题就是对PetStore规范描述的需求的理解和执行了一些对购物车部分的重构。

#### 4.2.1.4 传统团队第四周和第五周

在最后两周里，传统团队在对最后一个用例编码，测试和调试后完成了他们的应用程序。他们还合并了队员的代码，去掉了实现桩的代码。

这周获得高效率的因素有这么几点。他们始终得益于选择的桩实现架构。只要他们把接口确定下来就基本不需要什么沟通了，所以编码的效率非常高。同时，因为他们的IDE在WSDL自动生成和SOAP服务部署方面的能力，他们的web服务实现构建得也非常快。

这周遇到的问题包括对代码的重构和重新分析，例如在数据访问层就存在一些缺陷。由于他们的应用分成好多层也导致了一些问题，因为当出现错误时除非仔细查看这些层否则就不清楚问题出在哪一层。他们还有其他一些无足轻重的问题和Struts和属性文件有关。

### 4.2.2 MDA团队

#### 4.2.2.1 MDA团队第一周

第一周，MDA团队作的都是一些一般项目团队在项目开始时做的典型步骤。他们用MDA工具的UML建模能力创建了一个详细的对象模型。配制好了他们的源代码控制系统，然后就把项目任务分配给每个队员。他们还设置了一个项目目录结构并作了一些初始的开发工作。

从生产率的观点来看，好的方面就是他们发现不需要建造过多的组件，因为MDA工具具备生成这些代码的能力。实际上他们还发现工具不仅能够生成这些组件，还能生成框架代码把这些组件连接起来产生一个有机的应用程序。该团队估计应用程序的50%到90%都可以自动生成，包括web页面，Struts action，EJB实体bean，EJB会话bean，J2EE设计模式的框架代码和数据库表。

这周的关键问题是掌握MDA方法的核心思想。这种方法和他们以前用的传统方法不一样，他们需要抓紧时间来熟悉工具提供的许多层和结构。

#### 4.2.2.2 MDA团队第二周

第二周，MDA团队开始投入到开发工作中。他们完成了网站的很多部分，包括UML对象模型，组件框架，站点导航系统，JSP模版(用Dreamweaver开发)，主页，EJB购物车的大部分功能和用户账户管理的一小部分。

从生产率的观点来看，好的方面是他们能够从UML对象模型生成EJB会话/实体bean代码，web层代码，和一个DDL模型。创建测试数据集也同样非常简单。最后，由MDA工具生成的整体应用程序框架可以使他们快速地把组件组装起来。

这周遇到的困难和传统团队遇到的差不多。他们有一些开发环境上的困难，包括在一个协作的开发环境中习惯使用源代码控制系统。他们花了一点点时间就把工具配制好了。他们也知道着手使用MDA工具需要时间来学习，所以在这周里他们都在不停地学习以适应从传统的开发方法转变到MDA的思维模式上来。最后，他们意识到MDA的代码生成能力有点过于强大了，生成了一些他们不需要的代码。他们确实注意到有必要修改工具用于生成代码的算法。在一个实际的项目中这样做可能会减少麻烦，但是这超出了这次案例研究的范围。

#### 4.2.2.3 MDA团队第三周

第三周，MDA团队完成的系统用例比上周多了几个，包括购物车，订单处理，账户管理，登录模块。

从效率的角度来看，这周有几件好事。由于MDA工具辅助生成了代码，他们非常容易就构建好了安全系统。他们使用JSP模版的想法也起了作用。把JSP的通用代码集中在JSP模版中，这些通用的代码就可以集中在一个地方更新了。

这周有一个开发人员开始体会MDA方法的好处，并且情不自禁地做出了如下结论：

*MDA的价值和OO的价值相似，只要在设计阶段多下功夫，在实现阶段就能得到回报了。一旦你做了一两个应用程序之后，你就会真正开始从它受益。*

不过这周他们也遇到了一些困难。在Struts上遇到的一些细节问题一度造成他们工作停顿。他们还有一些会话标识符和浏览器的cookies的问题。这周最后遇到的问题是关于MDA的。他们发现由于在*认为应该*生成的代码和*实际*生成的代码之间存在差距，有时会感到开发工作非常郁闷。有时他们的开发工具要么产生他们并不需要的代码要么产生过剩的代码，这些都无助于提高效率。他们认为这些问题都是由于他们在MDA和代码生成上的经验不足

造成的，因为他们在预测将会产生出什么代码时毫无经验。他们相信随着使用MDA工具的经验的增加这些问题将迎刃而解，而且是用得越多，效率就越高。正是由于对MDA工具一时的不熟悉，导致他们的时间有点拖延。

#### 4.2.2.4 MDA团队第四周

第四周，MDA团队完成了他们的开发、测试和调试任务。

从提高效率的角度来看，MDA代码生成，以及由MDA提供的Struts框架，身份验证框架都是值得肯定的。同时，模块集成过程也进展顺利。

这周遇到的问题是有时生成的代码变成了一种负担。他们要么彻底修改，要么丢弃不用。不过他们也认识到MDA工具生成的代码是基于模式的，这就给他们手写代码助了一臂之力。他们也有一些MDA工具的源代码合并问题。他们吸取的教训是当你使用一种象MDA工具一样的产品时，你必须随时掌握工作的进展，并且要注意如何使用好版本控制把他们编制到一起。

### 4.3 定量统计结果

每个团队最终使用的开发小时数统计结果：

| 团队 | 初始估算小时数 | 实际的小时数 |
|---|---|---|
| 传统团队 | 499 | 507.5 |
| MDA团队 | 442 | 330 |

就像你看到的，在这次案例研究中MDA团队取得了绝对性的胜利。他们在预算的时间之内完成了任务。需要强调的是MD团队不是立刻就利用了这种效率上的优势，而是随着项目的展开效率的优势才逐渐显现出来。队员把这归咎于MDA工具，因为它对于绝大多数开发人员来说都是一种新的产品，所以需要一个适应的过程。实际上有一个队员认为作为他们用这种工具开发的第一个项目，如果不存在这个学习的过程的话他们应该比实际的情况还要再快10-20%。

另一点也需要强调，在这个项目里我们没有执行“bug跟踪”；不过我们作了前文提到的手动用例测试。有意思的一点是在对传统团队的测试过程中发现了一些bug,但是MDA团队却没有。我们相信这是由于使用MDA开发方式产生的代码的坚固性造成的。

## 5 结论

根据这次案例研究的结果，The Middleware公司被MDA团队使用模型驱动架构所获得的效率所折服。我们鼓励那些致力于提高开发效率，特别是那些涉及企业级应用和Web服务的组织，在项目中尝试使用基于MDA的软件开发工具。当然有必要给开发团队短期的MDA方法和开发工具入门时间，由于在开发的过程中，特别是以后的应用中才能收到效率的回报，所以付出的努力完全值得。

我们认为MDA还有其他的一些价值。比如，平台无关模型(PIM)具有相当长的预期使用期限。软件开发组织有必要创建一个这样比技术生命周期还长的业务对象模型。和使用不同的工具和技术的开发人员交流时，这将是一个非常重要的工具。

我们觉得MDA的另外一个优点就是可以让一个组织中具有渊博知识的架构师确保平均水平的开发人员能够协调一致地使用设计模式。用MDA工具自动生成实现模式的代码，然后按照架构师的意图作调整，架构师就可以做到这一点。这样，使用模式就很自然地变成了开发人员编码工作的一部分。

我们有一个队员是这样评价MDA的：“它让一名脑外科医生成为一名更好的脑外科医生，但是不会让一个看大门的人成为脑外科专家。”他的意思是MDA让一个有经验的架构师的能力得以充分发挥，能够更好地把握项目的方向。同时你还要你的职员了解J2EE模式和最佳实践，面向对象的开发，和一些架构方面的考虑。

最后，在代码和应用程序质量方面，一个很突出的观测结果就是在手工编写代码时发现的bug需要修改和复测，而修改和复测还将不断影响整体开发质量。MDA团队通过模版自动生成代码就不会有这些额外的步骤了。

有必要指出，这次案例研究中很多MDA的特性没有被评估到，比如应用程序的性能和可维护性。我们确实也作了一些必要的性能测试，用来考查两个应用的性能是否相近。但是为了得到全面客观的结果，我们觉得可以把这些测试放到今后的研究中。同样，我们想在可维护性方面进行比较也将是非常有意思的，你将看到相比一个用传统方法开发出的系统，改变MDA建造的系统的代码将是多么容易的事。毕竟，当你重构一个基于MDA的系统只需修改原来的UML模型然后再重新生成代码即可。

## 6 附录：设计模式词汇表（略）

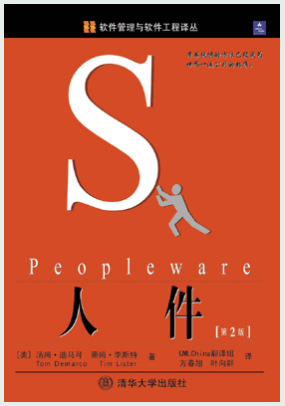
PH PTR

# 人件集

## ——人性化的软件开发

# The Peopleware Papers

NOTES ON THE HUMAN SIDE OF SOFTWARE




续篇

Larry L. Constantine 著

谢 超 刘 颖 谢卓凡 李 虎 译

人民邮电出版社

# 领会统一过程

Sinan Si Alhir 著，sunyanjia 译

查看评论

## 介绍

系统工程的工序将重点放在一种优雅的领域上，即如何从时间与空间的角度，使项目变化性和复杂性完美地相互作用。正是在这个舞台上，成员与团队的关键因素，方法，过程和工具以及所应用技术成为理想与现实之间的桥梁。在每一道成熟的核心工序中，从艺术到科学再到工程学，正是这种通用的语言和通用的步骤，使得那些实施者间的协作和工序的不断进化成为了一种可能。这种进化的核心是对知识的获取、沟通，共享和有效利用。通用语言建立了想法与行为的边界，它定义了概念；方法和过程则建立这种边界范围内的行为，它们应用这些概念；工具建立这种边界范围内的行为的自动化。显然，如果我们不能够理解它，我们就无法正确的使用它和交流它，更不能使其自动化。在信息系统和技术产业中，统一过程（UP），Rational 统一过程（RUP）,统一建模语言(UML)以及软件过程工程元模型（SPEM）是这种进化的核心。

## 统一过程（UP）与 Rational 统一过程（RUP）

统一过程（UP）是由用例驱动（Use-case-driven）、以架构为中心的、迭代增量的开发过程框架，它使用对象管理组织（OMG）的 UML 并与对象管理组织（OMG）的软件过程工程原模型（SPEM）兼容。统一过程（UP）广泛使用于各种软件系统，无论是小型项目，还是包含各种管理的级别、复杂技术或是跨越应用领域、跨组织文化的大型项目。

1995 年 Rational 软件公司并购 Objectory AB 公司，统一过程（UP）是由 Objectory AB'公司的 Objectory 过程和 Rational 软件公司的 Rational Approach 合并而成。其中 Rational Approach 是 Rational 软件公司通过对各种客户经验进行开发所得到的结果，而 Objectory 过程则主要来自 Ivar Jacobson 在瑞典爱立信（Ericsson）的工作经验。

在《统一软件开发过程》("The Unified Software Development Process")（由 Grady Booch, James Rumbaugh 和 Ivar Jacobson 共同编写，Addison-Wesley 出版社 1999 出版）一书中详细讲述了统一过程（UP）。Rational 统一过程（RUP）是由 Rational 软件公司开发并投向市场的一种过程产品，它为项目的执行提供了必需的细节：包括指南、模板以及辅助工具；事实上，Rational 统一过程（RUP）是为统一过程（UP）架构提供细节实现的商业化过程产品。

在一个应用统一过程（UP）或 Rational 统一过程（RUP）的项目中，开发案例（Development Case）、过程框架实例将决定哪些应用统一过程（UP）或 Rational 统一过程（RUP）的元素应被自始至终地使用。基于 Rational 统一过程（RUP）的开发案例是 Rational 统一过程（RUP）（和统一过程（UP））的实例，它通过对 Rational 统一过程（RUP）内容的剪裁（或是拓展）和配置以确定统一过程（UP）框架的广度和深度。而基于统一过程（UP）的开发案例是统一过程（UP）的实例，它确定统一过程（UP）框架的广度和深度。

## 统一建模语言（UML）与软件过程元模型

统一建模语言（UML）是一种通用的，有工具支持和广泛兼容特性的工业标准化模型语言，或是提供细化、可视化、构造以及生成过程中系统级文档制品的模型技术集合。它广泛的应用于各种类型的系统（软件或非软件），领域（业务与软件），方法和过程。统一建模语言（UML）使这种由用例驱动（Use-case-driven）、以架构为中心的、迭代增量的开发过程的应用成为了可能（但不是必须）。

上世纪九十年代,Rational 软件公司和三人组(Grady Booch, James Rumbaugh, and Ivar Jacobson)发起了对信息系统和技术工业的集成,统一建模语言（UML）就诞生于这次集成过程中。此后统一建模语言（UML）通过统一建模语言合作者联盟（UML Partners Consortium）得到了很多组织的工业支持,并于 1997 年 11 月被对象管理组织（OMG）采纳，成为标准。

正像统一建模语言（UML）成为系统交流的工业标准模型语言一样,软件过程元模型(SPEM)也成为了过程、过程框架（相关过程集合）之间交流的工业标准模型语言，但软件过程元模型(SPEM)并未描述过程的制定（项目中过程的计划和执行）。软件过程元模型(SPEM)在统一建模语言（UML）的影响下应运而生，它也得到了很多组织的工业支持并且于 2001 年 11 月成为对象管理组织（OMG）的标准。

## 系统开发，系统，模型，视图

系统开发生命周期包括宏观层次上的问题处理过程和微观层次上的科学方法。而问题的解决则包括：对问题的描述、说明以了解问题定义需求；对问题的进一步描述以衍生或细化出对于解决方案的描述，从而解决问题；对解决方案的描述与解决方案集合之间的映射以准确实现和构造系统从而满足需求。在解决问题的每一个步骤中，科学的方法包括：对假设的计划或预计，对假设的执行或经验性测试，依据结果对假设进行的评估。衍生出用来校正假设的结论。无论出自有意或是无意，这些宏观和微观层次的过程在系统的开发中得到了非常普遍的应用。

统一建模语言（UML）使解决问题过程成为可能，它简化了对问题和解决方案的理解、细化、可视化和文档化等工作，同时也有助于问题的获取、交流以及有效利用策略上和操作上的相关知识来解决当前问题。统一建模语言（UML）实现了问题的获取，问题的交流以及对相关系统使用模型、架构视图和图表的有效运用。

系统是由元素或单元有目的地组成的。系统架构需要两个与其相关的尺度：结构尺度和行为尺度。结构（或静态）尺度涉及到哪些组成系统的元素以及它们之间的关系。行为（或动态）尺度涉及到那些组成系统的元素之间为达到系统目标而进行的交互和协作，以及它们的功能与行为。

模型通过对那些为解决问题而获取知识（语义）的系统进行完整抽象得来。架构视图则是对那些负责组织和协调知识与富含经验的用法指南之间关系的模型进行抽象而得来的。图表是那些用来描述问题和解决方案的相关知识（句法）以达到交流目的的模型元素的图形投影。运用统一建模语言（UML）基本元素中的符号来描述概念，而用符号之间的连线来描述概念之间的关系。

## 方法和过程框架

软件是整个过程的项目业务的集合。项目的目标是提供解决方案。方法为如何管理项目给出了指导和建议。方法对问题和解决方案的所需的知识给出指导和建议。方法说明这些知识如何使用以解决问题。过程是方法在项目中的具体执行， 方法学是一组相关方法的理论，分类和有组织的集合，这些相关方法用来确定由谁在那些产品上进行什么样的活动，包括针对于不同类型的项目，上述活动应该在何时、如何、为什么以及在哪里进行。其中进行活动的人（谁），进行的活动（怎么样），生成的产品（什么）以及对他们的启发都是过程的元素。方法学将各种方法组织成一个集合，方法描述过程，过程在项目中执行方法。

由于运用单独的方法不足以解决不断变化的问题，而使用整套方法又不切合实际，需要为整套方法提供更多

的变通性和可测量性。运用整套方法的一部分叫做过程框架，被应用在一个特定工程的实际的方法子集被称作过程实例。过程框架指定或建议了由谁在那些产品上进行什么样的活动，包括针对于不同类型的项目，上述活动应该在何时、如何、为什么以及在哪里进行。而过程实例则就某个特定工程回答了上述的问题。过程框架中的过程实例被描述为一组可变性可测量性更强的一组相关过程集合，而过程实例则执行过程框架的一个子集。因此统一过程（UP）是过程框架，开发案例是过程实例。

## 统一过程（UP）

为了顺利而高效地应用统一过程。我们必须理解协作、环境和交互等概念。在统一过程中协作主要关注项目中的元素，环境主要关注项目的过程架构，交互主要关注项目的执行。图 1 中显示了统一过程中的各种元素。

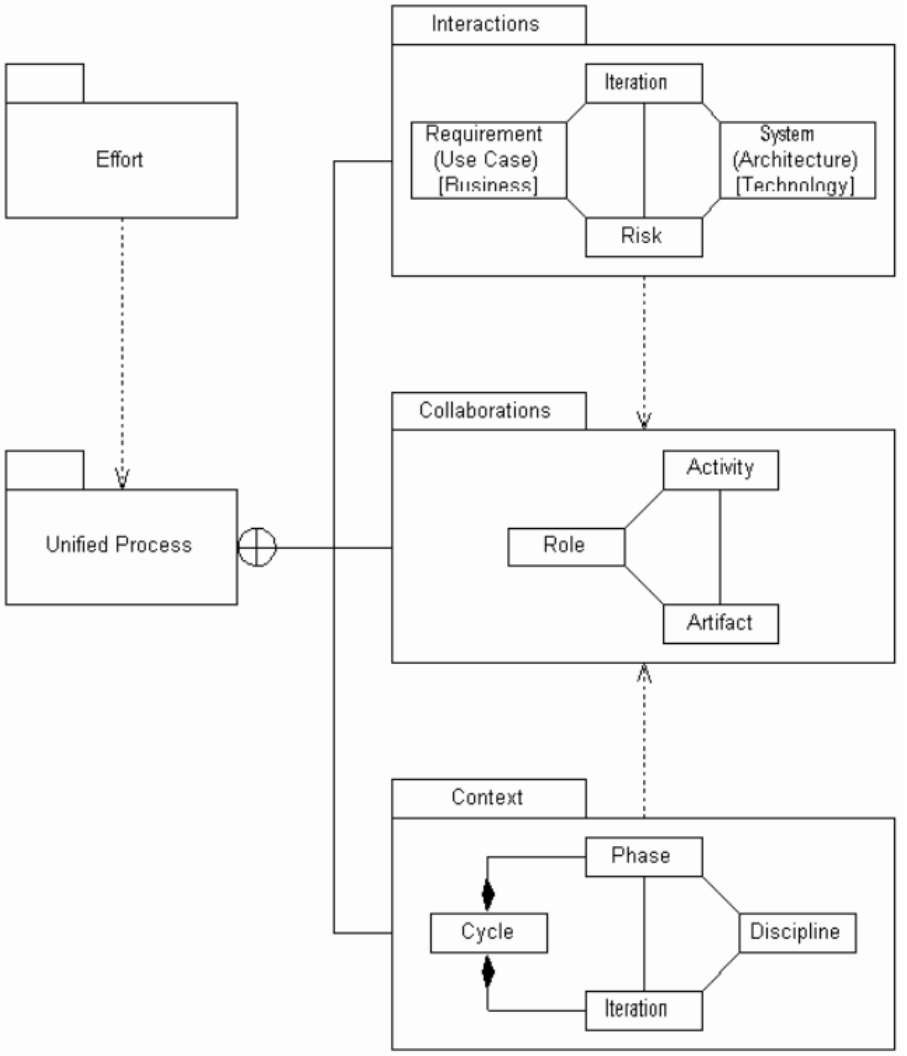



图 1 UP 的元素

## 协作

协作是环境中的交互。它获取了由谁在哪些产品上做什么样的工作。因此协作建立了项目中的元素。

角色是指进行项目活动和生成工件的个体或小组。活动是指由角色执行的工作单元或步骤。工件是角色职责以及由活动产生或消耗的信息元素。统一过程（UP）定义了多种角色、工件和活动。

## 环境

环境强调了协作的静态或结构表现，互相协作的元素和它们之间的空间聚集关系。通过环境可以得到何时何处进行那些产生和使用活动的工作。因此它为整个项目建立了一个环境。图 2 显示了统一过程（UP）的内容。

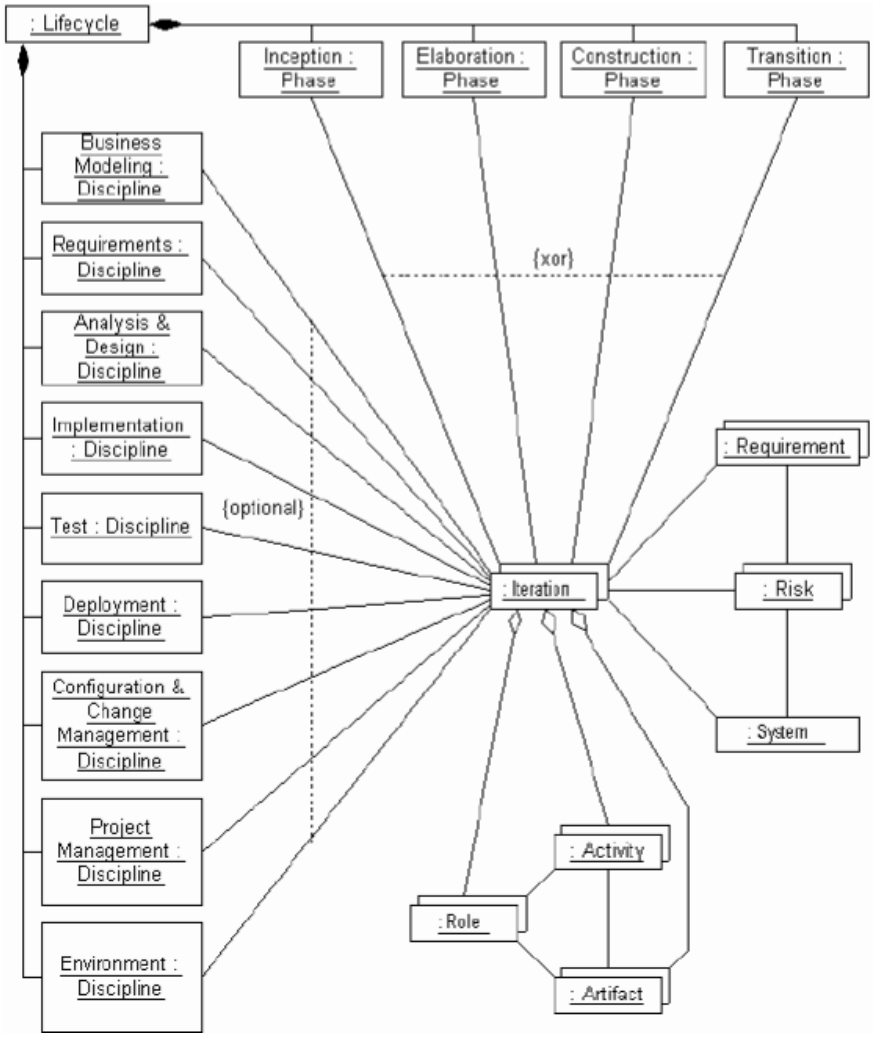



图 2 UP 建立的上下文

项目需要一个管理项目进展的管理透视图和一个执行并完成技术工作的技术透视图。项目的生命周期由许多满足规则的迭代阶段所组成。开发周期是由许多连续的阶段组成，这些连续阶段的最终结果就是主系统发布或称作系统生成。例如，系统的生成可能包括 1.1,2.0,3.0 和后续版本。阶段是一个主要里程碑，它是一个业务风险管理的决定因素。阶段包括那些宏观问题的解决过程；迭代作为辅助里程碑，作为技术风险管理的决定因素。迭代的结果是一个非最终版本系统的发布，叫做系统增量。例如系统增量可能包括 1.1,1.2,2.5 和后续版本。迭代包括了对科学方法的程序微观级别的描述。工序(discipline)是描述工作流和经常一起完成的详细活动（以及相关的角色和工件）集合的主题域。

统一过程定义了下列四个过程：

（1）先启阶段，以生命周期目标作为结束里程碑，建立项目的范围和版本；确定业务实现的可能性和项目目标的稳定性。

（2）精化阶段，以架构作为结束里程碑，建立系统的需求和架构；即确定技术实现的可行性和系统架构的稳定性。

（3）构建阶段,以最初操作性能作为结束里程碑，完成系统的构造和建立。

（4）产品化阶段，以产品发布作为结束里程碑，完成系统向用户群的产品化和部署。

统一过程（UP）定义了下列三个支持工序（discipline）：

（1）配置变更管理工序，用来管理系统和需求变更的配置。

（2）项目管理工序，用来管理项目。

（3）环境配置工序，用来配置项目的环境，包括所涉及到的过程和工具。

统一过程定义了下列六个核心工序：

（1）业务模型工序，通过业务模型获取相关知识以理解需要系统自动完成的业务。

（2）需求工序，通过用例模型获取相关知识以理解自动完成业务的系统需求。

（3）分析设计工序，通过分析/设计模型以分析需求，设计系统结构。

（4）实现工序，基于实现模型实现系统。

（5）测试工序，通过测试模型进行针对需求的系统测试。

（6）部署工序，通过部署模型部署系统。

各个阶段、迭代和工序的目的在于定位业务和技术上的风险。在先启阶段过程中，业务模型和需求工序将完成大部分工作。在精化阶段大部分的工作是在需求、分析设计和实现工序中完成。而构造阶段的主要任务则涉及到分析设计、实现和测试工序。在移交阶段则是测试和部署工序为主。支持工序贯穿作用于四个阶段。为了实现生成系统这一总体目标，所有的核心工序必须能够在不引入任何风险的前提下得到及时的使用；这就意味着，开发人员必须负责在需要使用这些工序的时候决定使用哪些核心工序。

## 交互

交互强调了协作的行为，动态表现，相互协作的元素以及它们之间的协作或暂时交流。交互获取了何时、为什么进行这些活动，产生和使用活动的工作。因此它在各种因素的支配下建立项目的执行。

正如辅助里程碑作用于主要里程碑范围之内一样，技术决定因素也作用于管理决定因素的范围之内，从而使技术性的策略和操作适应业务策略和基本目标，在业务与技术之间建立一座桥梁。

迭代可能是一小步，开发过程的一段或是通往目的地的路线。迭代是被计划而不是固定的，它有评估标准，其进展也是可以论证的。迭代过程涉及到重复的工作，在每一次重复中都包含一定增量，在迭代的过程中很多工作都是并行的。

用例是基本的需求。例如，登陆和退出系统，输入数据，处理数据，生成报告等功能。由于统一过程是用例驱动的，用例驱动迭代过程。也就是说，迭代过程将被设计、评估以解决那些大量的功能需求（或其中的一部分）从而与用户达成一致，使项目的所有活动和工件都可以回溯到需求中去。业务因素的统计是通过对那些以功能性需求为目的的迭代过程的计划和评估而得到。非功能性需求（可用性、可靠性、性能以及其他特性）在用例所涉及到工序中逐步递增地考虑。

系统拥有一个架构。例如，系统的架构包括各个元素的集合以及元素之间的协作和相互作用，还应该包括各种子系统以解决安全、输入、输出、数据存储、外部通信、报告等问题。由于统一过程（UP）是以架构为中心的，迭代过程的重点在于架构和系统的不断进化。也就是说迭代的目标主要在于解决大量问题所带来的困惑，从而控制系统的复杂性和完整性。系统技术因素的统计是通过对实际系统的改进或是产品的生成而得到的。

风险是项目成功的阻碍，包括人、业务、技术上的问题。例如，来自人的风险主要是由于人员不足，缺乏培训，缺乏经验等；业务风险主要来自资金不足、时间和来自业务交流中的某种承诺等。技术风险主要是对需求和技术了解得不够充分，或使用了尚未成熟的技术等原因所造成的。统一过程可以规避这些风险，迭代过程规避风险并且根据前期的迭代得到的风险反馈决定开发进度，发现其他未知的风险。也就是说，迭代过程规避那些来自于用例和架构的风险，以确保项目成功。

迭代是一种时间范围，在一系列协作被计划、执行和假设过程中可以调节迭代的时间范围，以达到不断的进展。迭代的开始和结束需要用户，项目管理者以及将影响项目进展或受到其影响的技术人员之间进行协商。应选择风险级别最高的用例驱动迭代过程。用例可能在跨越几个迭代周期并不断完善，也有可能在一个迭代周期内使用多种工序。迭代过程将产生系统的一个或多个过渡版本。它也将产生内部的基线和系统外部的评估。由迭代过程得到的反馈和知识积累将为下一个迭代过程做准备。在迭代的过程中评估指标也是不断迭代变化的，通过反复的迭代过程逐渐建立系统全面的评估指标。迭代过程持续时间的长短应与其相关风险的级别高低成反比。在迭代过程的执行中，迭代过程之间应尽量避免交叉。开发周期和各个阶段也可以制定具体的时间范围。然而，对于开发周期、阶段和迭代过程的计划而言，计划越提前，准确性越差。

迭代过程使用与“纯瀑布开发过程”相同的核心开发工序。瀑布开发过程的原则是，在开始下一个开发工序之前应完成当前开发工序的所有活动和工件。而迭代过程包括迭代协作，不断增量优化的目的，以及在整个项目生命周期内不同详细级别的工件。瀑布开发过程并不为系统的局部开发或向项目生命周期中引入的变化提供明确的时间，事实上瀑布开发过程阻碍了这种变化；而迭代过程则在当前迭代的结束之后，下一个迭代过程开始之前提供了局部开发的时间，从而预见系统的变化并做出相应反应。瀑布开发过程是通过一系列连续的开发工序不断进化的，而迭代过程可以在不同的开发阶段间不断前向或后向进化，以及时调整重点，运用各种不同的开发工序，及时发现项目开发的风险。

## 迭代

为了顺利而高效的应用统一过程，我们需要理解迭代过程以及如何线性连续地使用迭代过程。

迭代过程包括迭代过程的计划、执行和评估。确定用例和风险的优先级，其中用例的级别是根据可以减小风险的多少来定义的。在迭代过程的计划上，应选择那些可以规避最高等级风险并可以在当前给定的有限条件下（资金、时间、资源等）进行的用例来驱动迭代过程。在迭代过程的执行上，用例通过每一个核心开发工序不断完善的同时，系统和系统的体系也得到不断的完善。当然，用例无需在一个单独的迭代过程中的每一个开发工序上都

得到完善。迭代过程的评估，是通过实际的结果与预计的目标相比较而得到的，同时对项目的计划和风险进行调整和更新。项目的最终目标是完成系统，因此所有的核心开发工序应该在不引入新的风险的前提下准确及时地得到应用。也就是说，开发者应该负责在恰当的时候使用恰当的核心开发工序。

## 线性步骤

无论是在将重点放在业务建模的第一组迭代过程中，还是在将重点放在需求开发的第二组迭代过程中，以及其它同样在核心开发工序之间进行的迭代过程中，开发组都应该坚持在学习解决方案之前更多的理解问题本身，并将其作为整个项目中各个阶段的都需要做的工作。尽管对于整个系统而言，这些工作的成果可能只是在开发周期结束时才能显现出来。作为线性过程，这一点已经被大家所熟知。图 3 所示为这种必需的工作在线性开发过程中使用的模式。

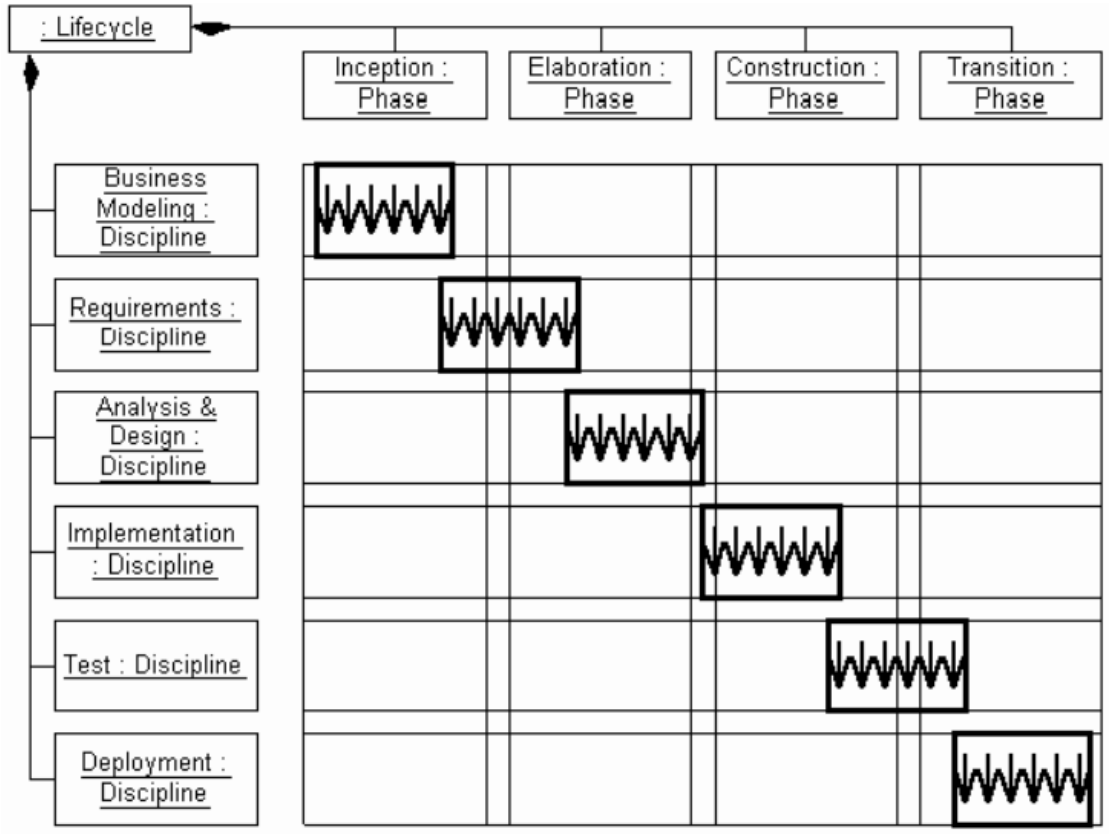



图 3 线性步骤

线性迭代将主要的重点放在开发阶段的宏观层次方面，在这些阶段中各个开发工序被更加分散地跨阶段分布。因此，那些存在于业务与技术之间的平衡将被团队的管理者打破。这些工作常常因为团队的管理者认识到某些必须被立即处理的业务风险，而影响迭代过程多个开发工序中所有的用例。

线性迭代在发现某个与业务相关风险或用例相关风险的时候，将尽量推迟对这些风险的解决或尽量规避它们。同时，这种线性迭代过程也将推迟那些对系统及其架构必要的验证，排除项目开发周期中的机会主义配置。从本质上说，它是一个将各种开发工序分布在各个迭代中的“纯瀑布”开发过程。

## 连续步骤

用例在一个单独的迭代过程的各个核心开发工序中不断进化，开发组都应该坚持在学习解决方案之前更多的理解问题本身，并将其作为整个项目中各个阶段的都需要做的工作。尽管对于整个系统而言，这种工作仅仅是确定了一部分需求，这些需求也许不能在整个项目开发周期内使用或发布，而只能在整个开发周期结束阶段才能表现出来。图 4 所示为这种必需的工作在连续开发过程中使用的模式。

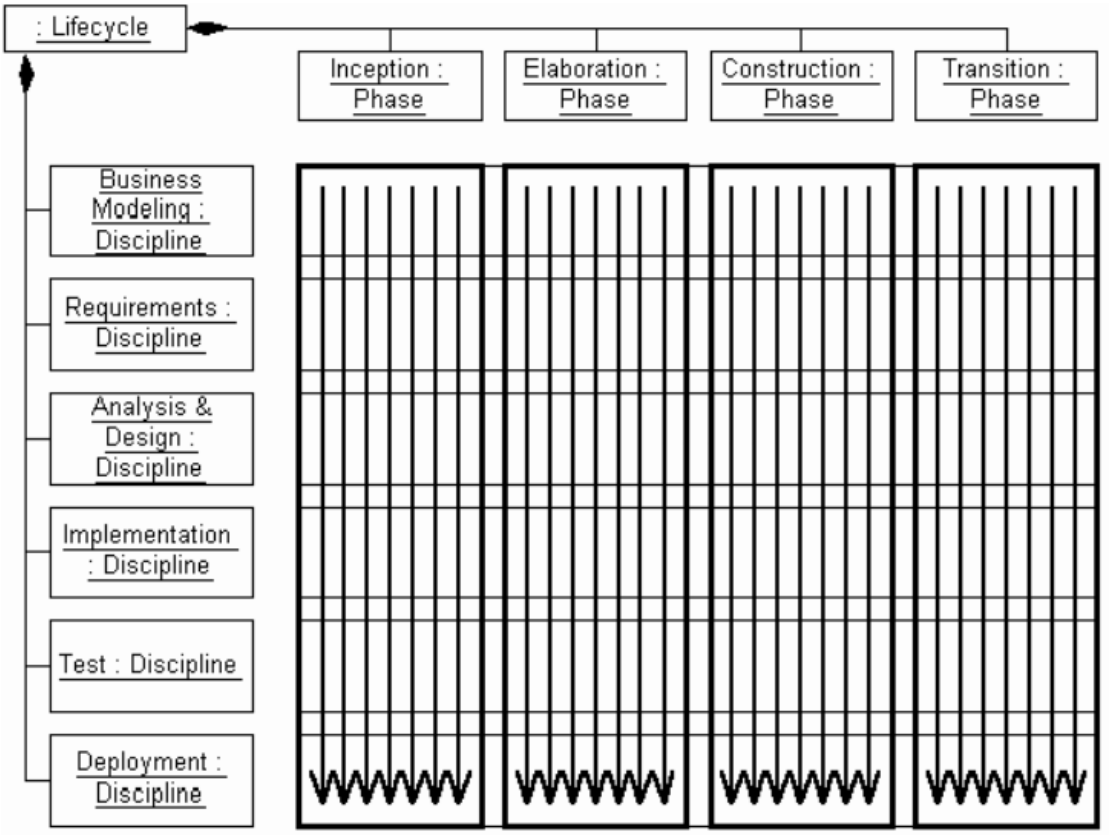



图 4 连续步骤

连续迭代将主要的重点放在开发阶段的微观层次方面，在这些阶段中各个开发工序在阶段间更加分散地分布。因此，那些存在于业务与技术之间的平衡就将被团队的技术成员打破。这种工作常常因为团队的技术成员认识到某些必须被立即确定的技术风险，而影响迭代过程所有开发工序中所有的用例。

连续迭代倾向于在发现某个与技术相关风险或架构相关风险的时候，将尽量推迟对这些风险的解决或规避这些风险。同时，这种连续迭代也可能增加系统集成和验证的难度，推迟架构演习达到充足的范围。从本质上讲，架构范围的过小很可能导致那种用例使前期迭代过程中衍生的架构失效的情况出现。

## 迭代步骤

迭代步骤是对以问题为重点的线性步骤和以解决方案为重点的连续步骤的混合应用。图 5 所示为这种必需工作在迭代步骤中使用的模式，其中四边形代表着那些需要根据具体项目进行调整的并行部分。将四边形中各个边都变成对角线，就是各个工序跨迭代阶段的“纯瀑布”步骤。

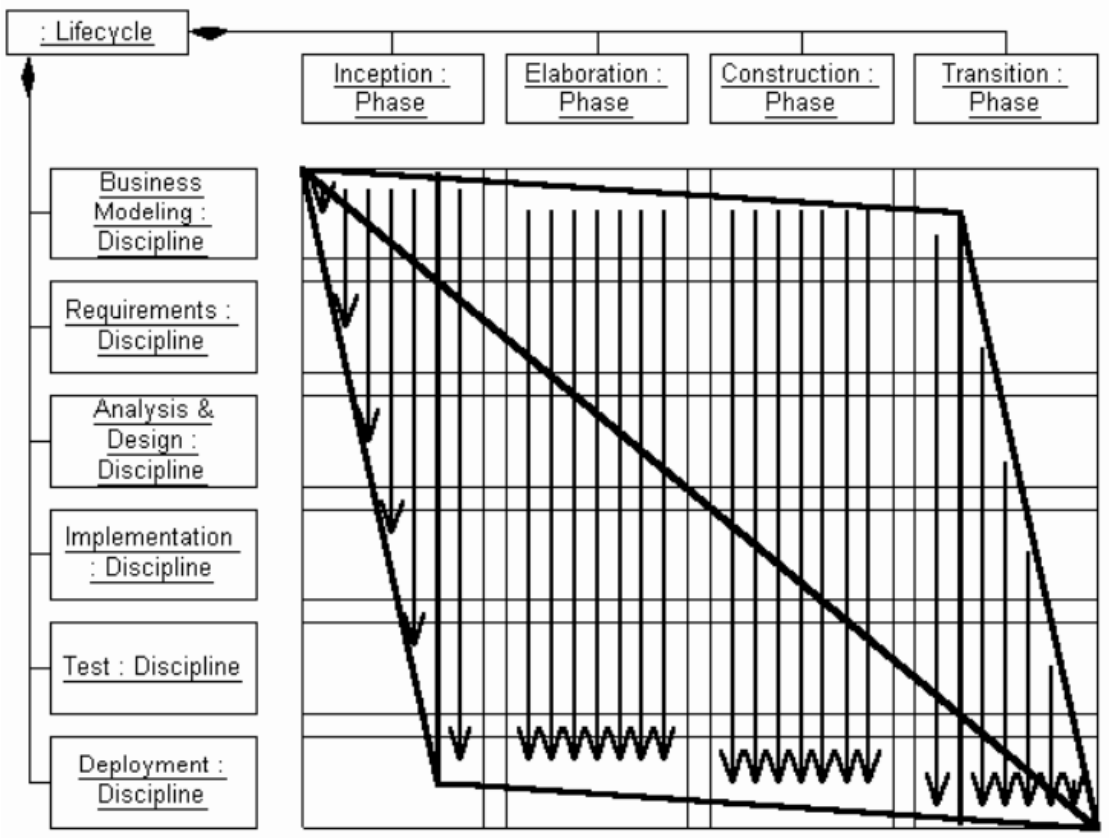



图 5 迭代步骤

迭代步骤的重点是在项目整个生命周期中逐步精化所涉及到的知识。在先启阶段，线性步骤重点是项目的范围，连续步骤重点是架构定义的证明。在细化阶段，线性步骤确定架构的范围，连续步骤确定架构的风险。在构造阶段，连续步骤提供更多的开发机会。在产品化阶段，线性步骤和连续步骤的重点是系统的完成和项目的终止。

总的来说，这种工作的进行在项目的开始阶段将比较困难，之后则逐渐实现并行应用所有核心开发工序，恰当运用所有辅助的核心开发工序，在开发周期的结束阶段则比较简单。在迭代的执行中， 在那些有着生成和消耗的关系活动中角色、活动以及工件的协作 一旦生成活动足够成熟后及时地进行更迭 。在迭代的执行中，其内容将包括生成和消耗的关系活动中角色、活动以及工件之间的协作，同时，将满足产生消费关系的活动联系起来，并及时地迭代以使得消费活动可以生成活动中的输入变得足够成熟后及时开始。

## 高效成功地运用统一过程

为了高效成功地使用统一过程，我们需要理解很多过程框架的应用指南（通过培训）。

发生在给定角色、活动、工件间的协作，其主要动力来自于项目经理、架构师和过程工程师等角色。其余的角色并不是次要的，而是辅助上述这些角色。项目经理负责整个项目；架构师负责系统架构；过程工程师负责统一过程的运用和开发案例。项目中各种角色的交流与协作是高效成功运用统一过程的关键因素。这些协作包括架构师所定义的系统架构，过程工程师所建议角色，活动以及提交系统所需的工件，项目经理为上述工件进行活动提供所需资源。这些架构、角色、活动、工件以及资源之间的交互包括也将包括其他相关知识的高效成功地运用。这些角色重点是平衡一个步骤范围内的多种相关因素的同时，弥合目标和实际版本之间的差异。也就是说，这些协作必须有重点地，平衡地，迭代地运用以获得成功。另一方面，项目经理过于教条死板会导致过分的线性迭代；系统架构师过于教条死板会导致过分的连续迭代；过程工程师过于教条死板会导致混乱的状态。上述角色的问题最终将导致迭代步骤的优势无法体现。一般来说，在那些包括项目经理，过程工程师等角色的项目中往往会曲解这些主要的动态性能，（由于每一个角色都有明确的压力，目标）导致这些角色的直接利益冲突；以至于增加了项目失败的潜在威胁。

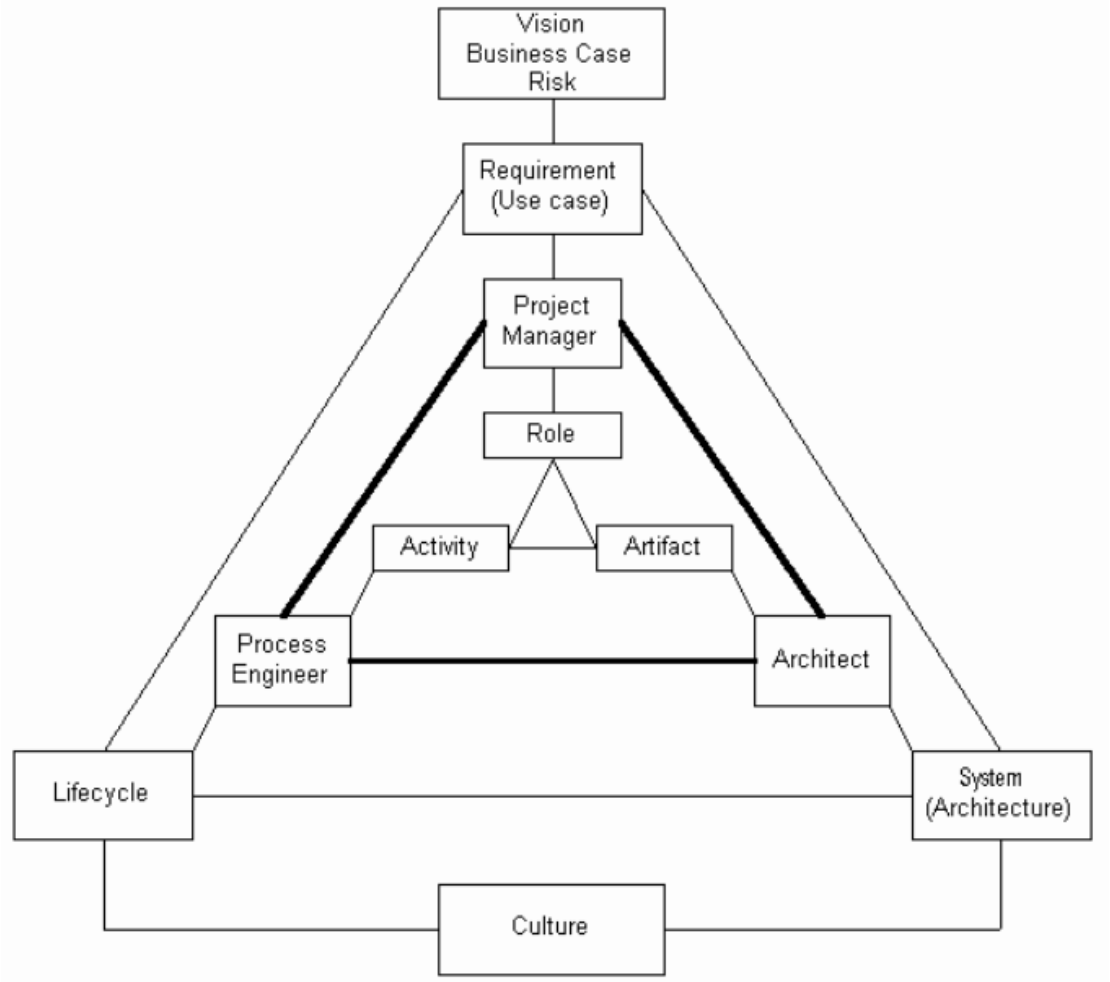



图 6 UP 的动态特性

本文讲述了焦点、平衡和迭代过程的知识，那些关于特定角色、活动以及工件的应用指南（通过培训）超出了本文的范围。

## 焦点

在应用统一过程的项目中，我们应关注并遵循下列指导步骤：

（1）鉴于统一过程中所有内容都是可选的，应根据不同的因素及其可能的分支做出相应的决定。一个经常要问的问题是：“为什么？”——没有必要执行统一过程中所有的内容，而只需要进行那些需要进行的。一个过程工程师将解释为什么角色、活动和工件这样被利用。另一个经常要问的问题是“如果这样，那么会怎么样？”也就是说，确定我们应该做什么。过程工程师将解释如果一个角色、活动以及工件被利用会有什么可能。还有一个问题经常要问：“接下来做什么？”在给定“如果这样，那么会怎样”的条件下“接下来做什么？”。过程工程师将解释接下来什么样的角色、活动以及工件应该被应用。如果无法回答这些问题，则表明其对特定项目了解得不够充分。

（2）聚焦于项目环境（context），然后是基本内容，并迭代地，增量地尽力弥合项目环境（context）与内容之间的差距。不具备对某个项目的环境（context）相关知识或是不具备统一过程的基本知识，而使用统一过程，都将增大项目失败的可能性。过程工程师需要能够弥合项目与统一过程之间的差距；也就是说能够将统一过程的基本元素运用到特定项目的环境（context）中去，其中关于统一过程的基本元素，本文已经着重强调了。

（3）聚焦于“统一过程的精髓”而不是统一过程的形式。统一过程既不松散混乱，也不死板迟钝，而是具有很强的灵活性，动态性。统一过程仅仅是说明或建议开发人员应该如何决定和执行。如果无法平衡其中的关系，则表明过分的关注“统一过程的形式”，而不是“精髓”。这一点是很多过程工程师运用统一过程的通病，也就是说，他们无法平衡“形式”与“精髓”。

（4）对项目经理，系统架构师以及过程工程师充分地授权以弥合交流中目标与项目实际版本之间的差距。在授权的同时，确定上述角色与整个团队中其他角色之间相互影响的因素，从而提高项目成功的可能性。项目经理应该是一个领导者，而不是简单的项目管理者，或是教条死板的执行者。系统架构师应该是一个领导者，而不是简单的理论家或是技术人员，既不过分教条也不过分关注细节。过程工程师应使过程的应用成为可能并且应用顺利，而不是强迫过程的实施。团队应该可以迎接挑战，把握机会而不是停滞不前。每一个工序都应该有一个角色来领导工序内全部的工作，得到并维护工序相关模型（宏观图），同时每个工序也都有其他的角色来获得并维护模型的细节（微观图）。

虽然有很多指南的应用明确针对于过程工程师，但是这些指南也可以应用于其他角色。除此之外，其他的指南也可以作为上述指南的一种补充。

## 平衡

在应用统一过程的项目中，我们应遵循下列指导步骤进行项目平衡：

（1）不断寻找平衡；这一点毋庸置疑，过程工程师必须考虑角色、活动和工件的必要性、充分性和一致性，确定风险以保证项目成功。对必要性的判断主要是考虑其是否是项目不可或缺的；对充分性判断主要是考虑其是否满足给定的目标；对一致性判断主要是考虑其是否与其它事务前后矛盾，相互冲突。其中，一致性可以使用指南进行判断。对统一过程基本元素重视程度的不足和对过程框架中各个过程元素的认识的不足通常表现为对必要性、充分性和一致性的判断错误。过程工程师应该提供建议，从而尽可能地考虑到项目潜在的风险，以确保项目中没有任何事情被忽略而导致无法实现目标，并给出相应论证。

（2）对于角色而言，在项目中，使用全部角色，或不使用任何角色；给团队中每一个成员都分配角色，或不分配任何角色的情况都很少见。应根据每个角色的意义做出决定。

（3）对于活动而言，在项目中，进行全部活动，或不进行任何角色；团队面面俱到的进行全部活动，或仅仅进行那些最必要的活动的情况都很少见。应根据每个活动的意义而做出决定。

（4）对于工件而言，在项目中，生成全部工件，或不生成任何工件；团队面面俱到的生成全部工件，或仅仅生成那些最必要的工件的情况都很少见。应根据每个工件的意义而做出决定。

（5）对于重复而言，在项目中，保持进行固定数量的重复，或不进行任何重复和改变；对于变化本身而言，没有原因而产生变化，或每一个原因都产生一个变化的情况都很少见。应根据每个重复的意义而做出决定。

（6）不要相信一个可以在没有任何前提条件下证明一切的过程工程师。这种情况下，对于上述元素意义方面的问题，他们经常回答：“嗯，这个……！”。不要相信无法证明任何事情的过程工程师。这种情况下，对于上述的“原始问题”，他们经常回答：“相信我……！”。

（7）不要相信那些只注重统一过程的形式而不是精髓的“纯理论家”。从实用的角度来说，这些“纯理论家”迟早都会为项目的成功结束而做出某些折衷和平衡，因为没有折衷和平衡，意味着项目存在重大的风险。对于大多数过程工程师而言，无法正确地折衷和平衡是他们的通病。

虽然有很多指南的应用明确地针对于过程工程师，但是这些指南也可以应用于其他角色。除此之外，其他的指南也可以作为上述指南的一种补充。

## 迭代

在应用统一过程的项目中，我们应遵循下列指南进行迭代开发：

（1）开发阶段以迭代为主。脱离特定阶段的环境而进行的迭代活动将失去灵活动态的特性，或松散混乱，或迟钝死板。不考虑或是过分考虑环境（context）的迭代都将阻碍统一过程优势的发挥。在对某个项目的迭代进行计划，执行，评估的过程中，应充分考虑迭代的阶段，确定这些阶段的具体目标以及如何在特定环境中达到这些目标。

（2）迭代的开始和结束应该取决于时间范围而不是涉众。在迭代中，涉众是否积极参与将直接影响涉众在整个项目中所起的作用。

（3）突出重点且兼顾平衡是项目迭代成功的关键。没有明确的目标，客户将无法分出轻重缓急，无法正确地交流，更不可能互相达成一致。当然，客户上述能力的积累是具有顺序性的，没有明确的目标，用户就不可能确定问题的优先级；不确定优先级，也就无法和其他用户进行建设性的交流；没有交流便无法协商讨论，也就没办法达成一致。因此，要确保客户具有这样的能力。

（4）不要无谓地去掉迭代过程！也就是说不要过早终止一次迭代，因为这将影响到迭代附属物的产生和用户对项目真实状态的了解。除非是由于灾难性的突变而终止那些既耗费资源而又没有任何成果的迭代过程。例如：如果当前正在进行的迭代可以被证明对于项目需求或是技术假设是不可能的，那么就应该终止它。

（5）不要无谓地修改迭代过程！也就是说不要为增加用例而修改迭代过程的作用范围，因为这将影响到迭代附属物的产生和用户对项目真实状态的了解。除非是由于那些未预料和计划的需求的增加——如果不增加这些需求会导致灾难性的后果——，而必须对当前迭代过程进行修改。——虽然更好的做法是将新增加的需求加入到后期的迭代过程中。例如：如果不增加这些未得到预见的需求，将会导致项目资金的减少，那么应该修改当前的迭代过程。

（6）不要无谓地延长迭代过程！也就是说不要为适应某个用例而推迟迭代过程的结束，因为这将影响到迭代附属物的产生和用户对项目真实状态的了解。除非是由于那些需求的确定——如果不增加这些需求会导致灾难性后果——，而必须对当前的迭代过程进行延长。——虽然更好的做法是将这些需求加入到后期的迭代过程中。例如：如果不确定这些在当前迭代过程中未确定的需求，将会导致项目资金的减少，那么应该延长当前的迭代过程。

（7）不要“拖延”（或者允许整个团队的拖延）而要不断取得进展。也就是说迭代过程将使整个团队具有成就感。

（8）在整个生命周期的各个迭代过程之间敏捷地（具有易适应性和对团队，个人，过程以及工具发展的前瞻性）进行计划，执行和评估。也就是说，运用给定的资源（团队，个人，过程以及工具）以达到局部目标，并尽可能多地取得整体上的优势。变压力为动力，突出重点，兼顾平衡，反复迭代地缩小和弥合现实与理想之间的差距，就一定会获得成功。

此外，其他的指导步骤也可以与上述步骤一起使用。

## 结论

显而易见的，人是项目获得成功的"最根本要素"。不要把统一过程教条化，也不要强制执行它，而是应该依赖人来运用统一过程！应把重点放在团队上而不是某个特定项目上，因为正是团队本身运用他们的过程并最终获得成功。同样地，统一过程（UP）应该被科学地定义，被艺术地运用。

统一过程（UP）是由用例驱动的、以架构为中心的、迭代增量的开发过程框架，它使用对象管理组织（OMG）的统一建模语言（UML）并与对象管理组织（OMG）的软件过程工程原模型（SPEM）兼容。通过正确地理解统一过程（UP），掌握各种指导步骤（通过培训）从而获得成功与高效，使得统一过程（UP）受到广泛赞誉！与此同时，也正是通过对统一过程及其各种元素的不断探索，运用和积累，使统一过程（UP）实现了其自身的价值。

Sinan Si Alhir是一位资深的顾问，也是"UML in a Nutshell" (O'Reilly and Associates, Inc., 1998)一书的作者,曾经在《软件工程百科全书》（"Encyclopedia of Software Engineering" (2nd Edition, John Wiley Sons, 2001)）中发表过《统一建模语言》（ "Unified Modeling Language (UML)" ）和《统一过程》（"Unified Process (UP)"）两篇论文(2nd Edition, John Wiley Sons, 2001)，同时也发表过很多其它论文。

Agile软件开发丛书

PEARSON Addison Wesley

# 有效用例模式

# Patterns for Effective Use Cases

Foreword by Craig Larman

[美] Steve Adolph
Paul Bramble 著

车立红 译

UMLChina 审

UMLChina 指定教材

清华大学出版社

# 扩展 UML 活动图实现生产系统中的工作流建模

Ricardo M. Bastos, Duncan Dubugras A. Ruiz 著, Michael.fly 译

查看评论

## 摘要

本文提出了一个使用工作流概念在生产系统中描述业务流程的方法。在此，我们对UML的活动图进行了扩展，提出了工作流活动图（Workflow Activity Diagram-WAD)的概念,它采用了C-WF模型概念。C-Wf模型描述了业务流程中涉及的结构性的和功能性的企业对象，如企业活动、人力资源、机器资源等等。WAD描述了一个工作流模型，该模型用来标识该工作流执行所需要的活动和资源，并定义了它们之间的关系和次序。通过UML用例的深入使用，我们的方法进一步增强了UML在业务建模方面的可用性。

## 1 介绍

UML近来已经发展成为用于信息系统建模的标准语言[11]。这一情况的出现归因于建模工具的大量涌现以及围绕它们的集成应用的增多。伴随而来的是，已经出现了多个采用UML作为解决不同类型问题的建模语言的案例。当谈及使用UML来描述工作流模型时，也已经出现了一些相关的工作成果－[1][7][8][14][15]。

C-Wf（[3]）是一个使用工作流概念、基于面向对象的方法来对生产系统中的业务流程进行建模的模型。 通过C-Wf参考模型类，使得有可能构建独特的企业模型，该模型包含生产计划处理所需要的元素，以及从工作流的观点出发所需要的用来监视业务流程执行情况的基础信息。

本文提出了一个使用工作流概念在生产系统中描述业务流程的方法。在此，我们对UML的活动图进行了扩展，提出了工作流活动图（Workflow Activity Diagram-WAD)的概念,它采用了C-WF模型概念。通过UML用例的深入使用，我们的方法进一步增强了UML在业务建模方面的可用性。

本文按下列方式进行组织：第二节对C-Wf模型进行了简要描述；第三节检查了UML工具在工作流建模方面的应用；在第四节中，详细介绍了工作流活动图（Workflow Activity Diagram）；第五节讨论如何应用工作流活动图；第六节陈述了最后的结论以及展望。

## 2 C-Wf 参考模型

C-Wf模型（图1）是一个对象参考模型，描述了业务流程中涉及的结构性的和功能性的企业元素，如企业活动、人力资源、机器等等（[3]）。C-Wf模型基于CIMOSA （[16]）和WfMC（[6][13]）的基础概念，并通过工作流管理系统（WfMS）实现对一定范围内流程执行情况的监控。 目标包括：(1)、捕捉描述领域流程需要的所有信息；(2)完成对所需的辅助信息的映射。

C-Wf 模型的基本概念分为两个主要方面：(1)、流程设计和定义；(2)、流程实例化和控制。图1中显示了这两个方面的信息，其中左边对应于上面所说的第一点；右边对应于第二点。

流程设计和定义基于 CIMOSA 结构，而流程的实例化和控制则是基于 WfMC 标准。事实上，C-Wf 从 CIMOSA 所提供的关于设计产品流程方面的丰富而多样的概念里获益多。而 CWf 之所以能够使用 WfMS 来管理产品流程的执行，是因为它的概念与 WfMC 技术直接相关。这样，使用 C-Wf 建模构造出来的产品流程的实例就几乎能够完全以可建模的形式被 WfMS 所管理。

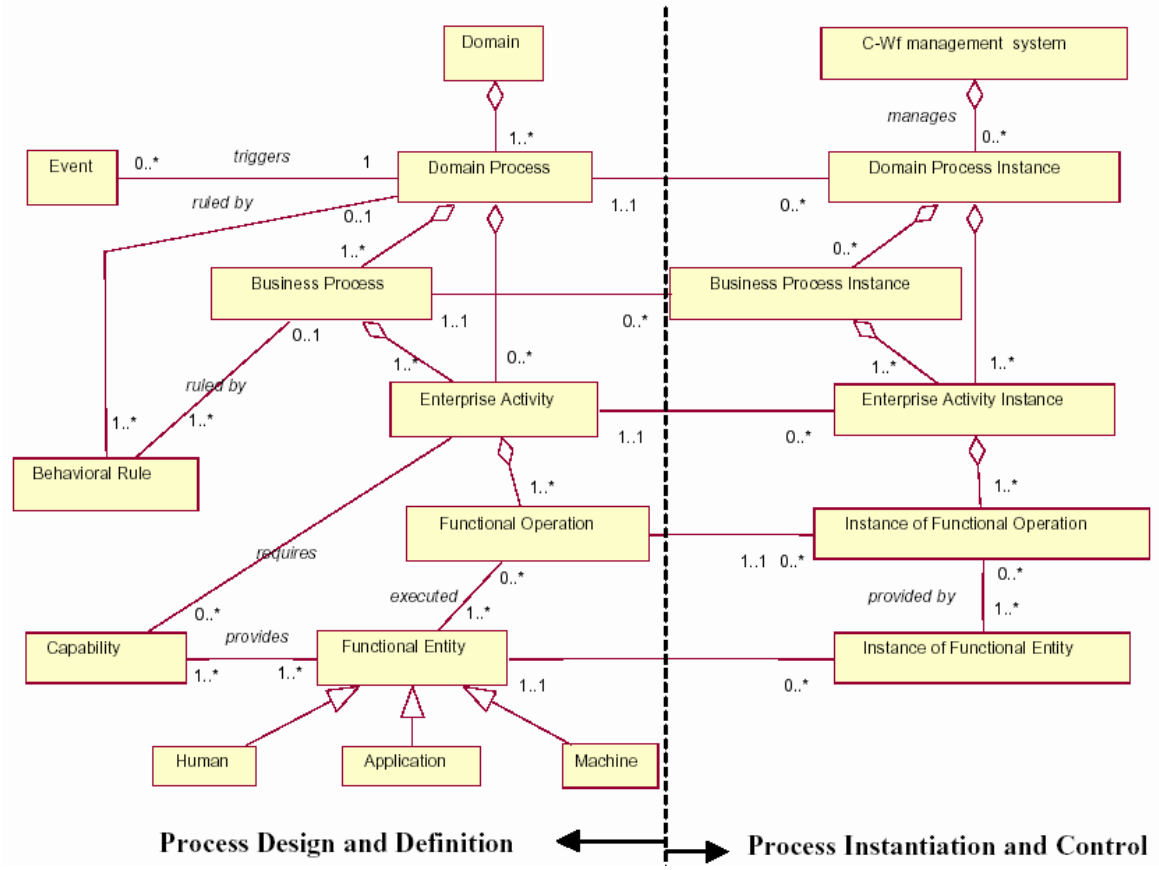



图1 C-Wf的UML类图

### 2.1 流程定义

领域（Domain）基本上由一组业务目标和组织约束所定义，并且由一序列交互性的领域流程（Domain Processes）组成。 领域流程在给定的约束下完成部分或整个领域目标。领域流程通过事件（Event）激活,在功能上可以分解为业务流程（Business Processes）和企业活动（Enterprise Activities）。业务流程代表企业活动的集合。业务流程和企业活动都可以用于不同的领域流程。企业活动定义了在企业中执行的基本任务，这些任务通过某些输入产生输出，并需要在它们执行的整个期间分配时间和资源。每个企业活动包括一个或多个由功能实体（Functional Entities）执行的功能性操作，这些功能性操作分别需要一定的执行时间。功能实体代表产品资源（人、机器和应用程序）。为了保证企业活动和功能实体之间的一致性，功能实体提供了一序列能力来完成企业活动的需要。

领域流程和业务流程的动态特性是通过行为规则（Behavioral Rules）进行定义的。行为规则描述了企业活动之间的互相联络，包含下述类型：流程的开始和结束；单线；与－分支；与－联合；或－分支；或－联合和循环。领域流程的完整描述在它的企业活动方面是通过与工作流产品流程相应的行为规则互相联络起来的，而WfMS在资源分配后可以实现对流程实例的管理。事实上，领域流程、业务流程和企业活动都是描述企业模型的功能和行为特性方面的概念。

### 2.2 流程实例化

C-Wf管理系统（C-WfMS）负责创建和控制所定义的领域流程，并且当收到一个（或多个）事件时负责发起领域流程的实例。当资源分配以及确定了活动和资源的计划安排后，C-WfMS将能够控制每个企业活动实例和涉及的功能实体（产品资源）的开始和结束时间。 由于需要定义分配到企业活动上的功能实体的时间粒度，我们考虑将一个功能实体在整个执行期间仅分配到一个企业活动上。

## 3 UML 工具和工作流

在[9]中，作者建议采用用例来标识系统的功能需求。通过用例，使得可能为公司构造一个业务模型。业务用例模型以业务用例和业务执行者的形式描述了公司的业务流程。与软件系统的用例模型类似，业务用例模型是从使用角度去描述系统，并且对系统如何给用户提供价值进行了概述。

实际上，该作者所介绍的软件开发过程被定义为用例驱动的过程，在这个过程中，开发员创建了一序列设计和实现模型来实现前面定义的用例。

按照[9]中定义的软件开发过程，分析类基本上是通过用例确定出来的。这种做法证明是正确的，因为用例的实现就是以分析类和它们的对象的形式来描述的。用例的不同流程或场景的实现是通过分析对象之间的交互作用来加以描述的。

企业软件总是工作于一个高层次的业务流程的背景之下，其中存在有与系统进行交互的执行者，这些执行者定义了该软件的用例模型。这些业务流程可以被看作是一些工作流类型，因为它们代表了业务的工作流程和目标（[4]）。因此，我们认为公司的工作流模型可以使用业务用例模型作为参考来进行定义，从而实现WfMS与企业软件之间的协调一致的集成。

### 3.1 活动图

按照[4][9]，活动图可以用于对用例建模。通常，对用例描述来说，需要标识出涉及的执行者，以及描述事件的流程（基本和可选路径）。以用例为基础，就有可能识别出要实现的功能需求所需要的对象，以及确定用例执行期间系统应有的响应。

在活动图中，我们确定了为执行用例必须实现的活动以及这些活动之间的关系。除此以外，还可能确定活动中涉及的对象，并定义这些对象的角色，以及它们的状态和属性值的变更。表1中描述了从[4]和[11]中提取出来的有关活动图的主要概念：

| 概念 | 详细描述 |
|---|---|
| 活动（Activity） | 在状态机内正在进行的非原子动作。活动最终导致一些动作。 |
| 动作（Action） | 一个可执行的原子操作，导致系统状态的改变或者是返回一个值。 |
| 动作状态（Action State） | 代表一个原子动作的执行状态，通常由一个操作发起。 |
| 活动状态（Activity State） | 它是复合状态，它的控制流由其他活动状态和动作状态组成。活动状态是非原子的，意味着它们可能会被中断。活动状态可能被进一步分解，它们的活动由其它活动图描述。 |
| 跃迁（Transition） | 代表两个活动之间的控制流。跃迁显示了从一个动作或活动状态到下一个动作或活动状态的路径。 |
| 对象流（Object Flow） | 代表活动图中的控制流中涉及的对象。 |
| 对象状态（Object State） | 对象的生存期间（也就是在对象满足某些条件，执行某些活动，或等待某些事件期间）的条件或情形。 |
| 泳道（Swimlane） | 为了对活动的职责进行组织而在活动图上进行的分块。泳道并没有固定上的含义，但是它们通常对应于业务模型中的组织单元。 |

表1 活动图概念

不管怎样，[5]提出活动图作为UML工具的应用还具有一些局限性，因为它还不能做到对事物的充分理解。事实上，还有一些作者也表达了同样的意思：[14][12]。也许，这可能就是为什么还没有出现很多使用活动图进行工作流建模的例子的原因所在。

## 4 工作流活动图

C-Wf 参考模型的主要贡献在于它集成了企业和工作流建模概念。为了使活动图在与C-Wf协同用于企业建模方面时切实可行，以及增强它在业务用例建模方面的应用，我们建议对活动图进行扩展，称为工作流活动图(Workflow Activity Diagram—WAD）。在此扩展中，我们使用版型（stereotypes ）来建立活动图元素与C-Wf的类之间的关联，此外，我们还为活动图定义了一些新属性。版型是UML的一种扩展机制，用来从现存的构造块中针对特定问题专门派生出一种新类型的构造块（[4]）。

WAD中定义的每个元素都是通过版型进行定义的，以便建立起与C-Wf类的关系。表2列出了活动图中的主要元素及其相关的C-Wf类。

| 元素 | C-Wf 模型类 | 注解 |
|---|---|---|
| 泳道（Swimlane） | 领域（Domain） | 代表领域流程执行时所包含的业务流程或企业活动所覆盖的组织单元。 |
| 活动（Activity） | 业务流程（Business Process）<br>企业活动(Enterprise Activity) | 业务流程和企业活动定义了WAD的分解层次。 |
| 跃迁（Transition） | 跃迁（Transition） | 根据来自领域流程中的行为规则进行定义。 |
| 对象流（Object Flow） | 事件（Event）<br>功能实体（Functional Entity） | 功能实体类特指三种类：Human, Application 和Machine。 |

表2 活动图元素和C-Wf模型类之间的关系

WAD所针对的主要方面包括：

（1）使用C-Wf 概念将工作流模型当作一组用例来描述；

（2）将来自C-Wf 的对象应用为版型，以便描述领域流程和它的业务流程及企业活动；

（3）采用工作流建模的原则来增强活动图的表现能力；

（4）利用UML已经是OMG的标准这一事实，并且它的使用还在快速增长之中。

在我们的方法中，业务模型中所标识的每个用例都与一个领域流程相关联，该领域流程由WAD进行描述。当业务流程出现在WAD中时，每个业务流程都代表了一个潜在的可重用的子流程，它们也可以与业务模型中的用例建立关联。这意味着每个业务流程都是在一个领域流程中进行标识，同时也必须在其他的WAD中详细描述，以便确定它的特定的工作流。为了保证WAD的一致性，所有连接到某个业务流程的对象流都必须出现在详细描述该业务流程的WAD中。

企业活动是一组功能操作（Functional Operations）的集合，这些功能操作由一个或多个功能实体（Functional Entities.）去执行。这样，我们就将每个功能操作与有资格执行它的功能实体之间建立起了直接的关联。资源分配过程将为企业活动实例分配执行功能操作所需要的特定功能实体。为了定义企业活动的时间粒度，我们考虑认为企业活动的各功能操作组件是同时执行的，这意味着每个功能操作执行所需要的功能实体必须在企业活动的全部执行时期内都处于可用状态（即该功能实体已分配）。

### 4.1 工作流活动图与 C-Wf 概念

WAD中包含的对象通过使用UML中定义的常用标记符号来描述，如构造型（构造型名称作为文本字符串放置在书名号中，然后再将这两者作为一个整体放置在对象符号中）。这样，为WAD定义的构造型类就有如下这些：

<<Event>>

事件（Event）实例用作为领域流程的触发器。这样，事件实例就与WAD中的第一个业务流程或企业活动建立了关联。事件是由一个执行者或一个用例产生的外部输入，从而就启动了WAD。

<<Human>>

这个类的实例代表功能操作执行时涉及的人类功能实体。因此，人类实例与企业活动是相关联的，虽然由于企业活动的传递性，它也有可能与业务流程相关联（业务流程是企业活动的集合）。

<<Application>>

代表通过UML包标识出来的软件应用程序，和与工作流活动相关联的各个对象。包是一个通用机制，用来将元素进行分组组织。包在图形上用一个大矩形框来表示，矩形框上带一个短小突出的部分（一个小矩形）连在大矩形框的一个角上（通常是左上角）。隶属于某个软件应用程序的对象被封装进包中，这些对象与其他包中对象之间的关系定义了这种类型功能实体间的综合链接关系。这样，来自不同应用的对象就可以与同一个工作流活动建立关联了。

考虑WfMC中的术语（[13]），我们最后选择仅将分类为工作流相关数据（Workflow Relevant Data）的对象加入WAD中（WfMC的其他类型还有：应用数据【Application Data】和工作流控制数据【Workflow Control Data】）。

应用(Application)对象与[9]中定义的实体和控制分析类相对应。实体（Entity）类用来保存持久性的信息，实体类通常显示了一个逻辑数据结构，有助于理解系统所依赖的信息。控制类用来实现协调、序列化、事务控制和对其他对象的控制，并且经常用来封装与特定用例相关联的控制逻辑。控制类也用来描述复杂的派生和计算逻辑。图2描述了与应用有关的对象：

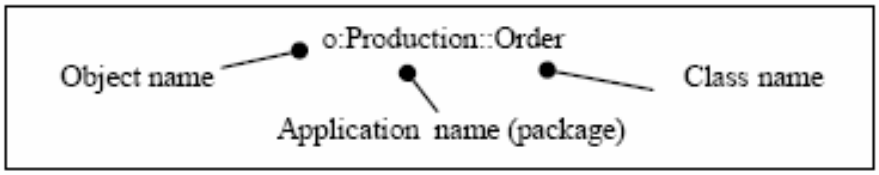



图 2：对象路径名

<<Machine>>

代表用来执行功能操作的所有类型的物理功能实体。 这样，一个机器实例就能够与业务流程和企业活动建立关联。人类、应用和机器实例的状态根据功能操作的执行状况来定义；事件实例无须定义状态，因为它们仅仅是领域流程和业务流程的触发器。

### 4.2 工作流活动图的表示

WAD这一概念的提出就是为了增强UML活动图用于工作流建模方面的能力，它所针对的主要方面包括：

1、描述领域流程规格说明书中所需要的不同活动层次（业务流程和企业活动）；

2、建立活动（业务流程或企业活动）与执行它所需要的资源（功能实体）之间的关系；

3、标识与活动有关的软件应用程序（通过包来表示），这些活动组成了领域流程工作流。

图3显示了一个为假想中的制造企业下生产订单用例所确定的领域流程的WAD。该工作流中的活动需要与来自生产系统和库存系统中的对象建立关联。组成WAD的元素依次被检测以确定工作流的下一步动作。

**业务流程和企业活动：**业务流程和企业活动是WAD中定义的两种类型的活动。我们将企业活动采用与UML活动图中的活动相同的表示法（例见图3中的【Opening Customer Order】企业活动）。而对于业务流程，我们为其定义了一个构造型表示法，例见图3中的【Schedule Customer Order】业务流程。

在某些情况下（如图3中的【Identify Required Materials】企业活动），需要表达出同一个活动根据情形可能需要运行一次或多次这种含义。在UML中，这种情况定义为动态并发，意味着该活动的多个拷贝可能会同时发生，在活动符号部分的右上角显示了并发的多重数字符串（[11]）。

当我们的工作流中存在有一个动态并发的情况时，你必须注意该并发活动的多个拷贝是在同一个时间间隔中执行的。交叉或同时执行两个或多个线程可以完成这一并发过程。如果情形需要同时执行，则需要为每个线程分配每个人类或机器功能实体的一个实例。另一方面，对于《Application》功能实体，则每个线程可以使用不同的实例作为输入，并且生成不同的实例作为输出。为了表示动态并发的不同可能性，我们扩展了UML中的多重性字符串的表示法，为交叉执行的线程使用【*i】标记，为同时执行的线程采用【*s】标记。

**对象流：**如前所述，功能操作(它们组成了企业活动)的执行需要具有一个或多个功能实体。如此一来，则每个企业活动至少需要一个相关联的功能实体来保证该企业活动的执行。【<<Human>>】对象和【<<Machine>>】对象对应于真实世界中用来执行活动的对象，如图3中的【*Opening Customer Order*】企业活动中的售货者对象（Seller）。

【<<Application>>】对象代表由企业活动产生或销毁的信息系统对象（如【*Opening Customer Order*】企业活动就创建了一个【*Order*】对象）或者用来完成一个功能操作的对象（如【*Identify Required Resources*】企业活动中的【*OrderResource*】对象）。

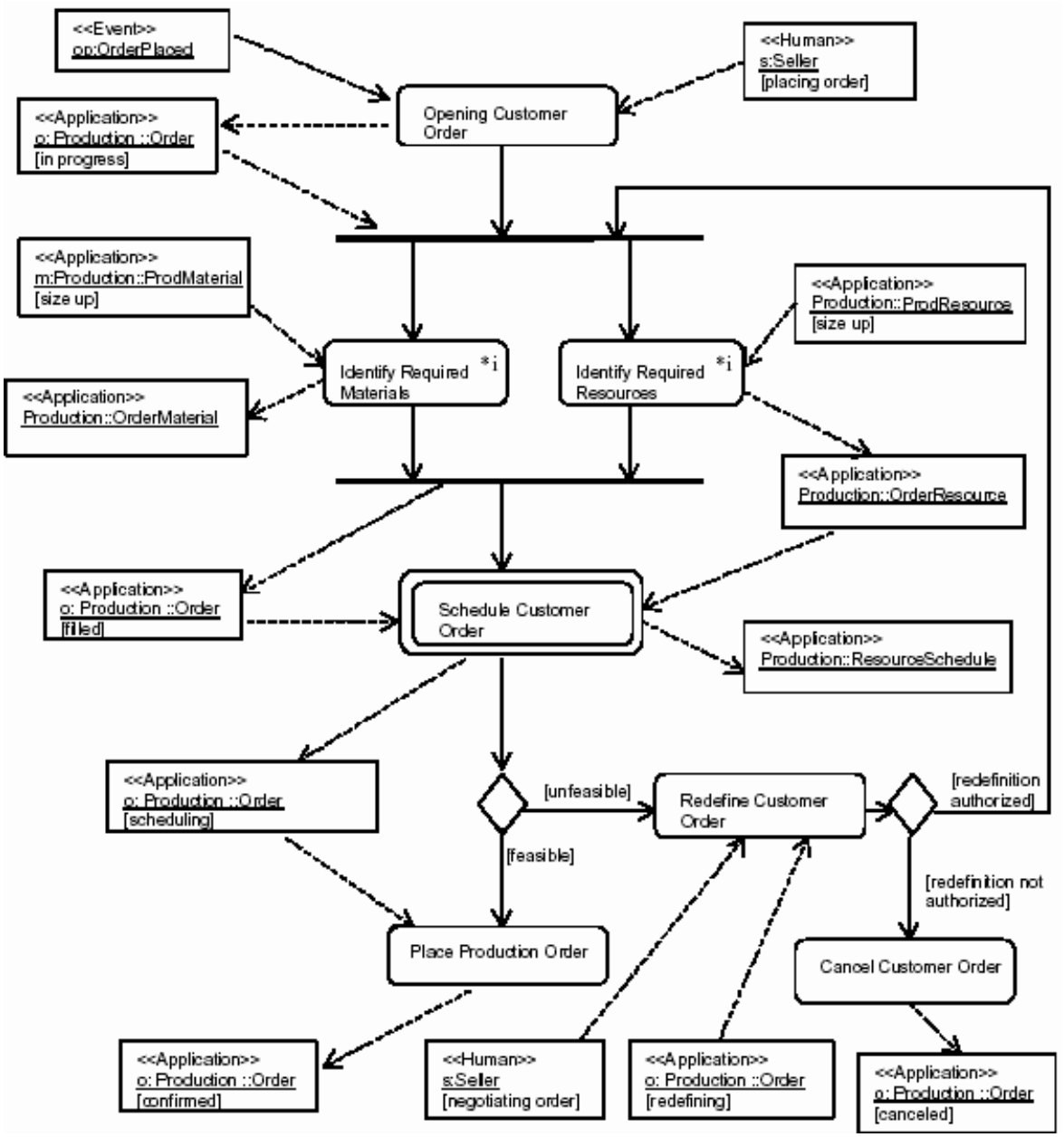



图 3 工作流活动图示例

为了表示对象流，我们采用了如图4所示的标记符号。图中的功能实体版型定义了该对象在WAD中的含义。对象状态标记具有相同的UML标记，它是可选的。对象状态与企业活动所需要的功能操作相关联，因为它代表操作执行期间对象的状态（例如：对象【*m:Production::ProdMaterial*】在【*Identify Required Materials* 】功能操作执行期间将保持状态为【*size up*】）。实际上，由于【<<Application>>】对象隶属于某个软件应用程序， 其中的功能操作实际上是由该信息系统中的这些对象的方法来具体实现的。

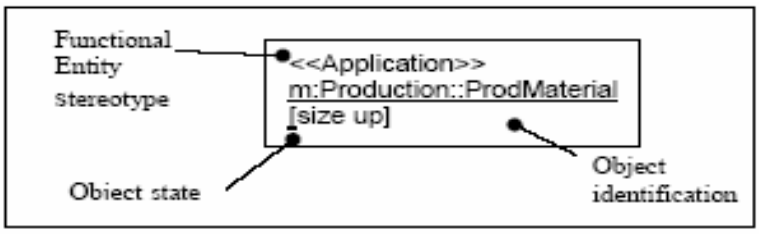



图4 对象表示法

这些对象可能是某个活动的输出，同时又是许多其他活动的输入。【<<Human>>】和【<<Machine>>】对象总是活动的输入对象，因为它们是用来执行一个功能操作的。而【<<Application>>】对象既可以是活动的输入对象，又可以是活动的输出对象；当活动无须执行它自身的任何操作就可以创建、销毁或仅修改某些属性值时，这些【<<Application>>】可以认为是输出对象，当活动需要该【<<Application>>】对象来执行某些功能操作时，则这些【<<Application>>】对象就可以考虑为是该活动的输入对象。

【<<Event>>】对象总是必须作为WAD中第一个活动的输入对象。确定工作流模型中由活动生成的【<<Event>>】对象可能会是一件有意义的事情，在此情况下，一个【<<Event>>】对象代表一个领域流程的输出，它将同时又是另一个领域流程的输入事件对象，从而在这两个领域流程之间建立起了链接。

图5中，显示了包及其中的对象，这些对象曾经在图3所描述的WAD图中出现过。在这个图中，所有出现在对象流中的对象都被显示在它的对应的包中。在【Functional Entity Machine】包中没有对象，因为这种类型的功能实体并不对应于用例【*Place Production Order*】。实际上，流程活动包通过功能操作类依赖于功能实体包，这意味着功能操作类的每一个实例可以被链接到一个或多个功能实体实例。

**跃迁（Transitions）：**跃迁定义了活动中的控制流。从一个活动到另一个活动的跃迁发生在当前活动已完成并且条件为真时。与UML一样，我们并没有为跃迁条件定义一个正式的格式。

除了顺序方式的跃迁外，可能还存在另外一种形式的跃迁路径来对控制流建模，表3中对其作了描述。这一符号标记是对【13】所定义概念在UML活动图上的扩展。

当在WAD中存在动态并发操作时，即使这些线程是交错进行或同时执行的，我们也认为只有当所有线程执行完后，才会去评估输出条件是否满足，也才会发生跃迁动作。

**泳道（Swimlanes）：**UML活动图的泳道元素定义了工作流建模的一个很有意义的方面，因为它力图描述企业活动发生的领域。事实上，虽然一个领域流程必须与一个领域相关联，但某些完成该领域流程的企业活动有可能是在其他领域中执行的。

使用泳道，使得有可能对领域之间的交互作用进行可视化的描述。除此以外，它还力图标识出领域流程中需要参与的功能实体，这些功能实体是其他领域的负责对象。

在下面的图6中，描述了图3中所示WAD的一个带有泳道的局部表示，由于缺乏空间，我们忽略了对象流。

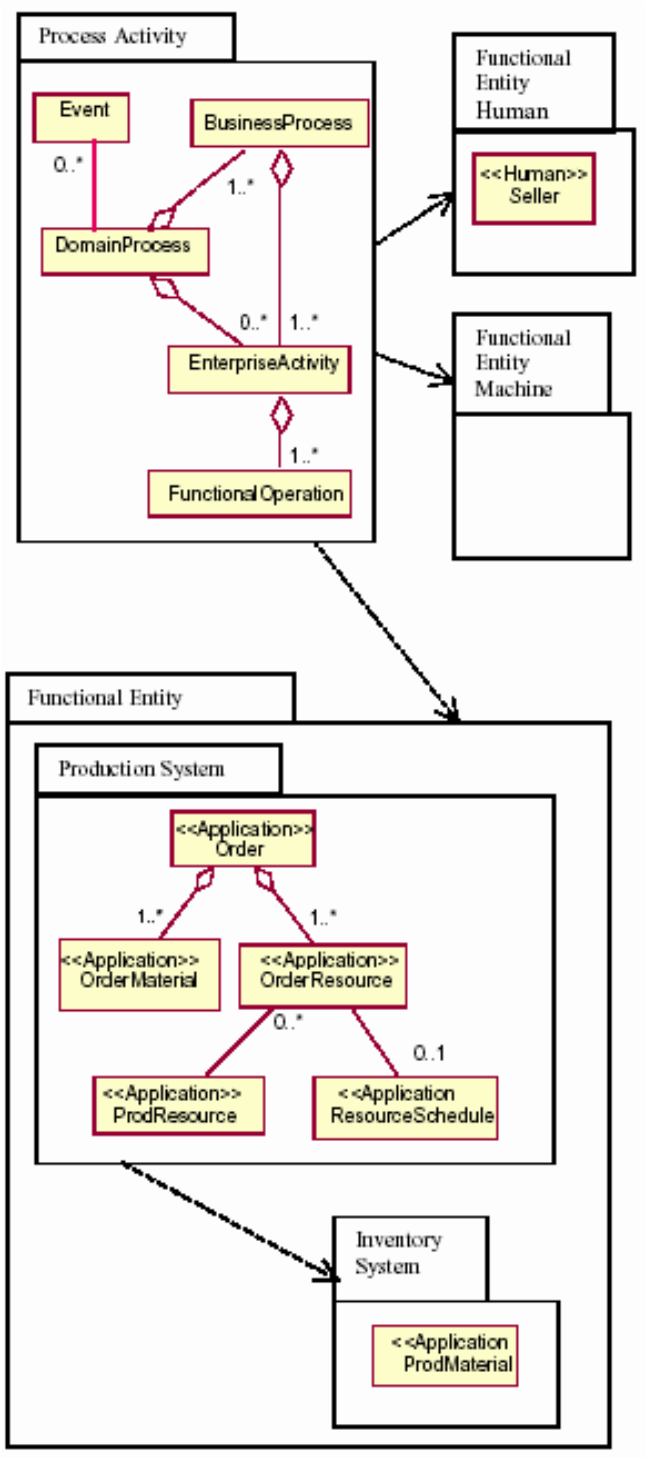



图 5 包和对象

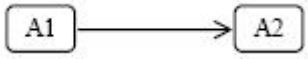
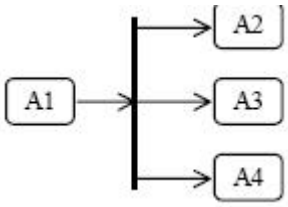
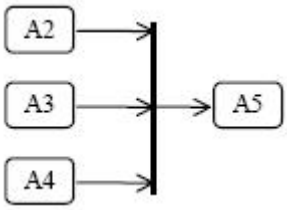
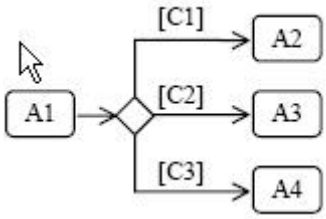
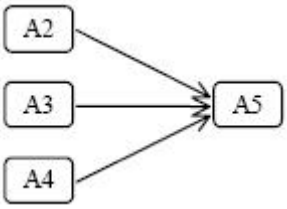
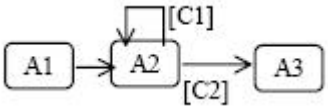
| 可能路线 | 图例（图中假定执行流程是从左至右执行） |
| --- | --- |
| **单线程：**一个活动可以接着一个活动顺序地执行。 |  |
| **与－分支：**控制的单个线程分裂为工作流中两个或多个并行执行的线程，允许同时执行多个活动。 |  |
| **与－联合：**工作流中两个或多个并行执行的线程会聚成一点后进入同一个公共的控制线程。该点是工作流的同步点。 |  |
| **或－分支：**工作流中的单个控制线程根据不同情况形成的一个决策点。该决策点通过跃迁条件来说明（如：C1、C2和 C3）。 |  |
| **或－联合：**指出工作流中的两个或多个可选活动工作流重新会聚到单个公共的活动中，作为工作流的下一个步骤。 |  |
| **迭代：**工作流活动周期涉及一个（或多个）工作流活动的重复执行，直到条件符合。C1和C2是跃迁条件。 |  |

表3 跃迁路径表示

## 5 应用工作流活动图

如前面所解释，在【9】中建议采用用例来确定系统的功能需求和构建系统的业务模型。在我们的方法中，每个用例与一个领域流程或业务流程相对应。为了说明用例的实现，我们通过WAD来描述工作的流程，为每个活动标识出与它执行时有关的分析对象。该分析对象被定义为执行企业活动的功能操作所需要的功能实体。通过这一方法，就有可能集成业务用例模型和工作流模型，因为两者都是使用相同的对象类来实现WAD中描述的用例。

领域流程或业务流程的工作流程的描述是WFMS的输入，并且与企业活动相关联的对象流定义了该企业活动执行时所需要的功能实体，功能实体必须由某个资源分配方法来调度。

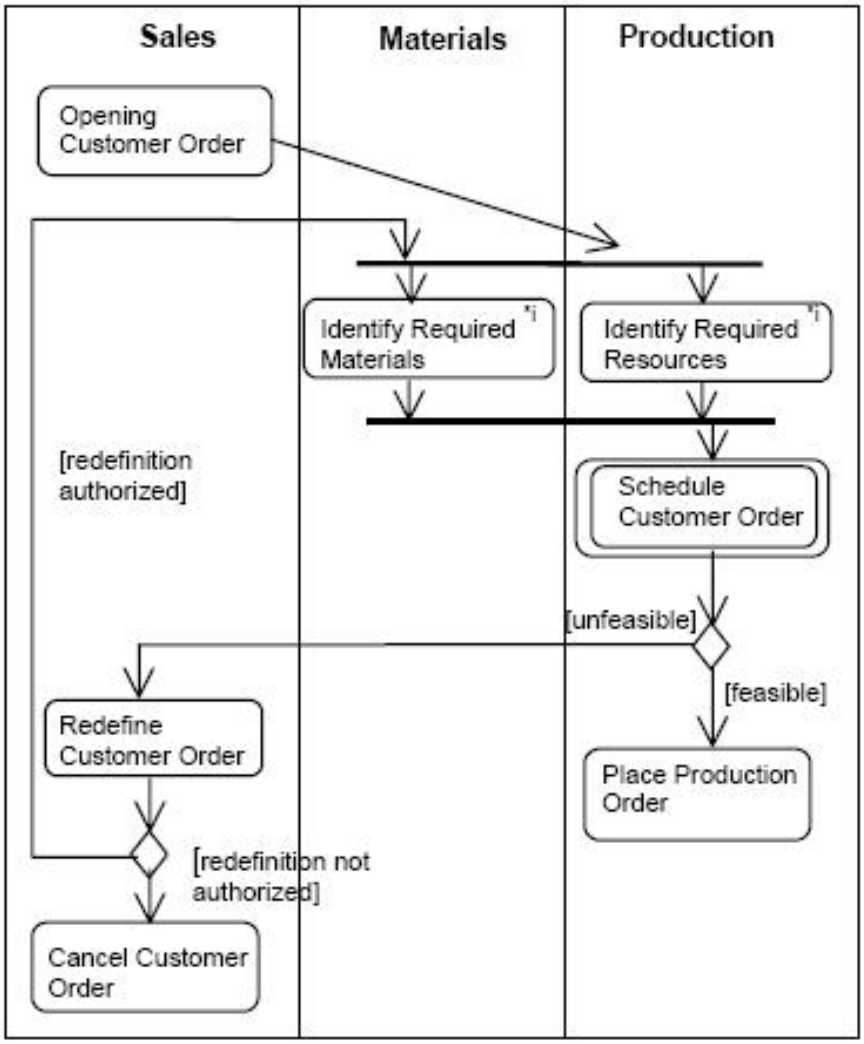



图 6 带泳道的工作流活动图

我们认为资源分配方法必须能够注意到影响生产系统的所有生产事件的动态情形，比如新的生产订单、机器崩溃、原材料故障、生产不合格、工人不合格等等。为此，资源分配方法使用生产系统中的所有状态信息。生产系统的状态基本上是由每个企业活动实例的属性值来定义的。 当执行到它时，该企业活动实例的属性值将从每个正在执行的用来完成该企业活动的功能操作的属性值中派生出来。这些属性值包含为每个功能操作实例定义的临时变量值，如最终期限、估计执行时间和功能操作执行的状态（如剩余时间、功能操作的执行时情形等）等等。

所有这些属性都是在C-Wf模型中进行定义的，包括功能实体实例经过特定的工作列表时所涉及的相关方面，工作列表定义了功能实体应承担的义务以及在每个工作时间周期时工作项的情形（基本可定制或不能定制）。使用WAD，就有可能标识出业务流程和组成领域流程的企业活动之间的关系。考虑到企业活动需要通过功能实体来执行，那么每一个活动都应该考虑规划到它的估计执行时间和所需要功能实体的可用性。整个计划的制订应以保证生产事件的最终期限为准。

## 6 相关工作

考虑到由WFMC和OMG所定义的特性，我们认为关于UML和工作流建模之间的集成，仅有少量工作要作。我们认为WFMC([6][13])建议了一个范围广泛的参考模型，并且，在当前，任何建模方法必须遵循这个参考模型。基于此，我们已经在相关文献上发现了下列相关的工作。

G.Wirtz、M.Weske和H. Giese的方法（[14] [15]）扩展了UML以允许进行工作流建模，该方法集成了标准面向对象结构建模技术，为了详细描述行为特性该方法联合使用了UML图和Petri-Net技术，这一点是通过使用对象协调网络（Object Coordination Nets－OCoNs）实现的。根据作者的观点，他们认为这一集成为所有方面的工作流建模提供了足够的支持。基本上，OCoNs工作于两种类型的地方，事件池（用于标记）和资源池（用于资源）。为了描述行为，作者提出了三种类型的网络：协议网（PN），服务网（SN）和资源分配网（RAN）。PN说明了所提供服务的外部可见行为。SN描述了当执行单个服务时的详细工作流，而RAN描述了资源管理。G.Wirtz、 M.Weske和H. Giese断定，只要严格遵循这一体系，将可以使得设计师以一种可伸缩的方式处理复杂的工作流系统。与我们的方法（WAD）相比较，我们建议采用UML模型作为解决方案，基本上使用版型来对工作流建模。同时，我们的方法对线程的分裂与合并也进行了清晰的建模，并且描述了可以被引入工作流模型中的同步和异步动态并发特性。为了达到这一目的，我们建议对UML活动图及其与其他UML描述工具之间的关系进行更深刻的理解。虽然UML+OCoNs的方法看上去要比WAD在体系上更丰富，但我们认为WAD对精通UML的设计师来说要来得更熟悉一些。

P. Hruby（[7][8]）介绍了一个使用UML作为WFMS的规格说明的方法。基本上，它定义了四种版型，分别对应于业务对象，业务流程，工作流和团队角色。此外，一个工作流应用程序的建模是采用如下方式完成的：UML的静态结构图用来描述团队结构；UML的顺序图描述了业务流程的实例以及业务流程与执行者之间的交互；UML的用例图描述了业务流程之间的静态关系；UML的协作图也描述了业务流程和执行者之间的交互与关系；UML的活动图可以描述业务流程之间的正确顺序。P.Hruby方法与我们的方法之间的不同是：(1) 我们的方法采纳了Wfmc参考模型，从而定义了一个更大的工作流结构集；(2) WAD描述了每个领域流程的行为。我们认为，如果没有对UML工具之间的关系进行清晰的语义上的定义，将很难映射Wfmc结构到UML上，此外，[14][12]还陈述了UML中的一个问题。

L. Baresi et al.（[1]）对WIDE工作流开发方法学作了一个简要描述，WIDE有三个主要步骤：(1)分析；(2)WF设计；(3)映射到目标WF系统。对分析步骤，WIDE采用了UML工具：用例，场景和序列图。只有当具有重要的复杂层次这种情形时，WIDE才使用活动图。对Wf设计，它用于WIDE的Wf模型。与G.Wirtz et al.的方法类似，WIDE方法对流程的描述也是通过一个特定的体系来完成的，而不是通过使用某些UML工具。虽然WIDE方法看上去已经处理了构造工作流系统所需要考虑的所有相关方面，然而使用它自己专有的形式化的建模方法来描述每个业务流程的动态方面也许对一个精通UML的设计师来说并不熟悉。

## 7 结论

本文介绍了对UML活动图的一个扩展，称为工作流活动图(WAD)。该图对于描述工作流模型是十分有用的，该工作流模型描述了领域流程中包含的活动、业务流程和企业活动。而且，对于每一个活动，该模型标识出了执行它所需要的资源－功能实体。这些资源被标识为来自于在分析模型中建立的企业业务模型的类图中的分析类。

WAD通过使用版型基本上重新改造了定义在UML活动图中的元素，以便描述C-Wf模型的主要概念（[3]）。从而，WAD中定义的每个元素都被一个与C-Wf类之间建立了关系的版型所描述。WAD描述了工作流模型，该模型标识了活动与执行它所需要的资源，定义了它们的关系与次序。同时，WAD也描述了工作流中的偶然性的动态并发场景，该场景可以是同步或异步执行的线程。

我们所提出的方法的贡献在于:(1)使用C-Wf概念将工作流模型作为一组用例来描述;(2)将C-Wf对象作为版型使用以便描述一个领域流程和它的业务流程及企业活动;(3)通过采用工作流建模的原则对活动图的表示进行了改进；(4)利用了UML是OMG的标准这一事实，并且它的使用正在快速增长之中。

我们的下一步是构建一个执行环境，它由一个基于JAVA的资源分配流程系统（基于[2]中所介绍的框架）和一个支持资源分配管理的工作流设定服务（Workflow Enactment Service）所组成。

## 8 参考文献

[1] Baresi, L. et al. “WIDE Workflow Development Methodology”. In: Intl. Conf. On work Activities Coordination and Collaboration (San Francisco-CA, USA: Feb. 22-25 1999). *Proceedings…* San Francisco: ACM Press, 1999. p.19-28

[2] Bastos, R.M. *Resource Allocation Planning using Multi-Agent Systems*, Porto Alegre - Brazil: PPGC-UFRGS, 1998.(PhD Thesis, in Portuguese)

[3] Bastos, R.M. & Ruiz, D.D.A. “Towards an Approach to Model Business Processes using Workflow Modeling Techniques in Production Systems”. In: HICSS´34 (Maui -Hawaii: Jan.3-6 2001) *Proceedings…* Los Vaqueros - CA:IEEE Press, 2001. (CD-ROM/Abstracts Proceedings)

[4] Booch, G. et al. *The unified modeling language user guide*.Reading, MA : Addison-Wesley, c1999. 482 p.

[5] Fowler, M. *UML distilled : applying the standard object modeling language*. Reading, MA :Addison-Wesley, c1997.183 p.

[6] Hollingsworth, D. *The Workflow Reference Model*. Hampshire - UK: WfMC, Jan. 1995.

[7] Hruby, P. “Specification of Workflow Management Systems with UML”. In: OOPSLA-98 Object-Oriented Workflow Management Systems Workshop (Vancouver - Canada: Oct. 18-22 1998). *Proceedings…* Vancouver - Canada: ACM Press 1998.

[8] Hruby, P. “Structuring Specification of Business Systems with UML”. In: OOPSLA-98 Business Object Workshop(Vancouver - Canada: Oct. 18-22 1998). *Proceedings…*Vancouver - Canada: ACM Press 1998.

[9] Jacobson, I. et al. *The unified software development process*. Reading, MA: Addison-Wesley, c1998. 463 p.

[10] Object Management Group. *Unified Modeling Language (UML) 1.3 specification.* Available at: http://www.omg.org/cgi-bin/doc?formal/00-03-01.pdf.gz.(access date: May 2001)

[11] Rumbaugh, J. et al. *The unified modeling language reference manual.* Reading, MA : Addison-Wesley, c1999.

[12] Schmidt, M.T. "Building Workflow Business Objects". In: OOPSLA'98 Business Object Workshop. (Vancouver, Canada: Oct. 18-22, 1998). *Proceedings* Heidelberg: Springer, 1998.

[13] Workflow Management Coalition. *Terminology and Glossary*. Hampshire - UK: WfMC, Feb. 1999.

[14] Wirtz, G.;Weske, M.; Giese, H. "Extending UML with workflow Modeling Capabilities". In: 7th CoopIS (Eilat - Israel: Sept. 6-8 2000) *Lecture Notes in Computer Science* v.1901, Heidelberg: Springer, 2000. P. 30-41.

[15] Wirtz, G.; Giese, H. "Using UML and Object-Coordination-Nets for Workflow Specification". In: IEEE Intl.Conf. on Systems, Man, and Cybernetics (SMC'2000)*Proceedings*…Nashville-TN USA: Oct. 8-11 2000.

[16] Zelm, M.; Vernadat, F. B.; Kosanke, K. "The CIMOSA business modelling process", *Computers in Industry*,Amsterdam: Elsevier, Oct. 1995, pp.123-142.

征稿

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

The Addison-Wesley Signature Series

# Patterns of Enterprise Application Architecture

中译本即将上市

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

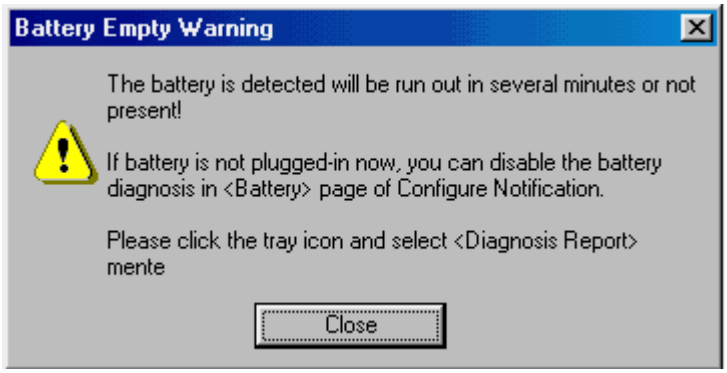
UMLChina 负责中译本最后审校，并指定为训练用教材

# 糟糕界面集锦（补遗）

iarchitect 收集整理，雷立辉 译

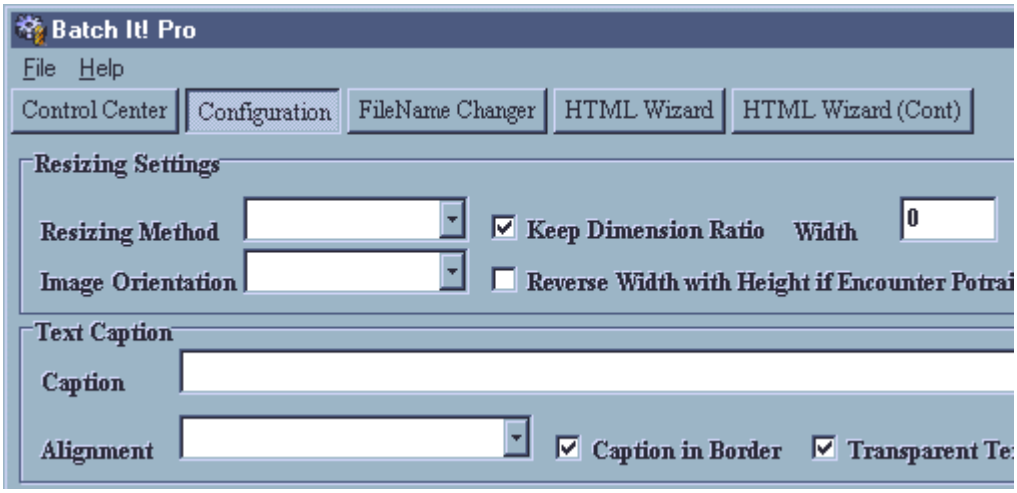
查看评论

Frans de Haan 给我们发来了从一个名字叫 *MTC Diag* 的电源管理软件的截图，这个软件是 Saxum 8004 笔记本上的随机软件。对话框的本意是警告用户电池电量快要用完了，没想到这条信息却没能够向用户表达这一点。




**********

Abbagail Winters 发来了 *BatchIt Pro* 软件的一幅截图。尽管设计者尽力使界面变得有条理一些，但结果还是一塌糊涂。

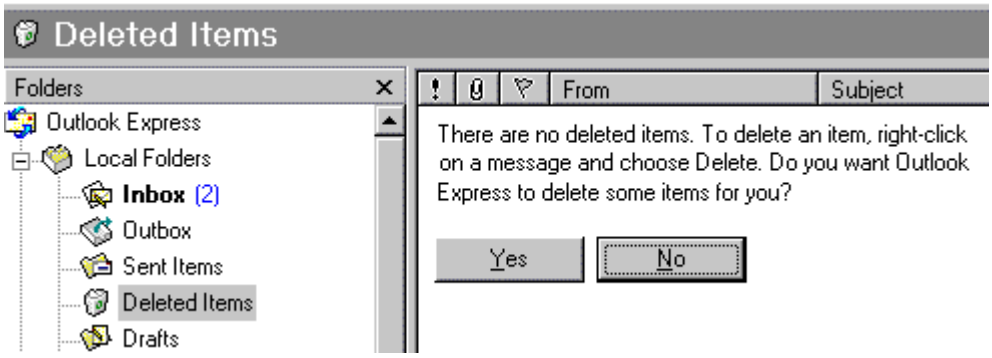



*BatchIt Pro* 的截图（着重强调）：

配置界面由四部分组成：缩放设置， 标题设置，水印设置及保存设置。尽管与缩放设置无关，设计者还是画蛇添足的添加了一个图像定向功能即允许将所有的图片旋转。(译者注：这个例子告诉我们,界面设计的每一项功能应该是目的明确,不允许有与大功能分类无关的内容)。

**********

Owen Rudge 发送了一幅从 Microsoft 的 *Outlook Express 5.0* 截取的奇怪图片：



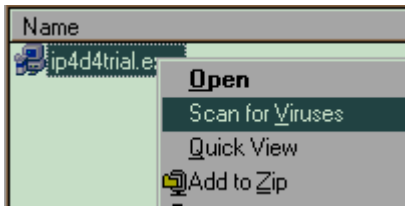

“当我打开 Microsoft Outlook Express 5 删除项目的文件夹时，我发现了一个奇怪的提示信息：它告诉我在删除项目文件夹是空的但还是提示我是否想让 OE（Microsoft Outlook Express）删除一些邮件。这是什么消息啊？！？难道它还要帮我随机删除一些邮件？我并不想知道它究竟会删除什么东西，所以我立即按下了‘NO’的按钮!”

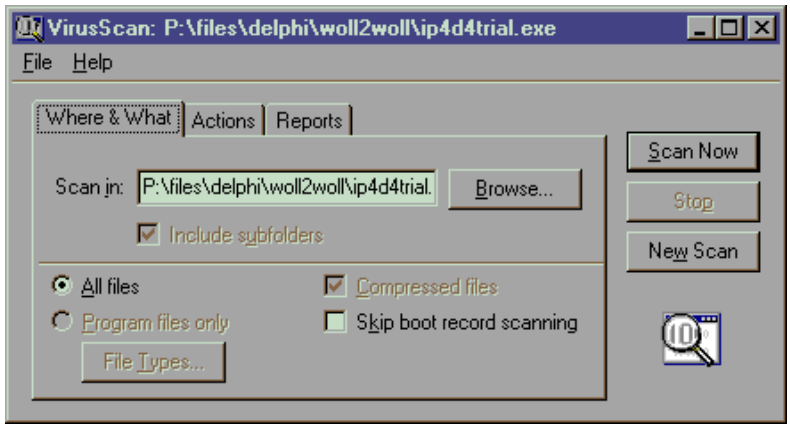
**********

Bo Bichel Nørbæk 在使用 *Reflection FTP* 想刷新文件列表时按下 F5 发现了这个对话框。可以理解 Bo 的想法：“为什么这个玩意儿不能立即刷新全部列表而是弹出这个傻傻的对话框？!!!”


**********

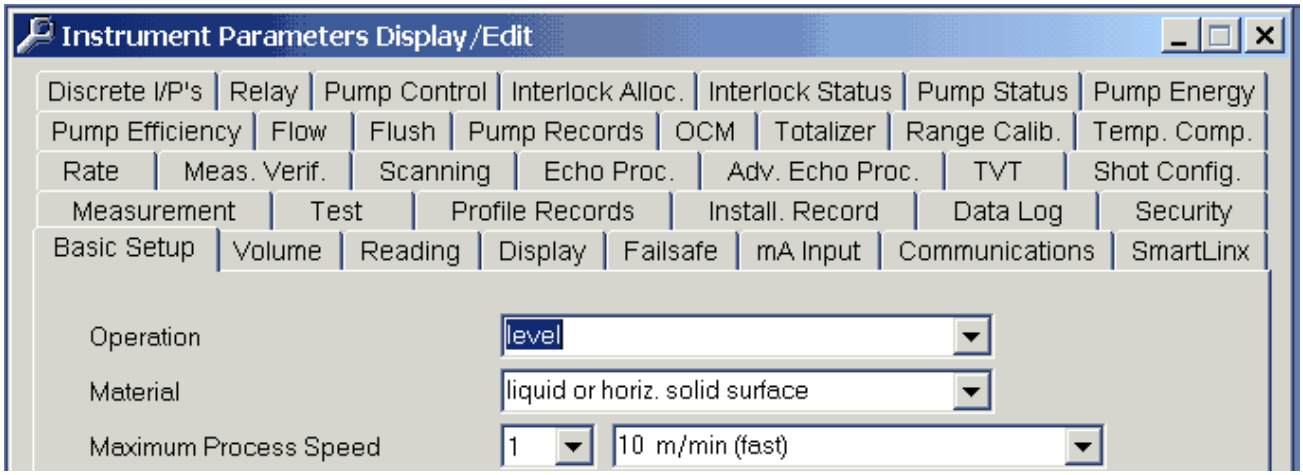
Ken Rachynski 问了一个关于 *VirusScan* 网络关联的类似问题：这个程序在文件管理器的右键菜单添加了一项，想让用户使用它在必要时扫描被选的文件或文件夹。但是，当用户选择扫描后，程序不是直接扫描而是弹出一个对话框询问用户是否立即扫描？




**********

一个匿名游客来自于 Milltronics' *Dolphin Plus* 的一副图片。这个软件是一个用于工业水准和流控传感器（industrial level and flow sensors）的配置工具。这也是分页对话框糟糕界面集锦的候选者之一。




"Waldo 在哪儿？"（译者注：一种训练眼力的著名儿童游戏）

**********

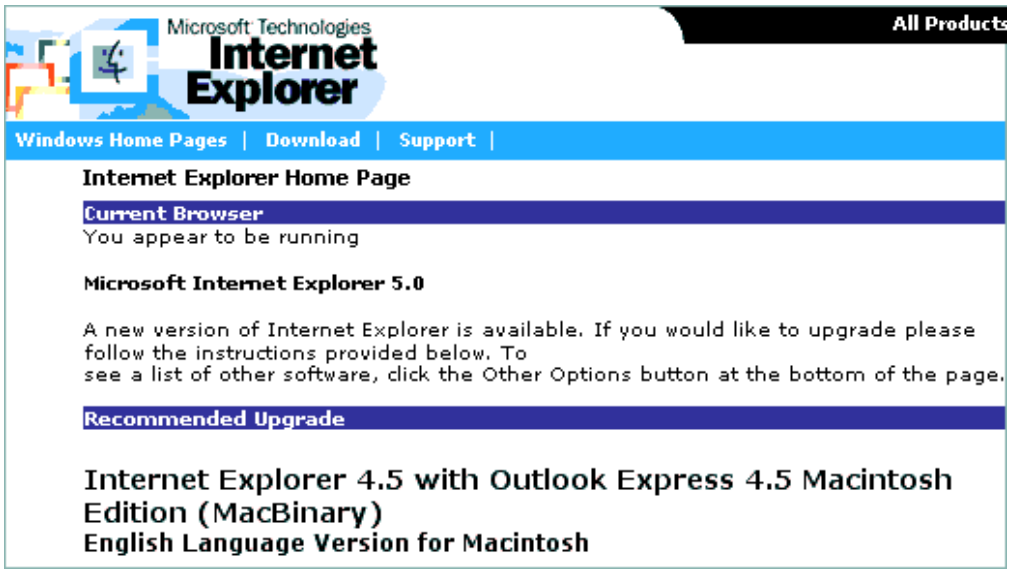
Ben Oram 发来了来自于 OneApp 软件包中 *English-German* 程序的一幅图片：


这幅图片存在一些错误使用 MacOS 的界面设计风格的问题，但最值得一说的还是这：这个对话框的起初用意是显示给定词语的查询结果，但是没有考虑查询结果的词条数（这个例子中有 105 个查询结果）。界面没有显示一个滚动条而是使用了一个自定义控件。通过使用两个向上和向下按钮及一个小点来显示用户现在所处的列表位置。

既然明白标准控件功能强大有效，但糟糕的是还有人想使用别的与之相比功能欠缺的自定义控件。标准滚动条除了可以一次只滚动一行之外，还可以让用户前后翻页，拖拽滚动控制条翻到想到的地方。此外，标准滚动条的控制条比起像是错误画上的像素（译者注：上面设计者使用的小点儿）明显而且形象多了。

**********

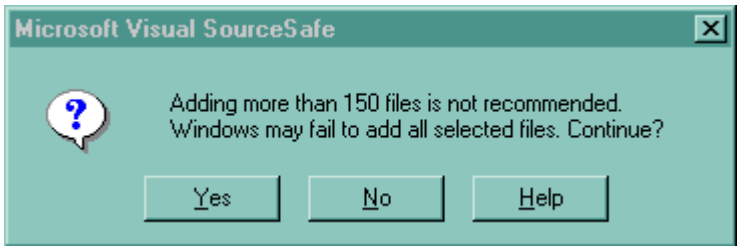
好像微软的人工智能决策系统发挥作用了。Pierre 发送了这幅他访问 http://www.microsoft.com/ie 时的页面截图：



难道 *IE4.5* 比 *IE5.0* 更好吗？（译者注：图片上显示当前用户使用的是 IE5.0，但是系统提示用户应该升级到 IE4.5）

**********

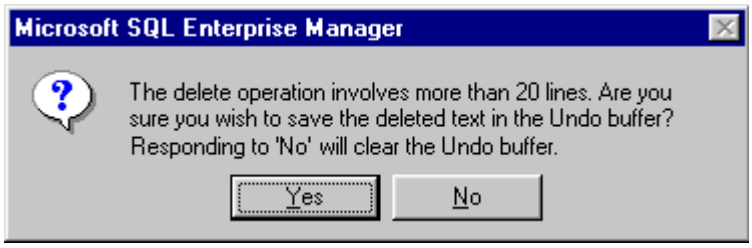
**Rick Lamoreaux** 发来了他使用 Microsoft's *Visual Source Safe* 在工程中添加文件时碰到的一个对话框截图。




Rick 指出:这个消息表明设计者对程序运行的性能不太有把握，所以给出这样一句模棱两可的警告以防出错。给出这样一个消息“这可能不工作，您确信要这样做吗？”本身就是将程序的错误责任推脱而强加于用户的一种方式。（译者注：这个对话框中没有这样一句话。对话框中的消息大意是说：不推荐用户一次添加超过 150 个文件。否则程序可能不能将所有文件加入，要继续吗？）

**********

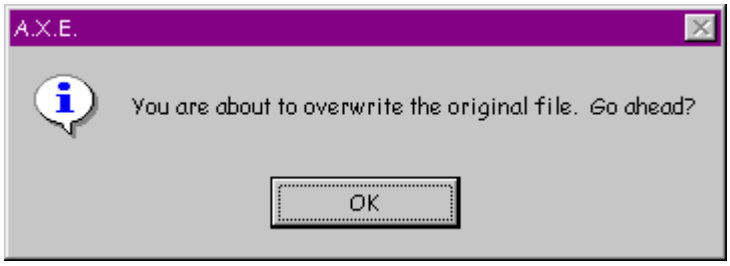
**Gary Walker** 使用 Microsoft's *SQL Server 6.5 -- Enterprise Manager* 时偶然删除了几行语句，他便看到了下面这个对话框：




如果您从查询窗口选择 20 行文本以上，并且按下删除键，您将收到这个显然有用的出错信息。但如果您改变主意，想选择“否” 避免删除。但是，消息却与您想象的结果不一致。选择“否”实际上还是删除了选择的语句并且清空了恢复缓冲区。所以不能避免用户取消错误的删除命令。

**********

Søren Erland Vestø 发来了其它方面都很好的一个具有优秀的十六进制编辑功能的自由软件 *A.X.E* 截图：




Søren 打开了一个文件但没有编辑，偶然使用热键（CTRL-S）保存它时发现了这个消息。这事实上说明软件作者根本没有给予与用户交流的机会。“不，这不太好”，用户只想知道为什么还会存在这样的消息。

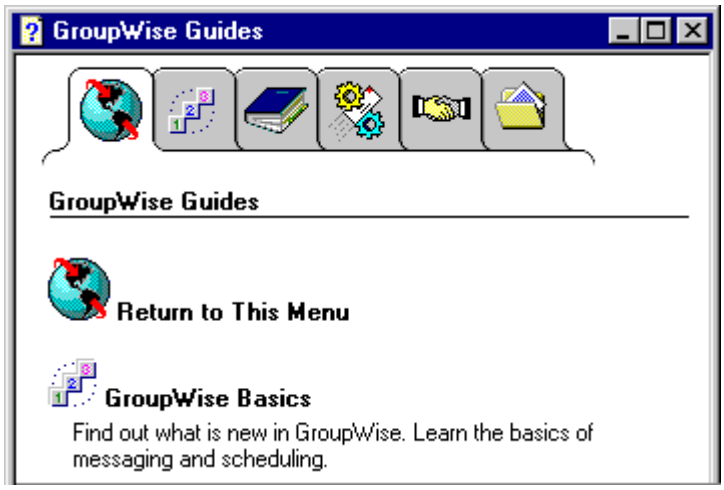
**********

安装 *Office 2000* 时，David Nottage 发现微软的程序员还没学会如何撰写一个有用的错误信息：


这个消息毫无意义：因为没有给出具体错误提示，没有提供解决方法，甚至没有告诉用户出错日志存放在哪里。

**********

Steve Dieke 发来了这幅图片用来解释 *Novell GroupWise 5.1* 中对属性页缺乏设计：

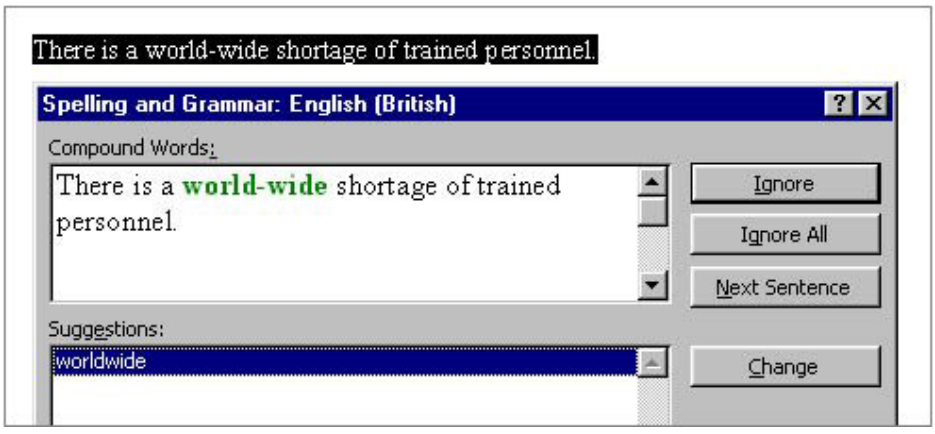


除了缺乏美感之外，这个界面还存在其它的一些缺陷。最严重的便是标签上的图标意义模糊。意识到这一点，设计者就多添加了一个属性页用来专门解释图标的含义。但是，如果您从此页切换到其它页面，您就会丧失图标意义的解释。尽管认为工具条提示不是用来解决含义模糊图标的意义的好方法，但是如果 Novell 想到这种方法的话，也不失为一个解决问题的方法。当然最好是添加文字标签来取代当前这种术语解释属性页和工具条提示。

**********

**Timothy Tan** 在 Microsoft's *Word 97* 发现了这个情况。当语言设置使用“英国英语”检查文档语法时，拼写和语法检查功能提示“wordwide”不在字典中而应该改为“world-wide”。

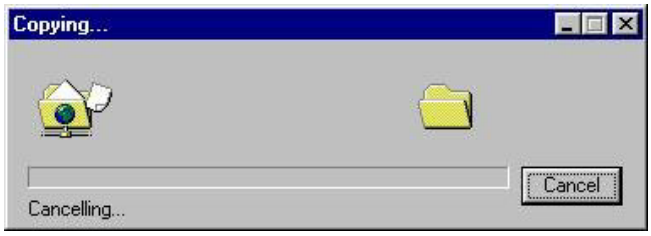

Tim 接受了这个单词检查的更正建议。但他又现这个 *Word 97* 的语法检查又提示将“worldwide”建议改为复合词“world-wide”更合适一些。




Tim 发现只有将“worldwide”添加到用户自定义字典中或者使用“美国英语”字典才能解决这个问题。

**********

Greg Funk 发来了这个让他困惑的微软 *Internet Explorer 5.0 FTP* 程序出现的问题：




这个问题足够可以使那些具有哲学头脑的用户争论半天。“当一个人取消他正在取消的操作将会发生什么？”（原话）Greg 说可以通过以下步骤重现这个问题：

1. 选择一个文件并点击右键菜单‘拷贝到’；

2. 指定目标文件夹；

3. 让系统开始拷贝文件；

4. 点击取消；

请注意其实这样拷贝并不能取消。Greg 最后只能点击右上角“关闭”按钮来关闭这个窗口。Greg“通过点击关闭按钮来取消了正在取消的操作” 。

**********

诚然我们可以将任何一个真实物体作为人机界面的参考，但是我认为使用手持遥控器来作为人机界面是错误得无可救药的决定。但是创新工作室就用它作为 *Creative PC-DVD* 软件的用户界面。

访客 Ilari Sani 发来图片及一篇深刻的批评文章来说明为什么从根本上避免使用这种使用真实世界的比喻的理由。

其它受到这种使用真实物体形象作为界面而被收录入糟糕界面设计的软件有：

1. 苹果的 QuickTime 4.0 播放器

2. ReadPlease 2000 文本语音合成工具器

3. IBM 的 RealCD player

4. IBM 的 RealPhone telephone application

这些评价指出了许多对于使用真实物体作为软件界面的弊端：

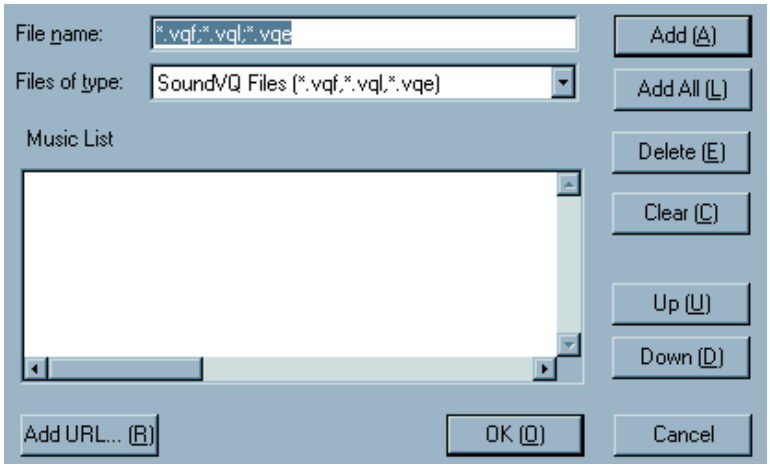
当您拷贝框内的东西时您不会想到框子外的事情（You cannot think out of the box when you are trying to copy a box）。

这样会使用大量的技巧来实现这个比喻性界面但却用很少去想怎么让用户一下子上手。真实世界的用具是针对如何使用手指操作方便而不是使用鼠标及键盘操作而言的。工业设计者设计时考虑如何获得视觉感受，而软件界面则强调如何让用户容易使用。

**对采用真实物体形象设计软件界面说不！**

**********

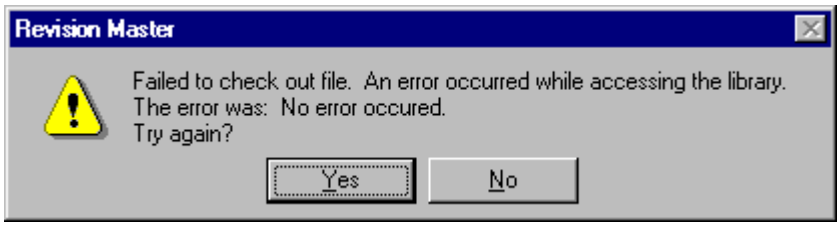
**Zing Zing Awungshi Shishak** 建议我看一下 Yamaha 的 *SoundVQ player*, 认为这个界面也可以作为糟糕界面集锦的备选者之一。当我看到这个软件中打开文件对话框之后便觉得此言不虚，下图便是这个对话框的一部分。



*SoundVQ*开发者见过 windows 程序的界面吗？

**********

**Tom Campbell** 使用 *Revision Master* 登出并编辑文件时碰到了这个消息：




正如 Tom 所嘲笑的那样：

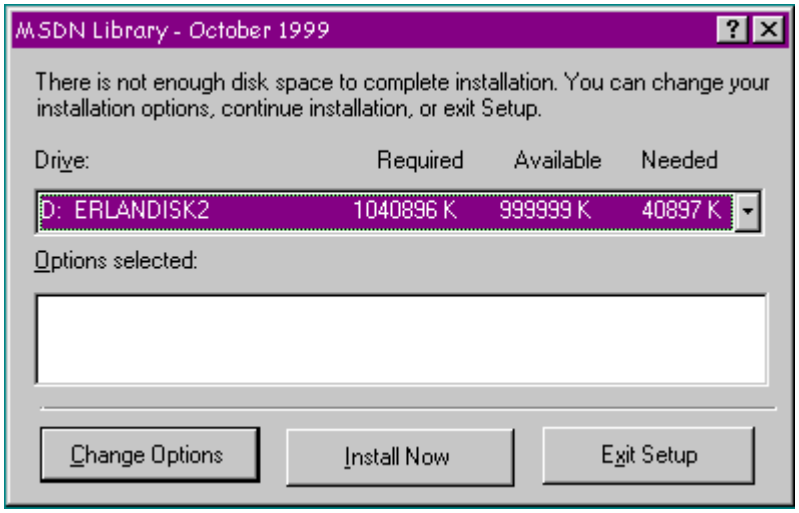
奇怪，这样自相矛盾的话竟如此的富有艺术魅力。两个不同拼写方法的“occurred”造成一种压力，再加上错误消息告诉这儿没有错误，一下子让这充满禅学意味的话变得玄妙而又优雅。

**********

**Jean-Marc Orliaguet** 提供了这种“警告成功”的消息并不是 windows 操作系统独有的。下图便是从 Mac OS X 服务器上 *NetInfo* 配置程序中得到的：


**********

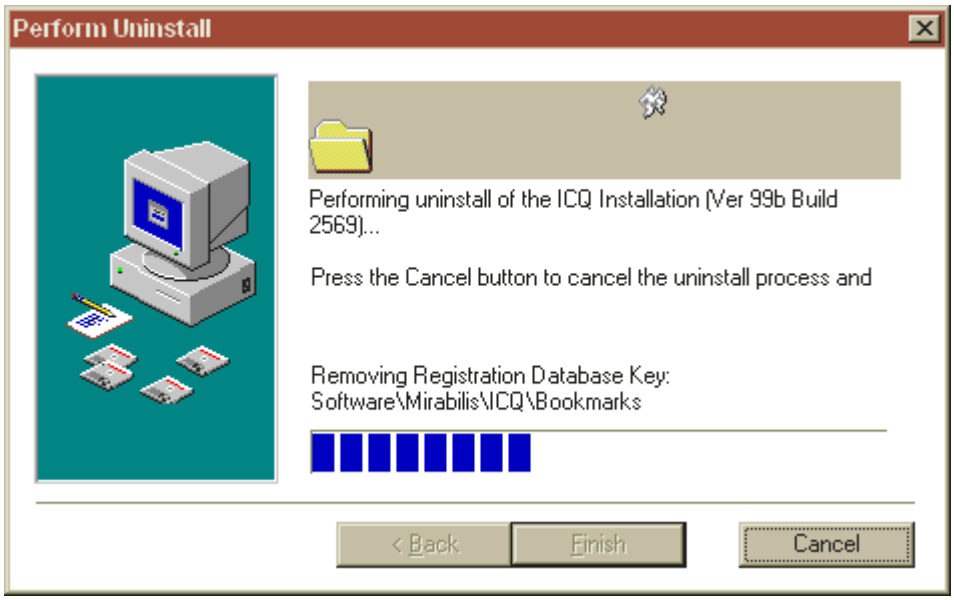
**Søren Vestø** 安装 *Microsoft Developer Network* 时碰到了这个问题：




问题出现在显示的可用存储空间只有 999999K，尽管 Søren 说他的磁盘还有 2.4 GB 可用空间。有趣的是程序一方面告诉用户没有足够空间不能安装，但另一方面却让用户继续安装。最后实际上安装成功了。

**********

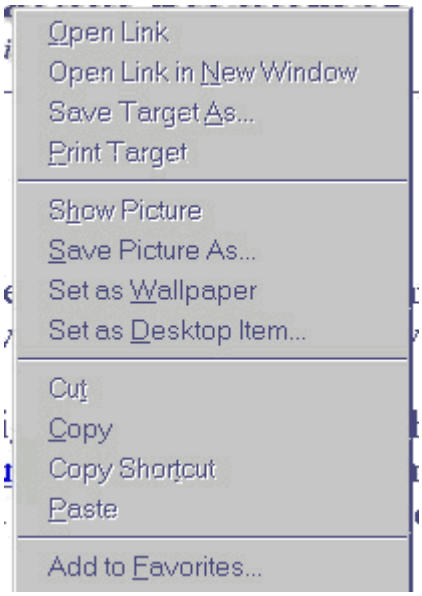
可以理解，Rich Adams 在卸载 *ICQ* 时迷惑不解：




*并且…(and…)订购个比撒饼？格式化磁盘？究竟要干什么嘛？*（译者注：界面上 and 以后的话没有显示出来）

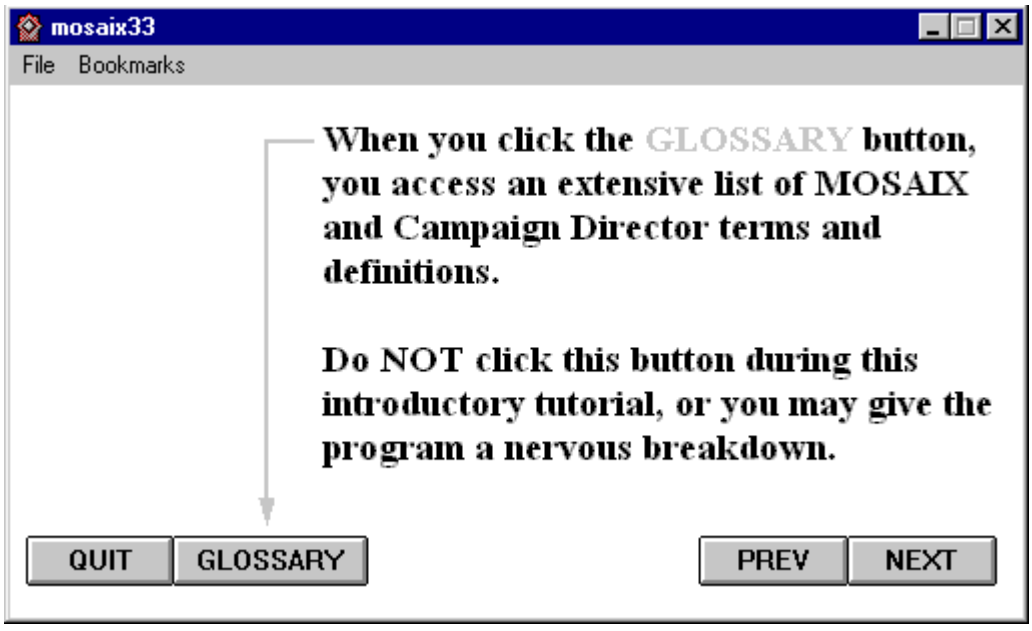
**********

Eric Hartwell 发现了 *Internet Explorer* 上下文菜单的一个设计问题。Eric 习惯于使用上下文菜单中的“图片另存为”命令保存网页图片。但是，由于菜单项“设置为背景”紧挨着它，所以有可能使用户错误的点击。问题在于“设置为背景”是立即见效而不可退回。程序并没有像“设置为背景”前后的菜单一样提示用户，所以如果用户做错了也无法恢复，必须到别的地方去设置去做很多步才能恢复原来的模样。（译者注：现在的 IE6.0 中不是这样的了）




**********

这个典型的失败设计例子来源于 *Mosaix* 用户手册。这个服务支持软件被众多公司使用。



成熟的软件开发者会认为这个消息非常荒谬并且知道只要让这个“GLOSSARY”按钮失效便可以了。然而这个用户手册并不是 *Mosaix* 开发小组所写，而是另一个从事基于计算机用户手册设计的专业公司 *MediaPros* 所为。这可以从 *MediaPros* 公司主页“成功案例”的链接中看出（译者注：这个链接已经失效）。这样说来，MediaPros 或许要重新考虑以这个教程来作为样例来展示自己的专业水平了。

**********

**Richard Sheridan** 提供了两张图片进一步提供了反对使用真实物体形象作为人机界面的证据。这两张图片都是从 Hewlett-Packard 计算机的随机多媒体软件 *AudioRack 32* 采集的。


*…就像 IBM 的"Real CD"，Audiorack 像一个 CD 播放器。但在细节之处却是极不协调。与 Windows 平和的环境相比，简直就像一个化妆成大鸟的人突然出现到一个正式宴会中。是的，设计者已被形式所限，让我们不能容忍：在“Mixer”按钮旁边为什么留有一片恶心的空白？其它的呢？我特别喜欢这个叫做“Steath”的按钮，地球人都叫它“最小化”且放置在窗口的右上角。但实际上提示信息显示“进入缩小模式”。还有那在显示面板上的那些可爱的 LCD 符号，代表什么啊？谁会注意到这些啊？*

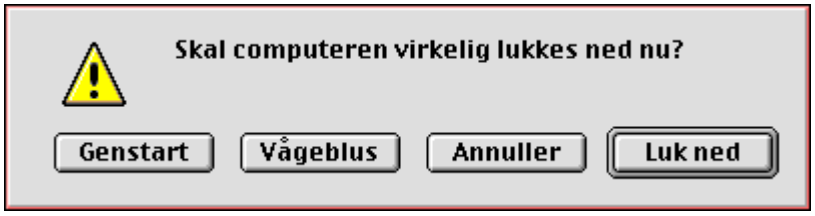
第二张图片就更有意思了。*AudioRack 32* 使用了自定义控件“Rack Control”。从用户角度来看，这个控件使用对用户隐藏的控件来让用户拖动。


如 Richard 所说，Rack 控件看起来是由特别具有艺术天才的人特意防止用户轻松使用而设计的。Richard 找不到控制这个控件的方法。只不过他偶然将光标移动到“把手”这个地方时发现了这个“功能”。（译者注：图上提示说通过单击这两个把手可以让这个 Rack 控件滚动）

Martin "Jay" Sundstrøm\发来了几幅图片批评操作系统 MacOS 8.6 丹麦语版的本地化问题。


操作系统想提示信息“Arkiverer kommentarer i informationsvinduer”（正在信息窗口中保存注释），但由于窗口太小了，没有足够空间来显示字符串。

当丹麦用户重启系统时，他们碰到了这样一条消息：



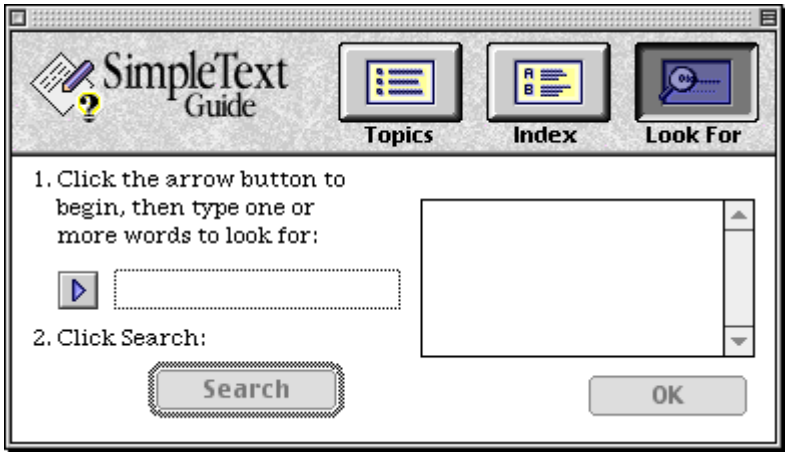

使用英语版的操作系统时，用户用“R”键作为“重启（Restart）”按钮的快捷键。但不幸的是，丹麦用户也不得不使用“R”键来作为重启的快捷键，尽管丹麦语的“重启”叫“**G**enstart”。（译者注：作者的意思是应该使用“G”来作为快捷键）。

**********

Chris Kostiw 发来了一幅图片，这是从 Mac 版的共享软件 Risk 中截来的。如果用户退出游戏但没有存盘，软件就会弹出这个消息。Mac 界面设计手册上说明了对话框中“Yes”和“No”的使用情况。但如果设计者像这幅图片一样设计，这个 Mac 界面设计手册就得重新改写了。


**********

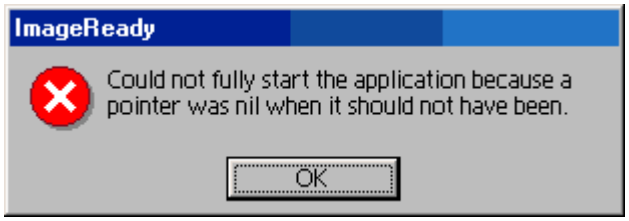
James "Kibo" Parry 发来了这幅从苹果软件 *SimpleText* 截来的图片。这个对话框精心设计了一个通过用户输入查找项来查找的输入框。但遗憾的是在他（她）输入之前，他们必须点击箭头按钮来是这个输入框有效。




似乎设计者认为必须保护用户使得他们避免不小心错误地在查找框中填写了查找项。

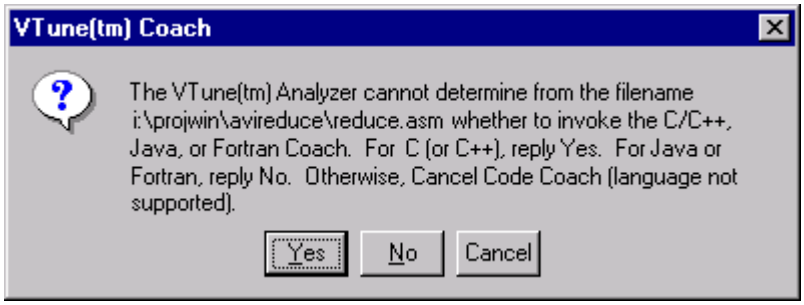
**********

这个无效提示的例子是 Joanna Southerland 在 Adobe 的软件 *ImageReady* 中发现的：（译者注：信息提示由于一个指针无效而不能启动全部软件功能）

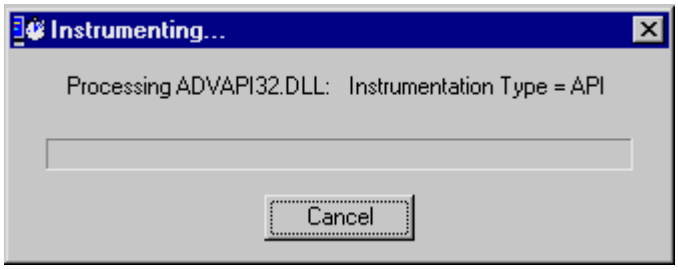



**********

Avery Lee 发来了这幅从 Intel *VTune* 4.0 性能测试软件中截得的图片。Avery 在他的汇编语言文件中调用“Code coach”时发现了这个消息。他发现 VTune 只支持 C/C++，Java 和 Fortran：

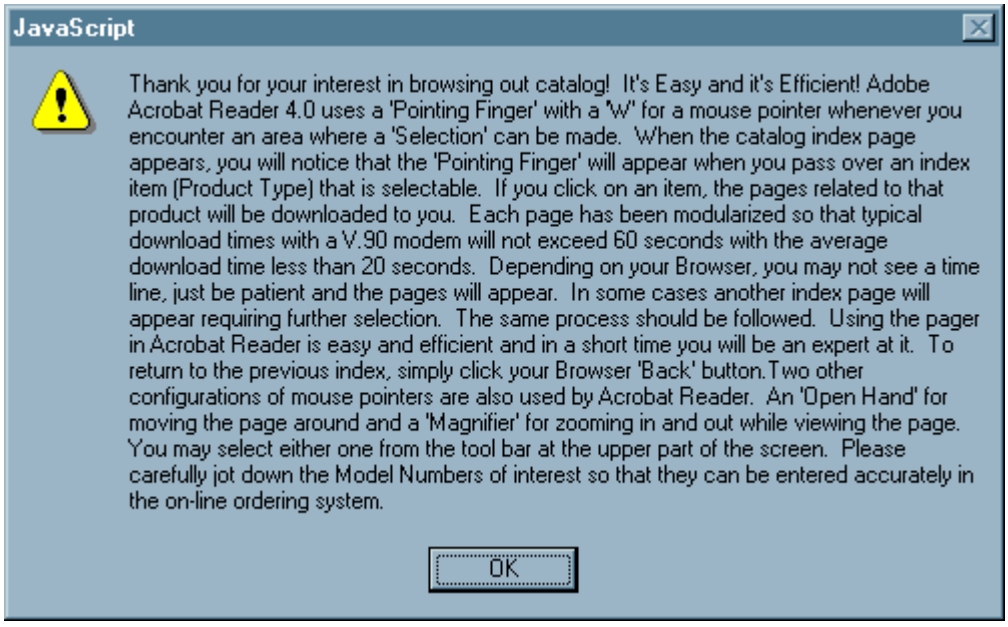


*VTune* 提供了另一个没有进度显示的对话框来表明他们的界面设计能力有问题：



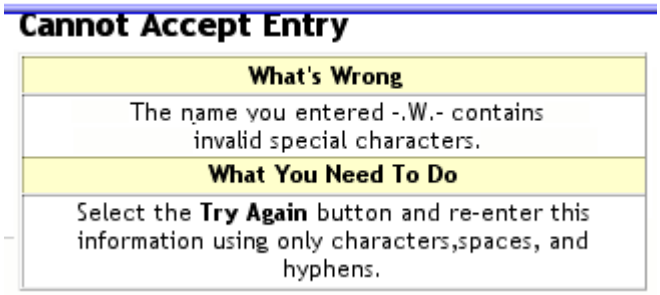

**********

Michael Wilson 发来了这个提示信息冗长的例子。这是他在网址 http://www.midwest-microwave.ltd.uk/index.htm 中阅读 *Midwest Microwave* 的在线分类目录中碰到的：




**********

James "Kibo" Parry 发来了这幅图片。这是他注册美国在线网站时碰到的：



Kibo 只是输入“W.”（不是“.W.”或“-.W.-”）作为他中间名字的缩写。显然，“W”或“.”被认为是“无效的字符”。Kibo 不知道他只能输入“字符”。

**********

看起来苹果公司已经考虑过了重新设计 QuickTime 4.0 的用户界面。

Steve Jobs 在近期举办的 MacWorld 展览会上演示了最新的软件界面。从图片上可以看出， 许多我们以前所指出的界面不足都已经改正了。最值得注意的是原先的“滚轮”音量调节器已经改成更为适宜的滑动调节控件；“播放”按钮也从 steroinds 上消失；还有，设计者也把“快进”和“倒退”这两项功能从“高级”控制中取消了。尽管从图片上看得不太清楚，但还是可以肯定苹果已经采纳了我的建议写了“收藏抽屉”。最终，设计者从像随身听的界面中摆脱出来，使用了一些标准窗口管理控件。但还是有点儿不幸，他们没有使用更好的图标来表示功能，苹果 MacOS X 设计者认为草莓红，橘红，石灰清及葡萄紫水滴按钮足够了。

草莓红　　关闭

橘红　　最小化

石灰清　　最大化

葡萄紫　　“单窗口”模式

*许多人都觉得这样不如这样：*

最左的水滴　　关闭

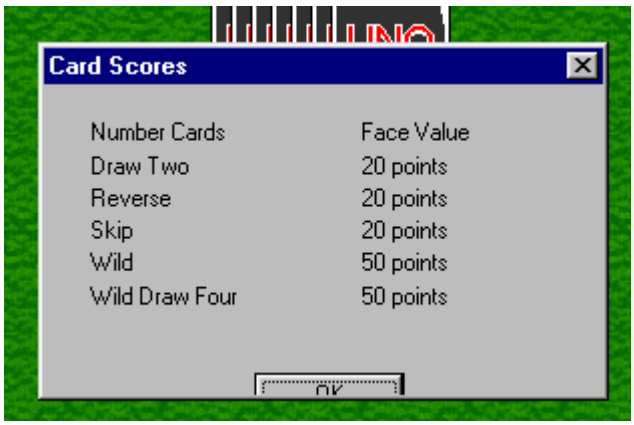
左起第二个　　最小化

左起第二个　　最大化

最右水滴　　“单窗口”模式

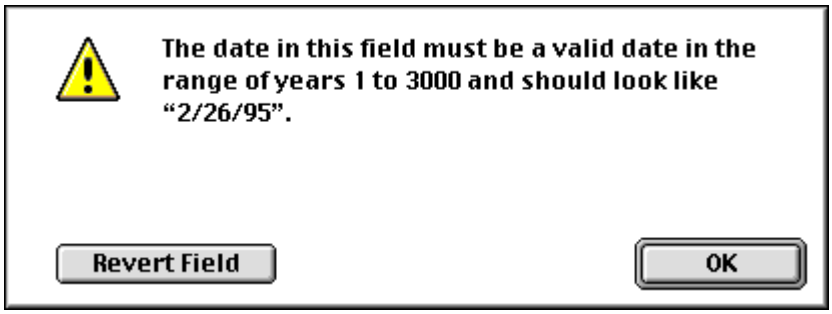
**********

我开始觉得应该给这个网站再添加一个新的部分，名字就叫“多动脑子”。Chad Lowry 寄给我下面这张对话框的截图，是他在游戏“Uno”里看到的。要求开发者再花一点时间运行一下他写的程序这一点儿都不过分吧。




**********

这是 Jeff Howser 在“交通办公管理（*Traffic Office Manager*）”软件里遇到的对话框。

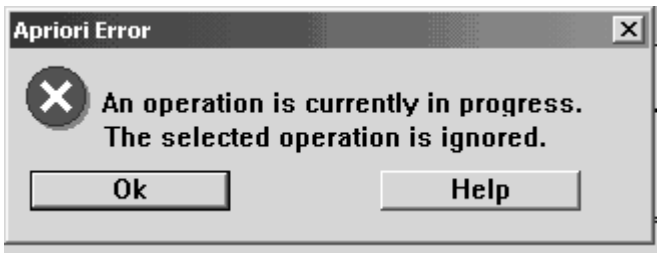


在我们许多人被千年虫困扰的时候，这个软件的作者却似乎独具远见。 但我依然疑惑，“2/26/95”到底是指哪年的二月二十六日，1895，1995，2095 还是 2995？

**********

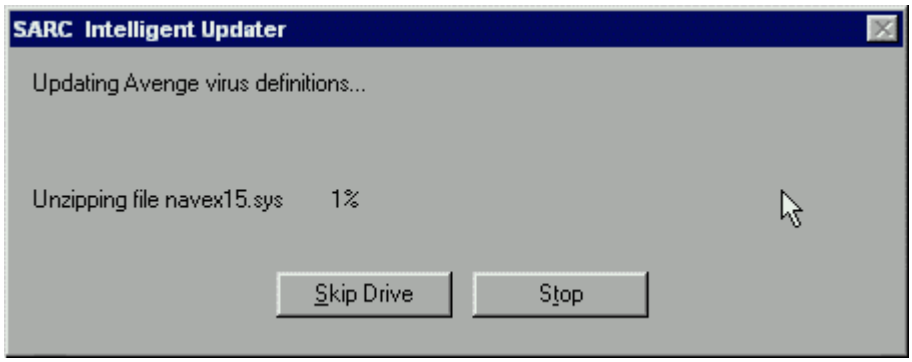
Adam Peterson 寄来的两张“Apriori”的截图，展现了软件界面设计中的“喋喋不休（In Your Face)”。在产生报表的过程中，应用程序提供“暂停处理”按钮，使用户可以暂停报表处理。


可当用户按下“暂停处理”按钮时，噢哦…（译者注：程序却告诉这个过程不能被终止）



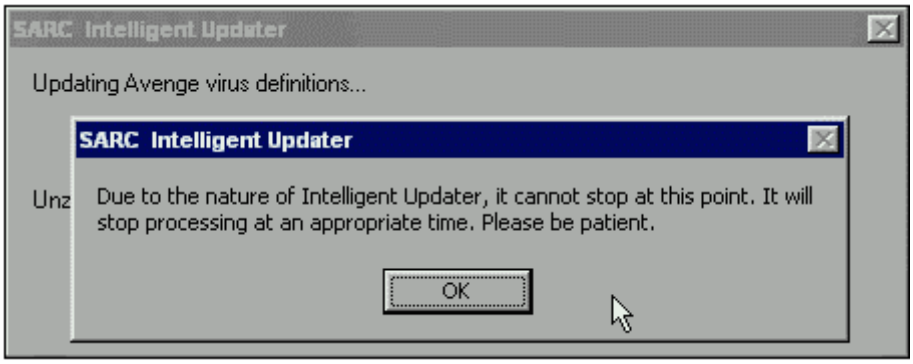

**********

Patrick McClimans 在诺顿用来更新病毒定义库的应用程序“SARC 智能更新器”中也发现了一个“喋喋不休”的例子：



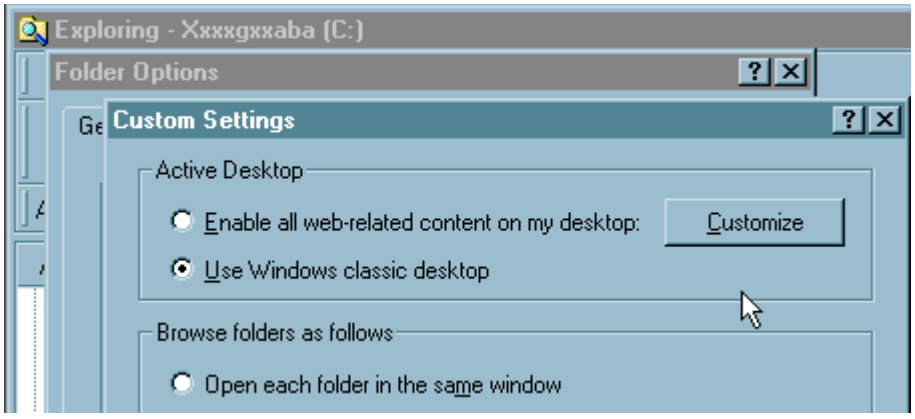
Patrick 说：

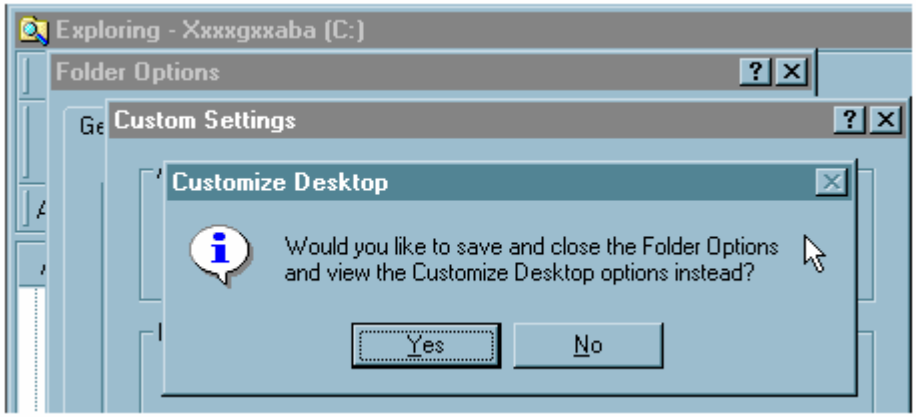
*我试图中止操作，它却对我作出傲慢的反应。真想给他来个 CTRL-ALT-DELI 并说“叫你停下你就停！”*




**********

另一个“喋喋不休”的例子来自 *Windows98* 的 *Explorer*：




（译注：这本来是一个 gif 动画，译者截取了两幅比较典型的图片。）

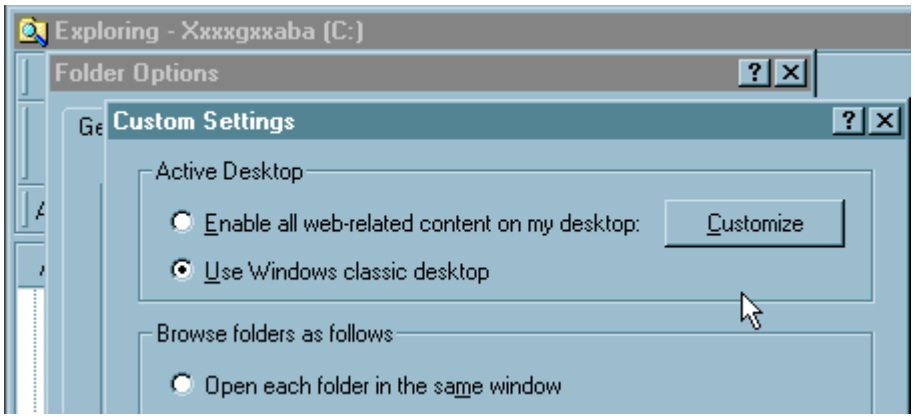
和 *Windows95 Explorer* 中的文件夹选项对话框（Option dialog）不同，Windows98 Explorer 中的文件夹选项对话框的确给出了选择项，也就是说，可以在这里对桌面进行配置了。可是，如果用户按了“配置”按钮，电脑又会反问用户，是否要转去配置桌面选项画面。这是怎么回事儿啊？!!!。

顺便说一句，无论用户是否改变文件夹选项对话框的设置，这个消息都会弹出。

**********

最后一个 "喋喋不休" 的实例来自 Jasc 的最新版的 *Paint Shop Pro (v. 6.0)*:

（译注：此图有误，原文如此。）




在默认设置下，用户将发现，他们不得不和用户界面里的标记玩捉迷藏。*PSP* 里的调色板工具有个神奇的功能，只有当鼠标指针从上面经过时，它才是可见的；当指针移出调色板，它的高度和宽度都会收缩。

这个设计的问题在于，为了关闭恼人的调色板，用户不得不和界面上的标记玩弱智游戏。当调色板在打开/关闭间切换，它的“关闭”按钮的位置也会随之移动几个英寸。如果用户按多了“关闭”按钮，它的位置又会改变。用户往往得再次移动鼠标打开调色板，而这只是为了能适当地关闭它。

**********

IHOS 研究了卸载程序，深入研究一下 Headlight 软件公司的 *GetRight*。每一个开发者都应该仔细想想他们的卸载程序是否符合功能的需求。

**********

**Dominic Start** 发来了这幅他使用 *Microsoft's SQL Server 7* 保存更新时碰到的错误消息截图：


经过广泛查找微软的技术文档，Domenic 最后终于知道了这条消息的真正含义："连接已经丢失，但是 SQL Server 已经重新连接。

更为巧合的是，**Adam Wilkinson** 在使用微软的 Visual Basic 时也遇到了这个对话框，他也发来了对此的看法：

当我们试图向一个只能向前的（forward only）或只读的记录集写操作时，也能碰到这个消息。这个 firehose 模式在这种情况下其实意思是说您只能按他们发送的顺序读取数据，而不能回滚或写入记录集。这种说法（firehose 模式）在这种情况下其实毫无意义，但仍然令人发笑并且让人好奇他们到底想让用户知道什么… …

**********

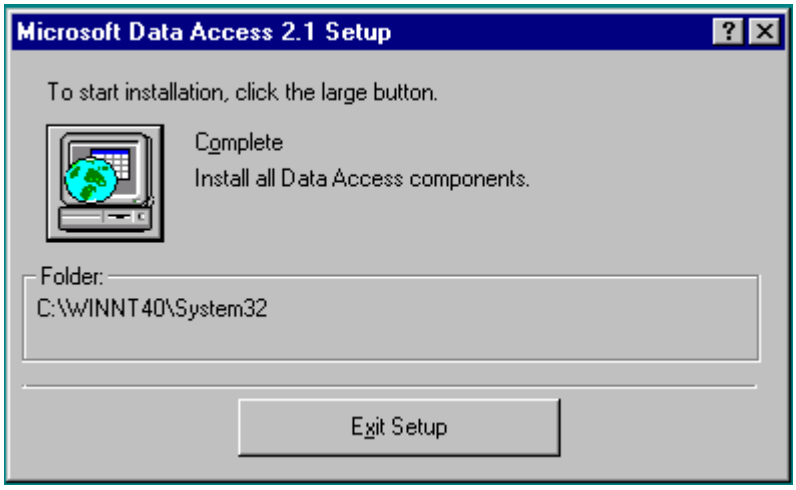
从一开始，我就认为微软将程序的安装和卸载并入一个可执行程序的做法是个失策。这种做法的愚蠢在 Windows NT 可选组件包的卸载过程中可以看得到：


**Adrian Cole** 指出，这个对话框不止一次的提到安装这个产品而几乎没有一处提到程序正在被卸载。

还有一点，想起通过点击"开始"来退出 Windows 系统，在运行"安装"中卸载程序也就不足为奇了。

**********

提起安装程序，James Lynn 说出了他对安装向导提供一个图标式按钮执行安装功能而用文字按钮执行其它功能的意见。下面就是从微软 DataAccess 2.1 安装程序的截图：

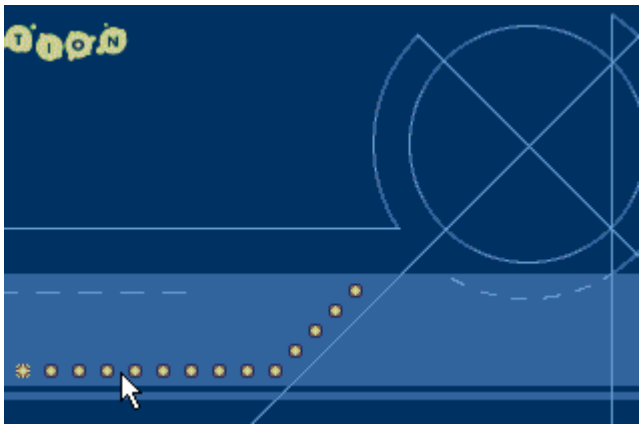


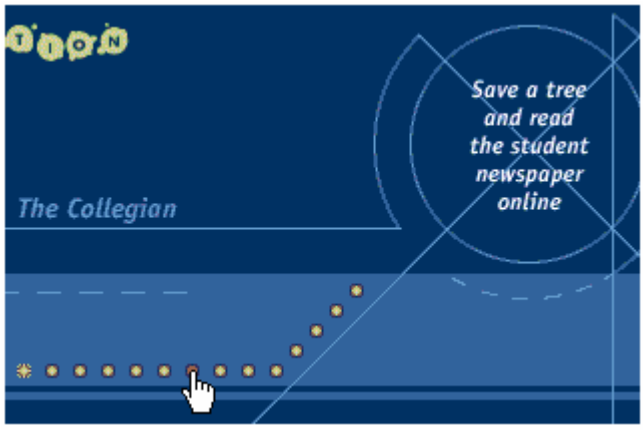

事实表明这个对话框缺少设计。不是一个而是好几个对话框里的提示都表明这一点。人们不禁猜测这些提示是因为用户（包括我）不能辨别那个图片是“开始安装”的标志。不幸的是，提示也是缺乏考虑：“退出安装”按钮出奇的比那个“安装”的按钮大。

很不幸的是，现在微软在开发工具中提供的安装包还是使用“哪个按钮大？”的设计。也许微软认为既然这个设计已经造成了很多用户的困惑，那用户习惯了就好了。我的意见是一个标着“开始”的按钮就能解决问题，甚至可以没有提示。

**********

Michael Keller 向我们提供了一个非常糟糕的设计方案，出现在塔而萨大学网站“学生栏目”上：





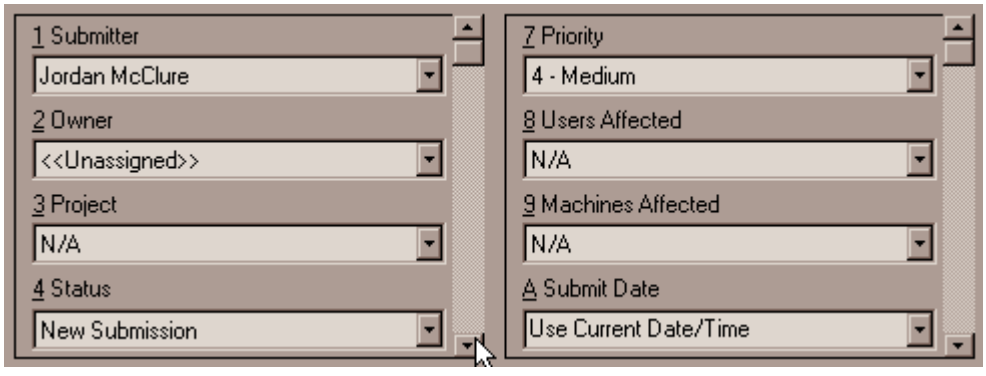

（译注：这本来是一个 gif 动画，译者截取了两幅比较典型的图片。）

这个站点上的各个不同部分的链接分别都由大小为 7x7 个象素的小点来代替。小点儿的意义只能通过将鼠标的指针移动到小点儿上来读到。没有标签表示链接和这些小图片。这些小图片毫无差异，聚在一起。毫无疑问，用户要找到和浏览这些链接真是及其困难。

设计者意识到这样的设计有问题。为了弥补最大的问题也就是访问者不会找到小点儿表示链接，所以他们在页面上加了一个动画提示“鼠标…移到…小点儿上”。在这页对应的纯文本页面上，他们称这个“小点儿”版本为“够劲”（Funcky）版。但我想到的能描述这个设计的单词是：可怜巴巴。

**********

Jordan McClure 发来了一些在配置管理应用软件 *PVCS Tracker* 误用滚动条的例子图片：



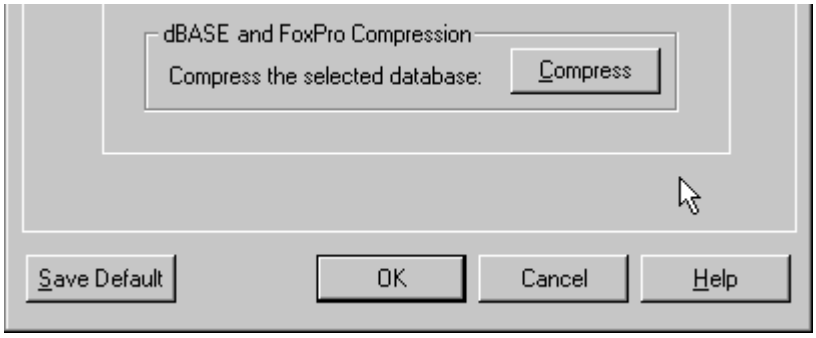

这幅图片大概需要解释一下。这是一个用来创建报表的对话框的一部分。报表的参数通过一些编号的下拉列表控件来设置。这些编号从 1 直到 B。不幸的是，设计者将这些控件分成两个独立的列表，并且两组控件的滚动控制相互独立。而从概念而言它们从属于一个列表。

还有另一个不可接受的事实：有些是数字编号，而另一些是伪数字编号。事实上这些编号是用来实现通过键盘访问每个控件而设置的，但设计者为了自己方便而直接给控件顺序编号了。设计者的懒惰理所当然的给用户造成了不必的后果：用户以为这样的顺序编号表明这些控件是有顺序联系的，他们**必须**按照这种次序来浏览这些项目，并且他们认为每项都不能省略。这样顺序编号的另一个严重的后遗症是由于后续版本会添加，删除选项以及选项的位置改变都会导致其它选项的设置必须相应改变。举例来说，用户发现他们在 1.0 版本里使用 ALT+4 访问这个软件的状态域而到 2.0 版本里就变成了访问优先级域了。

顺序编号是逃避问题的表现，是设计者为了自己简单而做的。

**********

Todd Hensley 描述了另一个 IBM 特色的界面解决方案，这次是用 *Lotus Approach* 举例的。



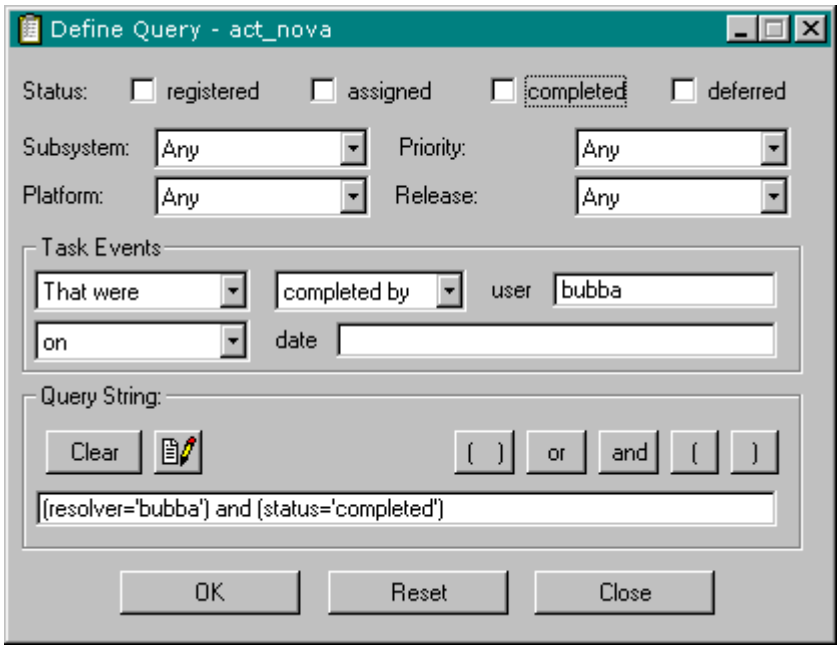

使用 *Lotus Approach* 来压缩数据库，用户必须选择“数据库”属性页，而这个属性页需要通过选择“文件|用户设置|参数选择”来弹出。我们暂且不说很少用户会觉得这样一个基本的数据库操作和这样漫长的操作是合适的。就说当用户最终历经艰辛找到了这个功能之后，这个界面还会让用户觉得它不能工作。

这个压缩操作用一个标准按钮上加二字“压缩”来搞定。但是，当用户按下这个按钮之后，它就变成无效的了。Todd 说他按下这个按钮之后，等了一会儿，他认为这个数据库应该正在被压缩。可过了一会儿，Todd 意识到可磁盘一点儿响动都没有。下意识的，他按下了对话框低端的“OK”，这样一来，他发现了这个对话框的功能。

在这个例子中，Lotus 改变了标准按钮的行为而让它变成了一个很像复选框一样的开关。这就足够糟糕了，它不像复选框那样，它没有提供关闭压缩功能的按钮。一旦被按下，唯一的不像压缩数据库的选择是点击“取消”来关闭对话框。

**********

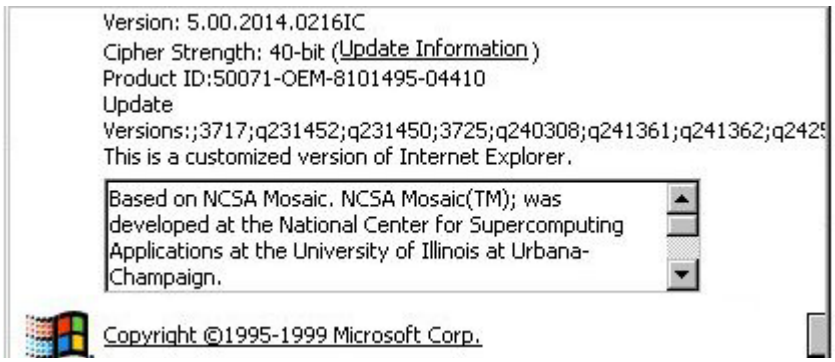
Hugo Russell 提供了这幅从一个叫 *Continuus* 的源代码控制软件上截来的图片。图片中对话框的用来对源码数据库生成一个查询语句的功能设计得非常糟糕。




最过分的是对话框中对复选框的错误用法。复选框的标准行为被改变从而变成像按钮一样使用。用户点击一下这个复选框后，一个查询短语被加到对话框低端的查询语句中。但是，这个程序马上又把复选框的选择状态改成没有被选择的状态。这样造成用户一个没有发生操作的错觉。如果用户又想将这个复选框变成“已选择”的状态而重新点击它，这个查询语句又添加了一段查询短语。

**********

Amit Schreiber 发来了这幅微软 *Internet Explorer 5.0* 中安装了补丁之后截来的“关于”对话框图片。

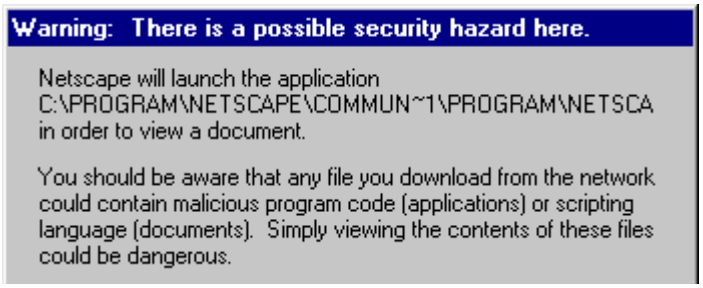


微软似乎想将每一个打过的补丁的历史纪录版本号用分号分开显示在关于对话框里。也许是因为这是第五个开发版本，也许开发者没想到竟然有这么多的补丁要打所以没有预留足够的空间来显示补丁的信息。天哪！

那个对话框右下角灰色的东西是什么？噢，那是“OK”按钮。或许是什么原因吧，开发者将补丁的信息宽度和按钮的位置联系了起来，所以就变成那样了。如果 Amit 再多打一个补丁的话，他将和“OK”做最后的吻别了。

**********

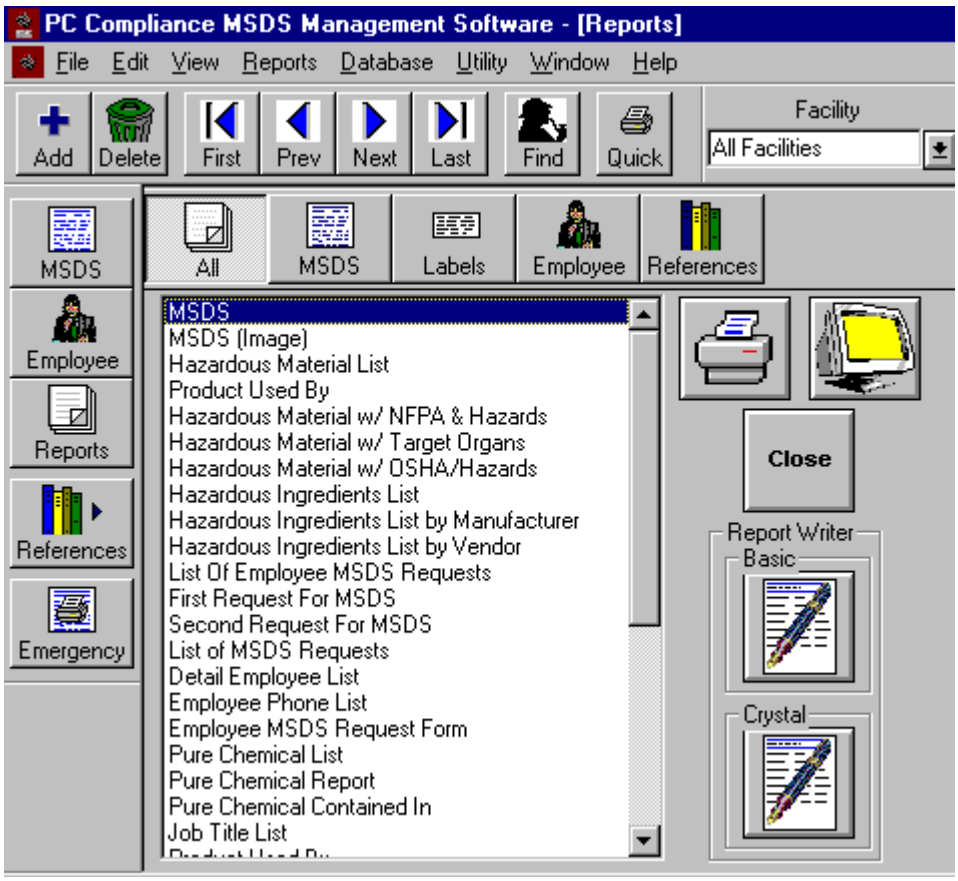
当然，不光是微软犯不能正确显示长文件名的错误。**Per Lindström** 发来了这幅从 *Netscape Communicator* 截下的不能显示重要信息的图片：




如果觉得告诉用户程序名很重要的话，那么就应该确保给程序名的显示留有足够的空间。这个错误就是因为程序存放在安装 Netscape 时创建的缺省的目录里，而目录又特别长造成的。

**********

一个匿名游客发来了一幅屏幕截图，部分如下所示。这来自于 RMS 系统的 *PC Compliance MSDS Management Software* 软件。这个程序的目的在于允许用户轻松管理材料安全数据页。这些数据描述了在工作间里使用的化学物质对身体和健康所危害的程度。同时，也描述了发生危急时应该采取的措施。




我们认为一个用来提供应急方案的软件应该尽可能高效地使用户获得相应信息。但这个软件按钮上的图片却只能迷惑用户而全无他用。整个界面充斥着重复的工具条，同一幅图片被用于不同的功能表示而不同的图片又代表同一个功能，而且，分组栏里只放一个按钮（？!），简直是一塌糊涂。

*讲授用户界面设计的专家学者请注意：*PC Compliance MSDS Management Software 为我们提供了一个极好的用于讲授用户界面设计问题的范例。该软件演示版可以从其网站 http://www.rmssystems.com/ 上下载到。不过重复使用这个软件的 demo 则会发生错误。

**********

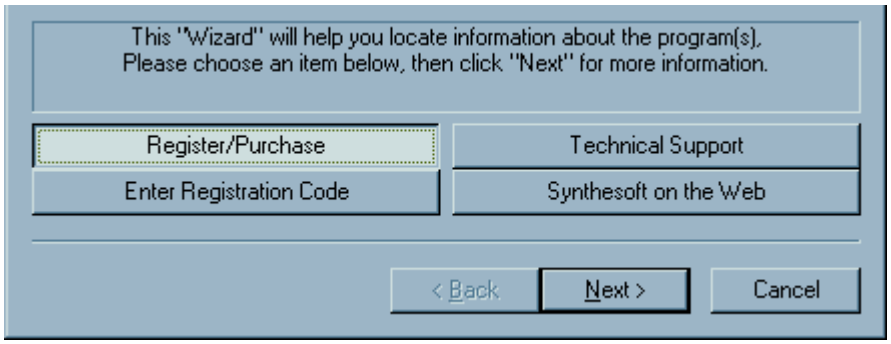
有些日子，我想给网站添加一个新的部分来说明对工具条的设计。与其同时，一个匿名游客发来了一个被描述为“Lotus Notes，被黑客修改得脱胎换骨后改名叫 StartNotes”软件的截图。看看这吧：


这个“按钮功能要点”窗口是通过点击工具条上的“帮助”按钮弹出的（即窗口中所指的第一个按钮）。这个“要点”的荒谬之处在于用户必须一个一个的数按钮所在的相对位置才能知道按钮功能的序号。

对于 StarNotes 开发者，我奉劝一句：没有什么能比图片本身解释的意义更为清楚。

**********

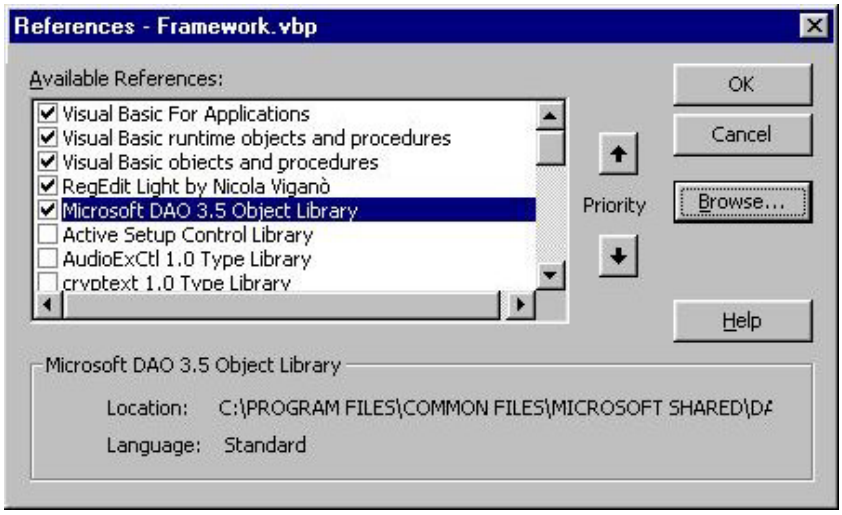
Steve Brickman 发来了这幅从 *Psychedelic Screen Saver* 软件上截来的注册页面截图：




在这个例子中，按钮被用来作为指定选项使用。用户通过选择一个按钮来指定一个选择，然后通过点击“下一步”来启动某个向导。这些向导的执行过程与被选择的选项有关，所以有各种样式的按钮展现在界面上。

嗯，这儿有个特点：按钮被用来…做选项控制。如果您想让用户做出一个选择，请使用选择控件：如单选框，复选框，下拉列表等等。如果您忘记了这条基本原则，您的按钮的使用方式将会与其它软件的方式相异。就和软件 *Psychedelic Screen Saver* 的一样了。

游客发来了微软 *Visual Basic* 的引用选择对话框截图：

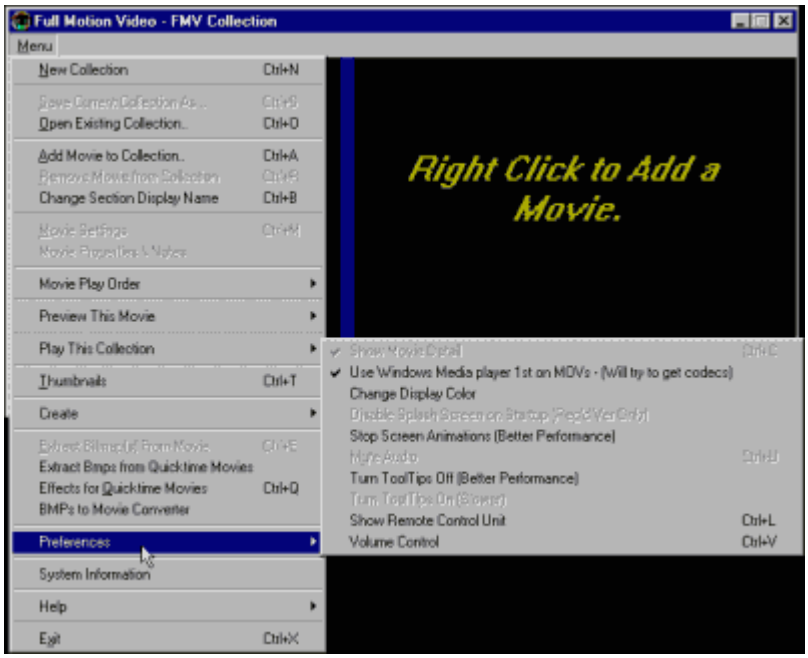



当用户点击任何一个引用时，它所关联的文件名将会显示在窗口的低端。不幸的是，微软没有足够的空间来显示这个文件路径字符串。也许有人会说开发者没想到有这么长的路径等着显示。不过这可是微软自己开发的程序啊。

如果您认为有很重要的东西要显示在屏幕上，请给它留有足够的空间。

**********

游客 **Avery Lee** 建议将程序 *Full Motion Video* 作为糟糕界面的候选者，主要是因为这个程序只有一个菜单。



或许有人认为设计者这样做是因为他使用的开发环境对菜单编辑功能支持得不够好。我们称这种菜单叫“Windows95 开始菜单式”：一个菜单（毫不奇怪它叫“菜单”），里边有许多层层叠叠的子菜单组成。不过，如果将这大菜单里的选项拆分成小菜单，这将方便用户快速访问比如像一般开发者不太在意的“帮助”功能。

**********

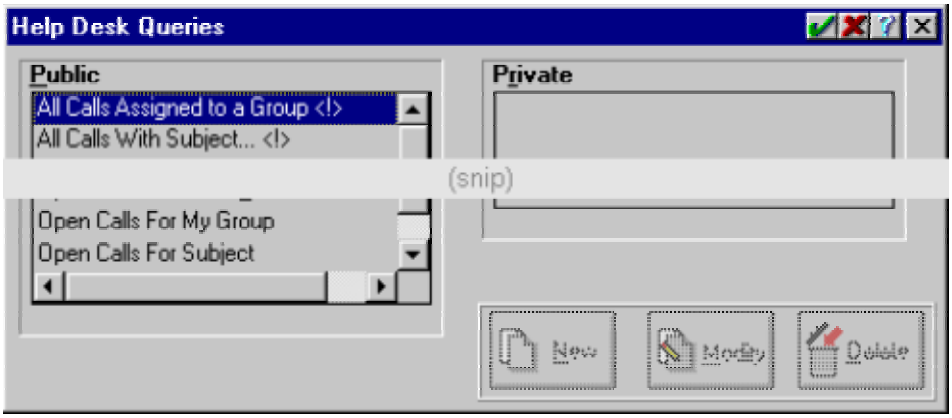
“Full Motion”并不仅是一个软件名字而已，它还是开发者哲学倾向的表现。从用户一开始运行程序到电影播放为止，一个动画一直在不停播放：


（译者注：这本来是一个 gif 动画，显示了不同的形式的 Right Click to Add a Movie）

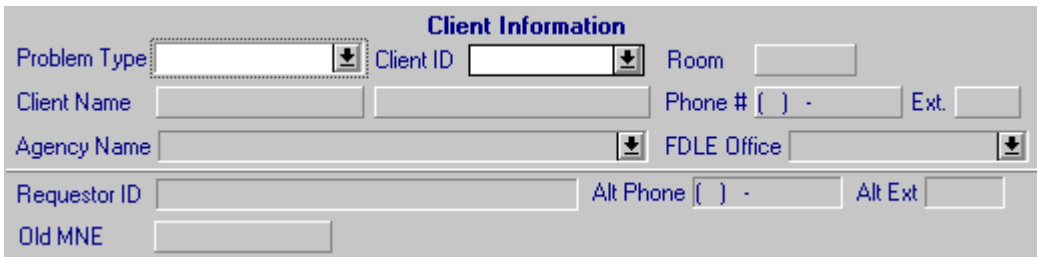
不幸的是，右击菜单并没有改变。用户还碰到的是那个超级大菜单。Lee，谢谢您。

**********

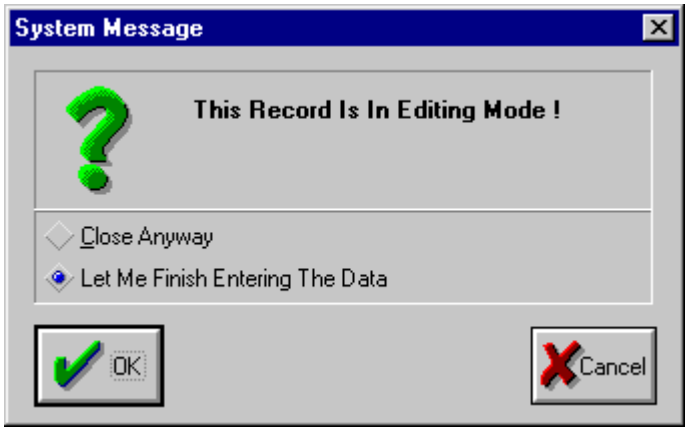
在近几年里，有不少访问者建议将 *SupportMagic* 添加之糟糕界面中。Wil Johnson 是一个使用截图而且说明为什么这样做的原因。因此，有几张截图被理所当然的添加至糟糕界面库之中。




尽管有数百张设计失败的例子在糟糕界面集锦中描述，我还是被软件 *SupportMagic* 中使用标题栏作为应放置在对话框的按钮的容器的错误方法所震惊。或许是微软开创了将窗口管理控件以小按钮形式放置在窗口标题栏的先例。这种错误做法从 Windows95 发布以来就开始了：我们想最大化窗口时却最小化了所有窗口程序，想最大化一个程序时却恢复了窗口原有大小，而想最大化一个窗口却错误的点了关闭的按钮。一开始，微软仅局限于将窗口管理控件放置入标题栏中；后来，他们又失策的将帮助按钮放到了有些对话框中。*SupportMagic* 或许就是因为被这不良方法所迷惑而随意在标题栏中添加控件。迄今为止，我还是没有搞定他们在标题栏中放置按钮的标准，对是否将“OK”和“取消”放入标题栏的规则也还没有概念。将按钮放入标题栏真的是一个错误的做法，它绝对会让您的程序碰到使用不便的问题。




这是软件 *SupportMagic* 的主输入列表的对话框的一部分。我们建议它应该更名叫 *SupportMystery*。任何第一次碰到这个软件的人都不知道它为什么被设计成这样。一些域被凹进而另一些却又凸出。并且被凸出的域看起来更像一个按钮。某些时候，类型相似的信息被不同形状的边框所包围：电话号码域被凸出而备用电话号码域又被凹进。除了下拉列表的背景是白色的之外（这是程序员不能更改的），其它控件的背景都和对话框颜色一样。而这业界通常都认为这些域是只读的。不过，在 *SupportMagic* 却不一样。这仅能说明这些设计者/开发者都对业界传统不太熟悉。



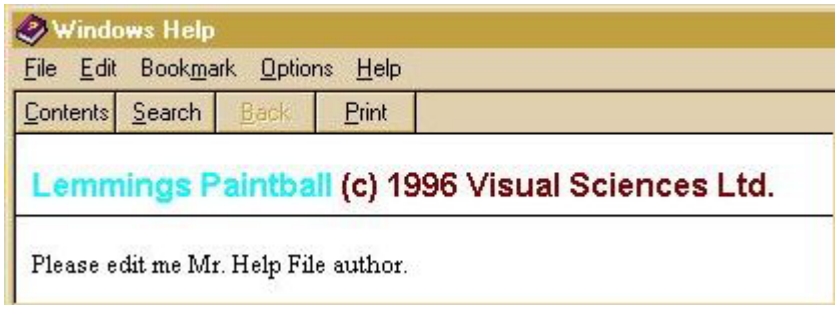
当用户试图关闭这个问题窗口而又不想保存数据时，*SupportMagic* 不是提供一个标准的确认对话框（“您确认要…？”），而是对用户**大叫**这是在“编辑模式”（看起来似乎很重要）。一个程序绝对不能有这样的口气，如果这样，所有程序和用户之间要建立的气氛将会丧失殆尽。

还有，这个对话的目的是确认用户是否真正想关闭原来那个对话框，但这个设计却完全背离了原意。这个对话框中的“取消”按钮——最显目的按钮——完全无用，因为用户必须首先选择这两个选项（非常独特的设计）而达到目的：为了忽略前面的编辑结果，您首先必须选择“强制退出”，然后选择“OK”按钮退出。而为了保存原有的结果，您有必须先选择“让我输完结果”（让我？）然后选择“OK”按钮。


这的确是让我们哭笑不得。这个图片是从 *SupportMagic* 软件中众多工具条中截下来的一个：在工具条上“水果”图标表示程序的“参数选择”命令（其实有必要放到工具条上吗？），而且，更为不解的是，“餐具”图标竟然用来表示检查在移交工作之前这个人是否空闲。

*SupportMagic* 谣传是一个位居前列的桌面帮助系统。不过，如果真是谣传的话，这就可以和它目前用户很难得到技术支持的事实相一致了。

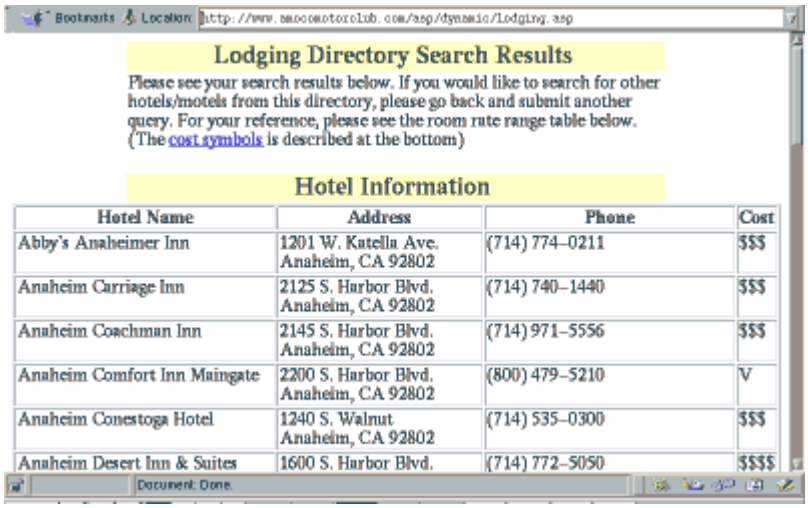
**********




Lisa Carter 给我们发来这幅从游戏 *Lemmings Paintball* 截来的图片。这幅图片充分的说明了质量评价测试必须包括应用程序和其支持文档两方面。（译者注：图片上这个帮助的主题是一句无用尚没编辑的垃圾信息）

**********

Carl Fink 发来了这幅图片，这是他通过美国石油公司汽车俱乐部网站（Amoco Motor Club）计划他的旅游行程是碰到的。这个网站允许查询旅游线路上的旅馆。不过，你只能将这查询结果按州显示。他想去加里福利亚去玩，不过他却得到了含有一个长达 630 个会员旅馆的列表。Carl 不得不一页一页的去找在目的城市的旅馆。如果网站设计者试一下这项功能，他就有可能考虑将查询结果重新做一个更小的粒度划分。



Lodging Directory Search Results

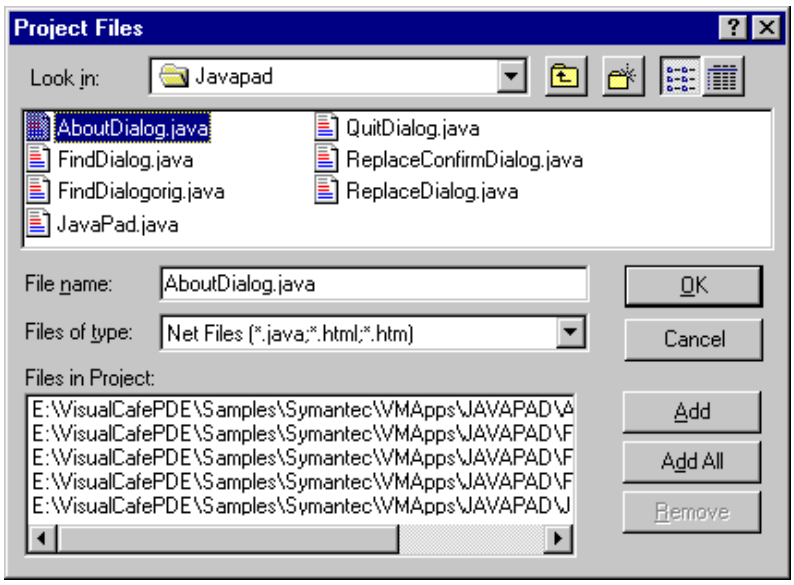
Please see your search results below. If you would like to search for other hotels/motels from this directory, please go back and submit another query. For your reference, please see the room rate range table below. (The cost symbols is described at the bottom)

Hotel Information

| Hotel Name | Address | Phone | Cost |
|---|---|---|---|
| Abby's Anaheimer Inn | 1201 W. Katella Ave. Anaheim, CA 92802 | (714) 774-0211 | $$$ |
| Anaheim Carriage Inn | 2125 S. Harbor Blvd. Anaheim, CA 92802 | (714) 740-1440 | $$$ |
| Anaheim Coachman Inn | 2145 S. Harbor Blvd. Anaheim, CA 92802 | (714) 971-5556 | $$$ |
| Anaheim Comfort Inn Maingate | 2200 S. Harbor Blvd. Anaheim, CA 92802 | (800) 479-5210 | V |
| Anaheim Conestoga Hotel | 1240 S. Walnut Anaheim, CA 92802 | (714) 535-0300 | $$$ |
| Anaheim Desert Inn & Suites | 1600 S. Harbor Blvd. | (714) 772-5050 | $$$$ |

**********

Bob Florian 发来了这幅从 Symantec 的 *Visual Café* 软件中截来的图片。他认为这是他碰到的界面最差的开发工具。这个对话框是用来添加或删除工程中文件的。上面的部分很像 Windows95 中的通用对话框，下面的部分是工程含有的文件列表。尽管界面很像 Windows95 的标准控件，但实际上却给用户带来了问题：




*当您看到这个对话框时，您总是想从文件选择控件中选择一个或多个要添加到工程的文件然后按 OK 这些文件*

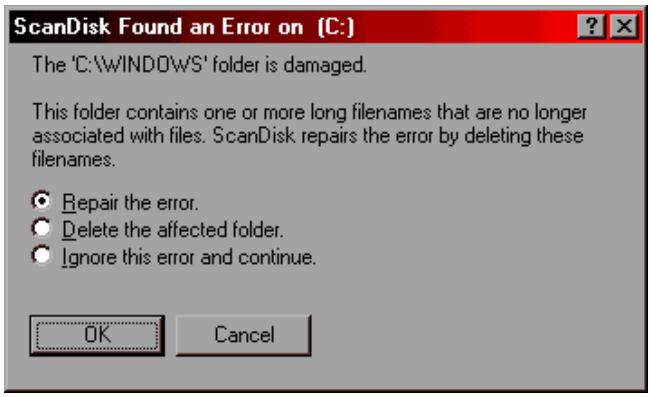
*就加到了工程里。但在这儿就不一样了这儿必须要您选择文件之后，先按“添加”按钮，然后再按“OK”才能搞定。*

*啊呀!我发现这种倾向是如此之强烈以致于我使用了这个软件六个月之后还是忘记先按“添加”按钮再按“OK”。所以，我老是被添加了文件却找不到文件搞得晕头转向*

这个对话框还有其他的问题。用户必须将“工程中文件”控件的滚动条移到最右才能看到工程中的文件，并且这个对话框不能记住上次打开文件的目录。还有，尽管这个对话框里含有“添加”和“删除”两项功能，但是通过菜单“插入->文件到工程…”也可显示这个对话框。或许 Symantec 被 Windows95 的一大特征所影响：点击“开始”按钮来关闭 windows 系统。

**********

John 在运行 *ScanDisk for Windows* 时碰见了这个对话框，如下图所示。想到这个文件夹的特殊意义，我们不禁想试试如果我们选择了“删除被影响的文件夹”会发生什么情况。（译者注：这个文件夹可是 Windows 目录）




删除“Windows”文件夹的确看起来的确像对这个丢失文件关联的错误的改正，不过，一看到就想到可能是微软干的，对吧？

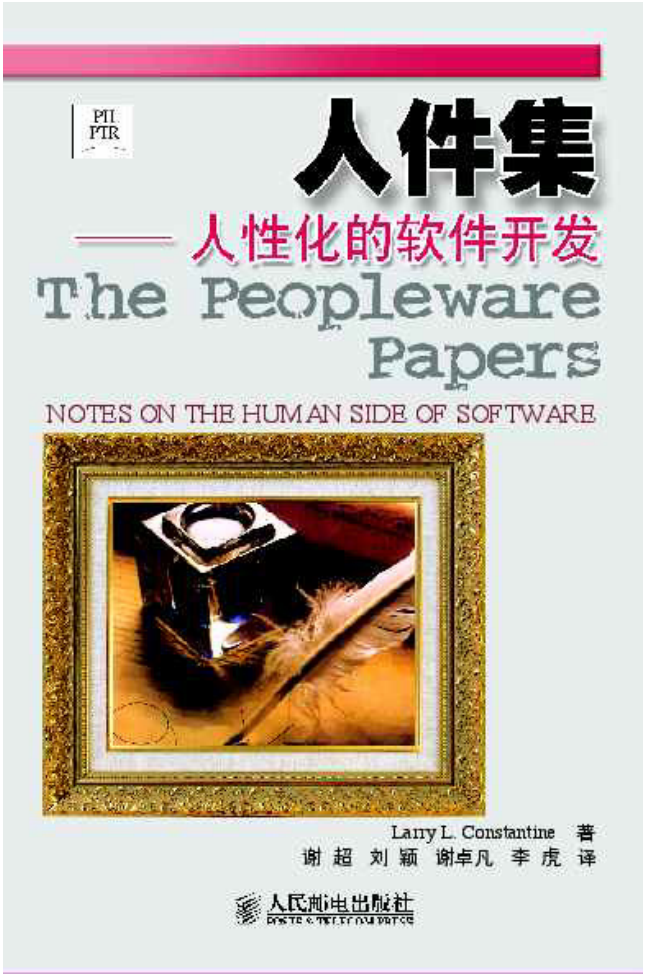
**********

Rob Domaschuk 发来了这幅从 Lotus 公司的 *ScreenCam* 软件的安装程序截下来的图片。看来这个在 Lotus 做事的人还不知道他是去是留。（译者注：那个无效的按钮（Change Folder）应该是有效的才对）。

# 《人件》续篇——《人件集》样章

Larry L. Constantine 著，谢超 等 译

查看评论




## 硬件、软件和人件（第 1 版前言）

单纯使用 CASE（计算机辅助人件工程）工具、可视化程序设计方法、快速设计原型或对象技术，并不能开发出一个好的软件。一个好的软件应该出自于“人”，而有趣的是，一个糟糕的软件也同样出自于“人”。1992 年，我开始定期为一个专栏写文章，专栏的主题不是关于硬件，也不是关于软件，而是关于人件。这是因为，当时我有一个简单的想法：既然软件是由人创造的，也是由人来使用的，那么只有更好地了解人是如何工作、如何解决工作中的问题、如何协调工作中的关系，才有可能设计、开发出更好的软件。

今天，我们每天都会遇到大量新的词，有大多数是旧有词汇的新解，而像“人件”这个词就是罕有的必须重新创造的词。Peter G. Newman 因为他的一份关于“人类的风险与真正的计算机和计算机程序危害”的报告而出名。他应该是第一个正式使用“人件”这个词的人。1976 年，他写了一本不太为人所知的书：*Peopleware in Systems*(《系统中的人件》)，在书中的一篇同名文章中，他首次用到了这个词。Meilir Page-Jones 在 1980 年所写的 *Practial Guide to Stmctured Systems Design*(《结构系统设计实践指南》)一书中，再次使用了“人件”一词（正是此书让一般程序设计人员能很好地理解我的作品中关于“结构设计”的内容）。但是**直到 1987 年 Tom DeMarco 和 Tim Lister 合著的《人件》(《非程序员》注：已由清华大学出版社出版）一书的出版**，才使人件才正式成为程序设计领域中的一个专业词汇。

人件，是第三次计算机革命的真正起源地。第一次革命源自“硬件危机”：在一段时期内，人们一直认为自己遇到的计算机问题都源自硬件方面。当时，人们以为，只要有了运行更快、功强更强大的计算机，有更多的内存和更棒的外部设备，就能建立更好的系统，也就能解决所有的问题。渐渐地，人们有了更好的计算机。年复一年，计算机运行速度越来越快，内存越来越大，外部设备也越来越好用而且便宜，可是计算机问题依然存在。我们仍然在使用运转不稳定的系统，而且，无法及时、有效地在预算范围内完成任务。于是，我们将遇到的问题归咎于软件方面，而第二次计算机革命也随之被称为“软件危机”。人们开始认为，只要有了优秀的编程工具、高级的编程语言、丰富的构件库和辅助程序建立系统，就能解决所有问题，及时、有效地在预算范围里开发出运转良好的软件系统。现在，第三代编程语言变得越来越来精密，并出现了第四代编程语言；编译器变得越来越快、越来越聪明；计算机辅助软件工程工具随处可见。结构化革命让我们认识到结构设计和分析，面向对象技术也开始变得成熟、流行。但是我们还是不得不经常改动我们的工作计划，追加预算，计算机问题依然存在。

最后，我们不得不重新认真考虑一下，问题到底出自什么地方？“我们的敌人其实就是我们自己！”是的，人件，就是问题的症结所在。“人”是问题产生的原因，也是解决问题的工具！

人件包含的范围包罗万象。在软件和应用开发过程中，凡是与人有关的任何事物都可以归于人件。我所写的书和专栏中都谈到了人件中所涉及的各式各样的内容：质量和生产率、合作、团队动力、个性和程序设计、方案管理和组织问题、界面设计和人机交流、认知、心理学、思维过程等等。

以上所有话题都是我感兴趣的，也能让我感到兴奋。我当初攻读管理学的部分原因就在于，这门课能让我将计算机、系统原理同心理学联系起来。我的毕业论文就是关于计算机程序设计心理学的。多年来，我已经将心理学家 George Miller 先生和他神奇的数字（当然是 7±2 了）介绍给了成千上万的学生们和数十位同事了。为了更好地进行软件、应用程序的开发，人们精心设计出结构图表的结构帮助开发人员形成可视化概念，并用于解决相关的问题。接合和连接描述的是人们所看到的计算机程序的效果，它们是结构设计核心中重要的度量尺度。程序设计人员在设计、维护、修改程序时思维过程是复杂还是简单，直接决定到他们设计出的程序是复杂还是简单。

从某种程度上来说，我的工作既不能脱离人，也不能脱离计算机。1976 年 7 月，当美国庆祝独立 200 周年时，我曾宣布自己告别计算机界，当时，我以为我自此就可以脱开身了。10 年来，我一直像一个训练有素的家庭临床医生一样，每天，或以个人身份或以代理身份出现在社会上，为大大小小、老老少少、男男女女们解决各种各样的问题。但是来自业界的压力又将我重新推回了技术前沿。

人件，就是上述提到的技术前沿的十字路口，诸如管理、组织发展、个性、模型、工具、方法、过程、人机交流等方面的问题最终都会体现在人件上。在我写文章时、工作时或教学时，都会不时地提及所有这些方面。为专栏写文章，让我有我机会在人件这个广阔的天地中畅游，不时停下来思考一些有趣的想法，直面随时遇到的挑战，在软件和应用开发的大道或乡村小路中信步。

本书记录了我在人件世界中的旅程，从《计算机语言》杂志开始，直到《软件开发》。我做的专栏题目也叫做“人件”。本书中，包含了人件专栏中的所有文章和发表在其他地方的一些相关内容的文章。所有这些短文和文章都已做了编辑，以确保其连续性；其中一些素材，当初为了适应杂志文章长度要求，做了相应删减，此次出书时经过重新整理又加了上去。当然，这样或那样的改动，都是为了让书中的章节看上去更连贯、更流畅。但是，请记住，本书不是一本百科大全，也不是什么教科书，更谈不上是一份人件世界的路线图，人件世界的疆域实在是太广阔的了，本书，充其量只不过是一个旅行者的游记罢了。

我，还将会继续我在人件世界的旅行。

# 第 5 章　办公空间

当你在那里绞尽脑汁地思考一个难题，并试着解决它时，你的同事在旁边一边嚼着口香糖，一边打游戏，时不时地发出一些怪声，还动不动地问你一些愚蠢的问题。这时，你首先考虑的就是：我到公司这么年了，该给我配一个私人办公室了。你找到老板，向他诉说你需要更多的私人空间，以避免被他人打扰，这样你才能更好地发挥自己的才能。你甚至还说出了房间的格局，至少应该有 100 平方英尺的活动空间和 30 平方英尺的办公桌椅，房间最好还有窗户。在丹麦，已经制定出了相关法律，只要雇员正式向自己的老板提出了带窗户的请求，那老板就必须为雇员安排这样的办公室。

在软件开发界，大家都认为：更大的办公空间、更安静的工作场所、更少的骚扰就意味着更好的软件。于是，100 平方英尺的空间和 30 平方英尺的工作平台就成了每个程序员所向往的办公环境。在经典书籍 Peopleware（《人件》，1987）中，Tom DeMarco 和 Tim Lister 首次提出了这个观点。他们引用了多种资料，但资料来源主要还是以他们自己创办的年刊 Coding War Games（《编码战争游戏》）为主，他们总结出：更安静、更宽松的工作空间会带来更好、更高的编程效率。

## *空间的形状*

办公室、家里、教室，不同的物理空间形状适合做不同的事情。你和一堆人坐在长长的宴会桌旁，你能够和你两旁的人方便地交谈，但是如果你从这头传话到另一头，要保证原话不走样就很困难了。亚瑟王（King Arthur）首创了圆桌会议，现在这个词已经成为平等的代名词，亚瑟王当初的本意也就是为了各位骑士能够方便地自由交谈。

建筑或办公室的设计和装修情况极大地影响着人与人之间的交流和协作。我曾经在一栋公寓楼中住过两年，但是很少能碰到我的邻居们。那栋公寓楼的入口非常小，夸张点说，如果你背着一个大包裹你就进不了门，公寓楼的大厅又窄、又黑、又小，邻里间难得碰面，也没有地方进行交谈。

办公楼一般都有着宽大的入口和走廊，但是，现代化的高科技办公室在结构设计上也有着非人道的一面。“柔性分割”、“开放办公”的设计思想主导着现代办公室的布局，尽管听上去很好，但实际上，这种布局既不柔性也不开放。

一个典型的计算机公司布局就是，由长长的走廊把一大堆小房间十字交叉地连接起来，房间内部分成多个隔断，把人员分离开。如果你想看看别人在做什么，你必须伸长脖子从隔断上方看过去；如果你想和别人交谈，谈话效果也非常差。这种布局不但没有满足个人隐私要求，也没有满足开放性的要求。如果在一个办公室里，两个人相隔 20 英尺以上，那么他们之间可能几年也不会交流一次。来公司参观的人必须有人陪同，但这并不是出于安全性的考虑，而是怕他们找不到地方。邮件上必须标明类似 KK14-HDQ:117N\BB.R3 字样的内容，这行看似神秘的字符实际上指的就是诸如楼层、单元、第几排等信息。柔性办公室的柔性是根本靠不住的。如果要换个地方，就需要把所有的东西都搬走，甚至包括更换邮件标识，因为隔断可以拆除，人可以挪地方，但邮局的人可不会自动识别这些。

合理的办公空间可以促进软件开发效率，如果从这个观点来看，这种所谓的“柔性设计”空间是最差劲的。既不能满足单独工作的要求，也不能满足协同工作的需要；既不方便大家迅速撤离，也不方便大家闲时小聚。

但是，那种认为为开发者提供更安静、更宽松的办公空间就可以提高软件开发效率的观点也是不正确的，办公室布局与开发效率之间的关系远远要比这句话复杂的多。首先，从社会科学研究的角度来看，“原因－结果”之间的关系是很复杂的，很难说清哪个是因，哪个是果。高效的程序员有可能是在安静、宽松的办公环境下办公，但也可以说，正是由于他们的高效开发使得办公环境显得安静、宽松。

更重要的是，这个观点是 Tom DeMarco 和 Tim Lister 基于他们的软件编程测评结果所提出的，他们的软件测评只关注了编码环节，而忽略了完整的软件开发流程中的其他环节。另外，在他们的软件测评中，所有参与测评的程序员都被要求独立工作，而不是协同开发。因此，我们只能说安静和宽松的办公环境对提高独立编程效率有作用，那么，对于团队协作开发是否同样起作用呢？

## *协作交流*

为了有效地协同工作，项目组确实需要足够的工作空间，他们需要一个能容纳整个小组的空间，一个让整个小组更像一个团队的空间。事实证明，一个开放与封闭相结合的空间有利于充分的交流。办公室内应该有适合两至三人短期或长期协同工作的地方，也至少应该有一个类似于会议室的地方，可以让全体项目组的成员一起讨论问题。如果没有一个合适的地方让全组的成员可以方便地交流，那么要想将全组人员团结到一起就是一件很困难的事情了。

有一个软件经理很热衷于团队建设，但是，他对团队的协作质量却十分失望。他们工作的楼层布局很有意思，有很多办公室，办公室之间由矮玻璃墙隔开。只有很少的几间办公室能容纳两人同时办公，而那些比较大的办公

室里却坐满了不同项目组的人。我观察过其中一个项目组的开会情况，他们在那个楼层中的最大的会议室里开会。11 个开发人员挤在那个会议室中，而会议室里只能摆放 1 张会议桌和 6 把椅子，在桌子与白板之间只留有狭窄的空间，项目负责人在白板上写字时连转身都很费劲。可想而知，会议很快结束了。

对一个需要协同工作的团队来说，在整个软件开发期间，拥有一个团队自己的类似于会议室的地方是及其必要的。团队需要一个可以当作“指挥部”的地方，整个团队可以在这里方便地开会、讨论问题，藉以躲避外面的喧嚣嘈杂。如果没有足够多的会议室，那么在迫不得已的情况下，与其他团队共用一间会议室也是一种较好的选择。

对使用“系统情节图”（system storyboarding，参见文献 Zahniser 1990,1993）和其他协同分析和设计方法的团队来说，拥有一间充足空间的“指挥部”就显得特别重要。房间内的墙壁和白板成为这个团队工作的战场，可以用于记录团队工作的内容和工作过程。记录的内容可能包括团队标志、团队状态、重要的设计文档等各方面的信息。房间及其内部装饰是团队文化的一部分，它为团队提供一种共享的气氛，帮助团队成员高效、和谐地协同工作。

如果一个团队在地理上就是分开的，比如分布在不同办公楼、不同区域、甚至分布在世界各地，那么协作交流就变得更加困难，开销也更昂贵。同理，一个项目分布在多个区域（即使只是在不同办公楼、不同的楼层），项目的开销也要上升 50%~100%。因为比起相对集中的团队空间分散的项目团队，需要额外的机制来进行交流与协作，因此前者的效率肯定高于后者。虽然现在有 E-mail 和远程电话会议可以帮助团队之间进行协作交流，但效果远不如在一起面对面地交流，后者至少可以增进感情。

当我在 M.I.T 时，正处于软件工程的黑暗年代后期，那时，软件界已经逐步意识到应该为研发团队提供一个便于交流的工作环境，而这种工作环境可以提高软件的研发效率。合理的环境应该是一种集中式的开放空间，这样可以促进和推动团队成员协同工作。虽然固定的工作空间和封闭式的管理会让团队成员感到不便，但是如果能够创造性地利用空间，一样能够起到积极的作用。Risa Hyman（参见文献 Hyman 1993）曾经描述过这样一个场景，一个团队最大限度地利用了传统的办公室布局，他们在长长的走廊墙上挂满了白板，并将走廊当作了他们的主会议室。

拥有布局良好的办公室当然会对团队工作有利，但是，对高效率工作真正起决定性作用的还是这个团队的管理者和团队成员。是把墙当作交流的障碍，还是当作沟通的桥梁，完全取决于人！

摘自《软件开发》，第 1 卷，#12，1993 年 12 月

# 第 56 章　意大利餐厅的柱子

托斯卡纳（Tuscany）的城市文化核心魅力就在于它的伟大艺术和美食：对美国人来说是 Florence，对意大利人就是 Firenze。在佛罗伦萨的建筑中，柱子是随处可见的。这些柱子，让我想起了一个餐厅，那简直就是一个最佳实践的代表模型，而且我相信，我们所有人都能从中有所收获的。

毫无疑问，Ristorante Alle Murate 是托斯卡纳最好的餐厅。我之所以这么说，是佛罗伦萨的餐饮历史告诉我的。而事实的真相是，当我在那家餐厅享受了一顿美食，渡过了一个难忘的夜晚后，我知道我不可能找到比这更好的餐厅了。那的菜味刚刚好，套用一句古话就是：多一分则肥，少一分则瘦。甚至连菜上的点缀品都恰到好处，极富艺术表现力。一切都很完美，无可挑剔。

这对我来说是一种很珍贵的经验。就像大多数软件界的人员一样，我没见过毫无瑕疵的程序。作为一个到处巡回的咨询顾问，我曾经在许多餐厅吃过晚饭，但是我的直觉告诉我那些餐厅都有待改进，而且我也一直考虑：如果这是我的餐厅，我该怎么改进它呢。但在这个餐厅里，我的想法变了。它太完美了，如果不改变它的性质，就让它还作为餐厅的话，你找不到任何可以改进之处了。

这个餐厅之所以这么完美，是因为三件事情，而这三件事对软件开发来说，绝对是值得借鉴的经验。第一：重视所有的细节，尽量做到无错；第二：近乎无缝的团队合作；第三：客户至上。

## *细节*

在软件系统中，由于交互的复杂性和细致性，迫切需要注意细节问题，但是，看起来似乎我们做得远远不够。在无数的产品中，尽管其他方面设计的非常良好，但由于某些细小的疏漏，最终毁了产品。一个重要的对话框与其他的界面风格极其不一致。或者因为缺乏一个选择项，使得选取阴影颜色成了一件很困难的事情；或者文档的每一部分都需要用户自己定制。有时，甚至连商业产品的设计看起来都像是随意而为的。这样的例子很多，咎其原因，不是因为使用了错误的编程语言，也不是因为使用了错误的方法，而是因为没有给予足够的重视。只有特别重视细节问题，才能克服这些错误——我们甚至可以采用强制的手段。

在 Ristorante Alle Murate，各种细节都得到了充分的重视，以至于你都感觉不到他们是如何做的。每一道菜都是一个微型艺术品；只要你把叉子放到空盘子上，立刻会有人把它收走；每份饮料、每瓶葡萄酒都用不同的独特的玻璃杯端上来。但是，你既不会感到烦琐也不会感到做作，一切都是那么恰到好处。

## 工作团队

我经常写一些关于团队工作的文章，而且如果能在现实中看到一些好的协作团队，无论他们是音乐会上的演奏团队，还是百货商店中的销售人员，我都会感到很高兴。在餐厅里，我很少能碰到好的团队。但是，如果是好的团队，哪怕是餐厅里那种一板一眼的工作，例如，由侍者将客人领到餐桌上这种完全由单人完成的单调的行为，都会让人觉得很舒服。在软件开发中，也存在这种排他性很强的工作。在进行系统体系结构设计时，某个人可能是唯一经过授权的专家；或者，某个程序员在编程时，将一个子系统变成了他个人的专属品，在这些情况下，团队间的协作都会遇到问题的。

但是，在这家餐厅中，却有所不同。屋子很小，每个人各司其职。餐厅的负责人可能是 Umberto Montana，但是所有的员工表现得好像这家餐厅就是他们自己的一样。

在最好的技术团队中，有各种各样的专家。但是，专家的存在并不意味着相互之间不能进行协作或者不能灵活的工作。举个例子，Sonia 是一个专业的葡萄酒品酒师，她可以精确地鉴定每瓶酒的属性。在品尝醇厚的托斯卡纳红酒前，必须先让它“呼吸”一下空气，所以她先倒出少量的酒，在一个深红色的杯子中来回旋转，随后，她不断地向杯子里倒酒，并来回旋转，她每次都只增加 1 毫米。然而，如果 Sonia 比较忙——她在向我们解说那道精美的用鱼做成的菜肴——会有其他人过来帮助，他们会将酒拿到另一张桌子上继续工作。

米开朗基罗是意大利最伟大的艺术家之一，他的才能之一就是在艺术家和工匠中培养一支好的团队。Umberto 很可能继承了这种才能，他简直就是“绕行管理”的代表。他在屋子里来回走动，时不时地与客人们交谈几句，很有教养地观察着整个就餐过程，并小心翼翼地补满所有需要的东西。真的，所有的员工看起来都在监督着餐厅里的一切事情，他们会及时注意到需要补酒的玻璃杯或者是自己走过去将其加满。

这种流动的监督不但可以注意到所有的细节，而且这种既重视又不声张的集体管理还是一种非常有效的节约模式，我对此深有感触。也许，我们应该把这种方式应用到软件开发中，应该更多地依靠那种持续的、相互的评审和检查，而不是依靠零星的、非常不力的固定结构的走查。

甚至客人也成了这个团队的一部分。订餐的过程就好像是需求定义。你想要什么样的葡萄酒？也许您可以先上这道菜，您会喜欢烤羊羔肉的。Umberto 甚至可以当场发明出适合你口味的菜肴并且配上一瓶特别的葡萄酒。他转身走到厨房的窗口，与厨师讨论几句。噢，她今晚无法做出这道菜了，但是，您看看换成烤鸭怎么样呢？当他重新回到桌边时，他可能会提出一个新的建议。

## 在危急中

这家餐厅之所以如此完美，是因为它那种既老于世故又难以捉摸的魅力，而这种魅力正是由于它所提供的服

务而产生的：它并不是给客人们所想要的，而是给客人们他们所需要的。想想吧，软件开发中的那个年代，软件完全变成了一种“膨胀功能”的系统，每每一个软件系统的所谓客户列出的“想要的功能清单”上有 900 多行功能点。好好想想两者之间的区别吧，这样我们可以理解得更深刻一些。

吃一顿正宗的意大利菜需要好的葡萄酒。事实上，可能不止一种葡萄酒：在吃意大利通心粉或者吃鱼时喝的葡萄酒，如果留到吃鸭子时喝，根本就不对味；在吃 Dolci——一种甜食时，需要一杯 Malvasia 意大利苏打白葡萄酒或者 Brachetto。然而，并不是所有的用餐者都愿意在吃不同的食物时饮用不同的葡萄酒，其实很少有人会要一种以上的葡萄酒。

所以，除非你自己拒绝，当你刚一到达时，侍者就会送上一杯白葡萄酒，帮助你振奋一下精神，同时也作为一杯开胃酒。当你吃 Dolci 时，他们会送上另外一种免费的白葡萄酒。

在软件中，一个设计良好的用户界面不会给用户提供所有他们希望的功能，但应该以最简单的方式提供所有可以帮助用户完成任务的功能。一个好的设计应该是一个细致的妥协的产物——客户认为他们想要的，以及真正对他们的工作有帮助的。这也正是现代的以用例为中心的设计方法（参见文献 Constantine & Lockwood 1999），和被取代的以用户为中心的设计方法之间最本质的区别。

这家餐厅的菜单也是非常简洁。你可以根据情况自己选菜，或者你干脆选一个包桌，充分的信任 Umberto 和他的员工们一下，最后，你还是会非常惊喜的。

在这个例子中，信任是十分重要的。由于老于世故而又小心翼翼，Umberto 可以帮助客人们远离那些他们不喜欢的葡萄酒，或者那些不适合同时进食的菜肴。就像一个最好的咨询顾问，他能够很快地意识到他的客户到底是谁。当我们向他征求该喝什么葡萄酒时，他认出了我们，我们上次在这里用餐时喜欢喝比较有劲的红葡萄酒，所以这次，他帮我们选了一种很棒的来自于一个北部小地方的“浓”葡萄酒。

当然了，并不是每一个人都认为餐厅是完美无缺的。坐在我们旁边一桌的是一对美国夫妇，可能是来自哪个大城市的。他俩可真是天生一对，都决定由自己来掌控一切。他拒绝由乌博图来帮助他们从酒窖的上千瓶葡萄酒中挑选一瓶，而是要求给他们单子，由他们自己来挑选，而且，他们也没有接受选菜的帮助。当他俩离开时，从他们的脸上可以看得出，他们达到了“自己掌控”的目标，但是，这顿饭他们却没有吃好。

Umberto 说，每个星期，大概会有这么一对夫妇，但绝不会有第二对。到底是不是如此呢？因为我们在佛罗伦萨的时间比较短，没时间等到下一次了。

摘自《软件开发》，第 3 卷，#8，1995 年 8 月


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

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

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