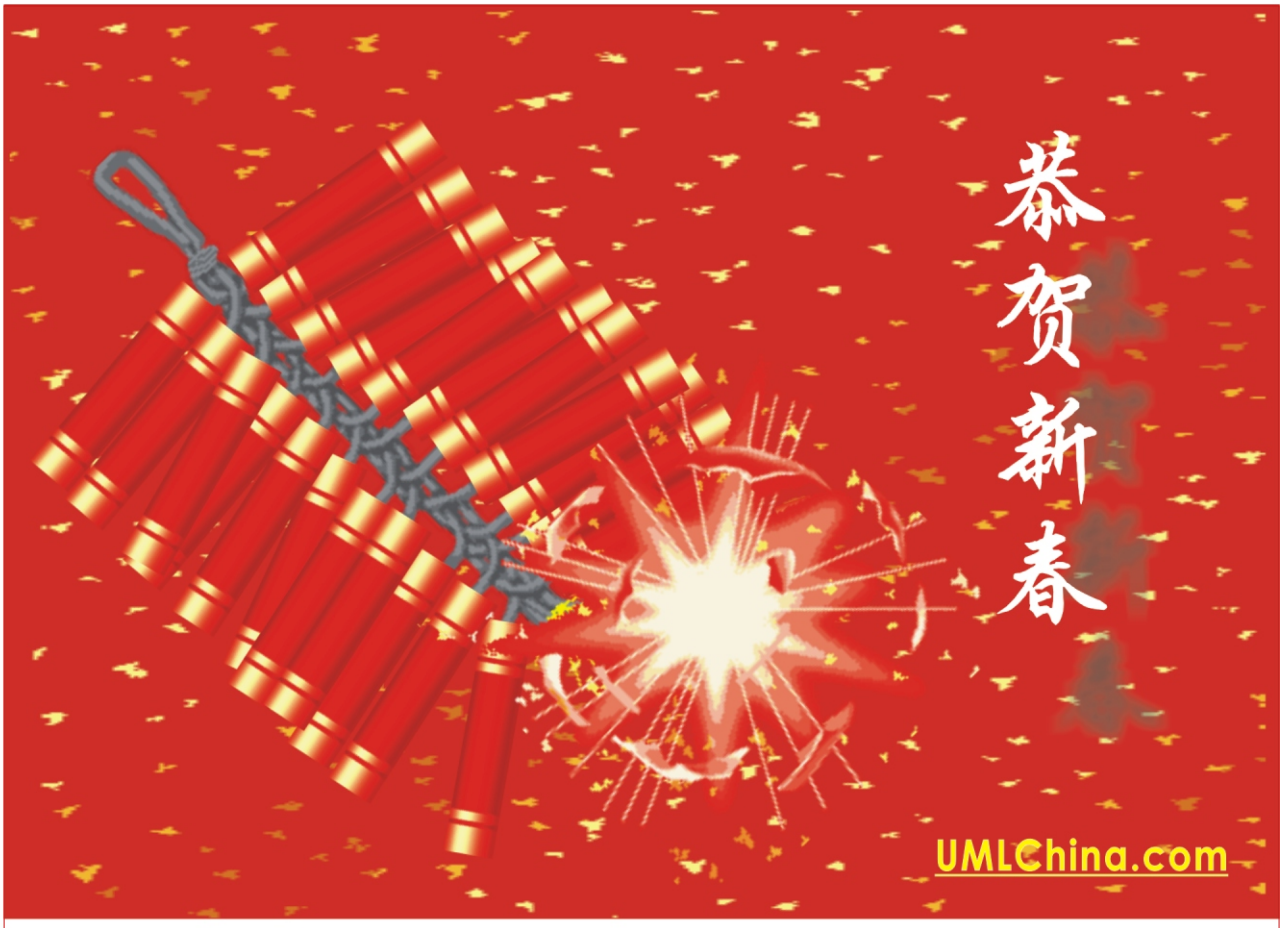
X-Programmer 非程序员

软 件 以 用 为 本

2003-1
第21期

>>目录

投稿：editor@umlchina.com
反馈：think@umlchina.com
下载：http://www.umlchina.com/xprogrammer/Index1.htm



祝：各位读者

新年一切如意！

UMLChina
羊年新春贺

年度回顾：开发工具的 UML/MDA 趋势

[2002/12/31]今年市场上的小小风雨只是 2003 年 IDE 大战役的序曲而已。

2002 年，在程序开发工具市场上，发生了一些骚动和合并。但这仅仅是来年不同集成开发环境（IDE）之间大战役的一个小小序曲而已。

IDE 公司的并购实际上只是集成工具范围扩大的一个表现，现在的趋势是：IDE 都试图控制从开发、部署到维护的整个生命周期。

这些举动基本上都是对 Microsoft 的 Visual Studio.Net IDE 试图垄断未来 windows 开发的一个回应。其它的 Windows IDE 也纷纷表明立场，最典型的态度莫过于声称各自的跨平台特性了。

2001 年末，IBM 通过 Eclipse.org 的形式支持其 Eclipse 开源 IDE，今年 Eclipse 2.0 已经诞生。

最前途未卜的公司应该算是各底层开发商，包括 IBM、Borland、Rational software 和 TogetherSoft。

岁末年初，Borland 正忙于吸收 TogetherSoft，IBM 同样也有 Rational，而他的拍档 Microsoft 却被不愉快地晾在了一旁。

BEA 靠其 J2EE 开发框架也加入了这支队伍，但其分支 WebGain 却拉了下来。

另一个关键的趋势就是 Model-Driven Architecture (MDA)的快速崛起，MDA 的目标是要实现从 UML 企业化模型到最终代码的自动化生成。

MDA 板块的第一批队伍有德国 Interactive Objects 的 ArcStyler，以及 Rational 购买的 NeuVis ，它应该会增强 IBM 在 UML/MDA 方面的能力。

类似的，Borland 对 BoldSoft 以及后来 Starbase 的收购保证了其 IDE 的 UML/MDA 支持，另外，Compuware 公司基于 MDA 的 OptimalJ Java IED 发展也非常迅速。

现在，IBM 收购 Rational 事件过后，所有的目光都聚集在 Microsoft 身上，等他在 UML/MDA 上的出招，同时还有一个类似的新问题，就是关于如何使用业务过程建模语言 version 1.0 进行建模。

2003，必将好戏连台。让我们拭目以待。

（自 vnunet，风自由 摘译，仅供学习交流，不得转载用于商业用途）

Visual Studio .Net 2003－－“Whitehorse”蓄势待发

[2002/12/20，Redmond]尽管关于开发工具市场方面可能进行的收购谣言不断，Microsoft 在 Visual Studio .Net 的主要升级版上的努力从来没有停止，其新版本在企业化方向又迈进了一步。

升级版本的代号是 Everett，在上周被正式命名为 Visual Studio .Net 2003。这一版本有望与 Windows .Net Server 2003 同期发布，后者的计划发布日期是在四月。

预计在 2 月 9-14 日 San Francisco 的 VSLive 开发者大会上，Visual Studio. Net 2003 将会受到大肆吹捧。

Visual Studio .Net 2003 中将包括对企业开发人员来说非常有用的一些企业化特性，例如对移动设备和企业数据库的支持。Visual Studio .Net 2003 企业开发者版本提供了团队开发方面的支持，并简化了应用的部署以及和 Oracle 及 ODBC 适配数据库的连接。Visual Studio .Net 2003 Enterprise Architect 版本还提供了基于 Visio 的 UML 建模以及数据库建模方面的支持。

一些观察家认为 Visual Studio. Net 2003 缺少企业用户呼声很高的智能建模、测试以及变更管理等功能，而其它人则认为微软不会通过出更高的价格从 IBM 手中夺过 Rational 或者购买 Borland，而会继续通过和第三方的联盟来填补这个空隙。

微软计划在 Whidbey 版本的发布中提供一个新的设计工具和框架，代号为 Whitehorse，它是类似于 Rational 的建模工具。资料上说，“Rational 和 Whitehorse 都是通过抽象模型或者可视化过程来装配一个应用，但在 for Whitehorse 的 Visual Studio 和.Net 框架中，联系会更加紧密。”一位开发人员这样说，“使用 Whitehorse，微软将为习惯于 Visual Studio ‘design-build’过程的开发人员展示建模的艺术和可视化装配”。

Borland 和 Rational 都没有对有关 Microsoft 的收购谣言作任何评论。

（自 CRN，风自由 摘译，仅供学习交流，不得转载用于商业用途）

《人月神话》与软件开发研讨会在京举行

20 余年来，《人月神话》毫无争议地被公认为软件工程领域的必读经典。自本书第一版面世以来，一直维持着稳定的销量，这在变化神速的计算机书籍中是极其罕见的。书中的许多观点被广泛引用、发展，业界对作者 Fred Brooks 各种观点的争论也一直从未停息，其中原因也许只有一个，《人月神话》是一本帮助软件工程师思考的著作。《人月神话》中文版推出后，引发了业界的普遍关注。在互动出版网，《人月神话》分别创下单周、单月销量历史新高，在冬日萧瑟的计算机图书市场，创下多项销售神话。软件业也许是变化最快的行业，技术人员没有人不需要读书，软件工程师更需要在不断学习的过程中不断思考。清华大学出版社、用友软件股份有限公司、互动出版网、UMLChina 在 12 月 28 日联合举办“《人月神话》与软件开发基本问题”的研讨会，一起借这本书回顾过去，结合实际，展望未来。

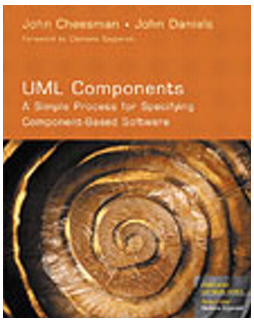
研讨会在清华大学东门外的学研大厦多功能厅举行，与会者将近 300 人，绝大部分是来自各个软件公司的开发人员和管理人员。会议在下午 1:30 开始，清华大学出版社和互动出版网的领导分别致词。随后，《人月神话》的译者，UMLChina 成员汪颖作了题为“《人月神话》与实践”的演讲。研讨会以圆桌论坛的方式举行。论题分别为“《人月神话》与当前软件开发实践”和“《没有银弹》与软件工程的未来”。

在第一个论题“《人月神话》与当前软件开发实践”中，用友软件的技术专家郑雨林教授针对《人月神话》中的“概念完整性”、“贵族专制、民主政治和系统设计”、“全新的软件产业”，结合用友的实践，向听众介绍了用友公司的发展方向：重视软件架构、重视行业和领域经验、作行业领域的应用框架、走产品路线等。摩托罗拉中国研发中心的崔峰也结合摩托罗拉中国研发中心在 CMM4 和 CMM5 评估中的实践发表了精彩的见解。奥捷特通信的刘天北则针对“外科手术队伍”一章谈了自己在开发团队组建方面的经验体会。在第二个论题“《没有银弹》与软件工程的未来”中，四通科技开发公司的技术专家欧阳巨星就软件开发过程中是否存在银弹以及软件工程在当前软件开发中谈了自己的看法.来自朗讯科技贝尔实验室的钱岭博士则根据自己的经验教训针对软件度量、新出现的银弹、软件团队构成发表了独到的见解。最后，软件工程与软件管理领域的高级咨询顾问周浩宇根据当前软件开发中软件工程文化缺乏的现状阐述了软件文化对软件项目的重要性并指出了相应的对策。

在研讨会进程中，专家和台下听众一起，针对两个论题进行了深入交流，会场气氛异常热烈，全场掌声、笑声不断。

《人月神话》的姐妹篇，Tom Demarco 和 Tim Lister 的《人件》第二版（“Peopleware：Productive Projects and Teams”）将于 2003 年上半年由清华大学出版社出版。《人月神话》关注“软件开发”本身，《人件》关注软件开发中的“人”。届时将再次举行类似的研讨会。目前，《人月神话》专题网站 mmmbook.com 已经开通，专门收录关于《人月神话》的各种资料。目的是使读者能更好地吸收《人月神话》书中的营养，以及有一个互相交流的空间。

2003 年 1 月 UMLChina 专家交流预告

软件过程 1 月 16 日（星期四）上午 9:30-11:30	 Barry Boehm	嘉宾介绍>>
UML/J2EE/.NET 1/22 日（星期三）上午 10:30-12:30	 John Cheesman	嘉宾介绍>>
软件需求 1 月 28 日（星期二）上午 9:30-11:30	 Ellen Gottesdiener	嘉宾介绍>>

Jutta Eckstein：没有客户就没有项目

查看评论

编注：

北京时间 8 月 30 日下午 15:00-17:00，Jutta Eckstein 女士作客 UMLCHINA 讨论组的聊天室。Jutta Eckstein 住在慕尼黑，是一名独立咨询专家，有超过 20 年的面向对象开发经验。她是 Hillside Europe 委员会成员，XP 2002, EuroPLoP 2002, OT2002 和 OOPSLA 2002 委员会成员。人民邮电出版社出版的《极限编程研究》（XP Examined）有她的文章“特洛伊木马中的 XP：重构统一软件开发过程”。以下是交流实录。由曲俊生翻译。原文链接。

Lhhwanmit：我想在我的项目中使用 XP，但是 XP 适用于小型的项目。我可以修改 XP 的哪个部分来运用于我的项目？

Juttaeckstein：你的项目有多大?

Lhhwanmit：大约 600 个人月。

Juttaeckstein：项目团队中有多少人？

Lhhwanmit：大约 30 人。

Juttaeckstein：所有或者部分开发人员同时也是业务人员吗？

Lhhwanmit：是的。这是一个政府项目。

Juttaeckstein：你是否一定要与拥有这么多人的开发团队工作？

Lhhwanmit：哈哈，您真幽默。我是这个新团队的领导者。

Juttaeckstein：这样啊，那么为什么你希望由 30 个人组成开发团队？是否少数的人会更好？

Lhhwanmit：JAVA 组有 8 个人。Notes 组有 5 个人。有 3 个领导者。因此，总共有大约 30 个人。

Juttaeckstein：那么，lhhwanmit，对于我来说，如果要运用 XP，你的项目并不大。

Lhhwanmit：哦？那么您的意思是说 XP 对于这个项目来说是完美的开发过程。

Juttaeckstein：是的，我认为团队的大小不成问题。同时，需要进一步确认的是开发人员应该坐在一起，这样能够保证沟通。

Ool：您能描述一个利用 UML 设计项目模型的一般过程吗？

Juttaeckstein：我不是十分确认您所谓的项目模型的含义。我在软件开发过程中，大部分是使用 UML 进行设计，而且这样做是行之有效的。

Ool：我指的是一个一般项目的模型，同时我想了解的是构造模型的一般顺序。

Juttaeckstein：对于构造一个模型的一般顺序而言，通常的做法是建模。这一点可以通过 Story Cards 和沟通来进行。但是，有些时候 UML 可以给你一个更好的概览。只要你需要它们，其它的模型会依次生成。比如说，当你操作一个场景的时候，你首先对这个场景建模，接着编写测试代码和实现代码。有一点你需要在项目开发就应该确认的是，有一个测试小组。他们应该在开发者进行代码编写和单元测试的时候，为这些场景定义接受测试。

Gigix：您如何看待软件工程？它是一门科学吗？

Juttaeckstein：你说软件工程是否是科学？它是科学，但是我觉得更接近的说法是它是一门艺术，一种工艺，同时有一点制造的意思。

Lhhwanmit：哦，软件工程是个哲学问题。

Juttaeckstein：你说的完全正确。

Andiyang：嘿，juttaeckstein，我希望了解如何利用 XP 来构建模型？

Juttaeckstein：你说的是哪种模型？

Gigix：在软件工程中，我们谈论的大部分内容是关于“如何”，而不是“为什么”。尽管我们利用一些软件工程的方法解决了一些问题，下一次，我们能够证明我们能做到吗？

Juttaeckstein：Hmm，我不同意。我们也经常问为什么，否则，我们就不能发现任何模式。我们下一次要做什么，gigix?

Lhhwanmit：谢谢您回答我的问题。您同意软件工程是一个哲学问题吗？

Juttaeckstein：是的，我同意。

Gigix：我的意思是说，有没有任何方法来证明我们的方法是正确的？

Juttaeckstein：只要你成功了，任何项目都是证明。那么，什么是成功呢？答案是客户很高兴。

Andiyang：嘿，juttaeckstein，在 XP 中，哪种图用的最多？

Juttaeckstein：我们没有必要使用所有的图。只是使用那些有利于工作的图。我个人喜欢使用类图和协作图。但是，尽管我非常喜欢它们，我并不是所有的时候都使用它们。

Ool：XP 中的 User Story 与 UML 中的用例有何异同？

Juttaeckstein：简短的回答：如果，你是对用例图而言，两者并无不同。User Story 更多依赖于与客户的沟通（因为无论如何，她会更改主意）；而用例则更多依赖于仔细生成的场景。

Gigix：我有一些关于模式的问题：有时，我利用一个模式解决了某个问题，但是，下一次，我在一个类似的问题上使用相同的模式却失败了。因此，我想要了解的是：我如何才能证明某个模式适合某种问题？您能够给出一些建议吗？

Juttaeckstein：不能。实际上并不存在什么灵丹妙药。仔细查看模式的背景和约束，以确定它是否使用你所面临的问题。真正去验证一个模式的办法就是去尝试使用它。如果它并不适合，那么就不用它。我希望你设计的系统应该足够的简单（你应该有测试）。以至于模式不是问题。

Mycluster2001：请问 UML 是否提供将业务模型映射到软件架构的功能？我感觉，创建一个描述应用的模型并不能自动保证它的正确性。

Juttaeckstein：我不是非常清楚。但是，我无法确定这个过程如何自动进行。软件开发中有一些事情值得我们仔细考虑。

Ool：在 XP 中使用 UML 是否比不使用更好？

Juttaeckstein：不同的人采用不同的策略。尽管我使用 XP，但是在 UML 有利于我工作的时候，我会使用它。另外一些人在头脑中勾勒模型，而不是写在纸上。

Mycluster2001：您是否了解专业领域模型的相关事宜？

Juttaeckstein：非常正确。一个模型是否正确只有在你编码之后才能证明。因此，不要长时间地建模，你应该为你的模型编写测试。

Gigix：Eckstein 女士，您是否有代码重构的实践？是否做过 C++代码的重构？

Juttaeckstein：我做过。但是，老实说，这是很久以前的事情啦。在过去的五六年中，我并没有使用 C++开发任何东西。为什么这么问？

Mycluster2001：根据我的实践，我通常希望将抽象模型写下来。这些模型也许会贯穿编码的很多部分。

Juttaeckstein：我也喜欢将模型写下来。我尽量不是它们很抽象，否则利用代码就很难去证明它们。进一步讲，如果你建立的模型太抽象，以至于你要假定一些不必要的事宜。只建模和开发用户要求的东西。

Gigix：您能解释一下代码重构吗？我对它并不是十分理解。谢谢。

Juttaeckstein：代码重构是指改变代码使得内部结构提高，而外部行为并不改变。

Athlonshi：“不要过早加入功能”这句话您如何解释？

Juttaeckstein：“功能只是在需要的时候再添加”是指只有顾客要求这样时才这样做。尽管你相信这个功能肯定会需要，但是你永远不知道，用户是否会改变主意。

Gigix：C++语言是一种编译型语言。这就使得 C++程序更准确地表达问题。那么我们如何证明我们在 C++中所作的重构是正确的呢？

Juttaeckstein：你能够获得的关于代码重构的最重要支持不是语言，而是工具。我听说一些人正在为 C++开发代码重构浏览器。但是，我不清楚它现在是否可以使用。请密切关注 refactoring.org.

Mycluster2001：完全正确。但是，有时候我可能需要在系统中加入更多的组件，而这些在我们给用户查看的模型中并不存在。我想，为了获得可配置性、性能或者只是 porting 的原因，这都是必要的。

Juttaeckstein：你没有必要将模型展示给客户看，而是应该将运行的系统演示给他们，客户并不关心代码和开发人员。只有模型有利于软件运行，客户会同意的。进一步说，如果客户需要考虑性能方面，你就应该做到。同时注意，软件开发应该是首先使它运行，而后使它正确，最后才是使它运行得最快。

Ilobebear：什么是 XP?

Juttaeckstein： XP 是一种过程，它将以往经过验证的技术（最佳实践）整合起来，从而在软件开发中方便帮助我们和客户。

Ilovebear：我知道啦，我怎样才能获得更多关于 XP 的信息？

Juttaeckstein：请查看 http://c2.com/cgi/wiki?ExtrexePregrammingRoadmap, http://www.extremeprogramming.org/。另外，Kent Beck 关于 XP 的第一本书《eXtreme Programming explained: Embrace Changes》是了解它的一个很好的开始。

Mycluster2001：我认为，正确的模型是任何项目的开始。尤其是对那些需要开发很长时间的软件项目来说更是如此。

Juttaeckstein：我不这么认为。因为我在一个项目的开发从来没有看到一个正确的模型，而是在项目结束的时候。

Mypine：XP 是否意味着我们否定了设计？

Juttaeckstein：不是。恰恰相反，XP 中有很多地方涉及到设计。但是，XP 承认在我们使用编码证明它之前，我们永远不能给出正确的设计。

Mycluster2001： XP 需要在软件开发的过程中调整模型和相关的代码和数据。因此，如何保持这项技能就是一个很大的问题。

Juttaeckstein：你只需要维持你的团队人员。在我的项目中，我经常使用 UML 图来表达我想要做什么。但是，随后我就不会使用它。如果随后我要使用它的话，我就维护它。

Mycluster2001：这是否是配置管理呢？

Juttaeckstein：不，不是配置管理。重要的是，你能获得不同的版本。配置管理并不能帮助你生产这些版本。

Mypine：在 XP 中，什么是最重要的？

Juttaeckstein：这个问题难以回答，因为任何事情之间的关联都很强。我想，我要说的是沟通。为什么？因为我所看到的项目之所以失败，真是因为沟通的问题。

Mypine：您认为在我的项目中应该如何开始使用 XP 呢？

Juttaeckstein：关注最严重的问题，使用 XP 技术来尝试解决。通常情况下，但是测试是一个很好的开始，它是重构的前提条件。

Mycluster2001：我有一个问题是关于组件或者对象以及基于操作系统的传统过程。我认为这两项技术本质上并不相同。或许我必须在对象中封装几个进程。

Juttaeckstein：通常，你可以使用 XP 构建一个 COBOL 系统；而使用线性方法（例如瀑布模型）构造一个 RUBY 系统。一个过程与另外一个并不相关。但是，如果你在使用一门更现代的语言，那么更现代的软件开发过程会使你获益更多。（重点在于“更”）

Mycluster2001：这将不利于我使用通常的设计模式来建立框架。例如，对象 A 被设计为使用 B 的接口，但是它们不在同一个进程中。我不能确认 A 是否能够直接调用 B，因此它们在不同的内存区域。也许进程并不是问题。

Juttaeckstein：那么，问题在那里呢？

Mypine：用户在 XP 中的作用是否是项目先决条件？

Juttaeckstein：人在任何项目中都是先决条件。否则，你怎么知道你开发的是需要的呢？当然，有时候你必须寻找客户，匿名的客户（例如，市场部门），或者并不总是能够找到的客户的需求。但是，你总是需要在与客户或者顾客保持紧密关系的基础上进行开发工作。

Eurekahy：如此多的面向对象编程技术之间的区别是什么啊？

Juttaeckstein：有一些很僵硬，不灵活，它告诉你先做什么，再作什么；而另一些则允许你改变软件开发过程，使之与你的项目相适合。

Eurekahyj：PSP 对一个团队是否适合？

Juttaeckstein： PSP 适用于个人，而个人对于团队十分重要。但是，对于整个团队，你也许想看看 TSP。

Mycluster2001：因此，我需要使用 corba 或者其它别的什么。

Juttaeckstein：你真的确定你需要使用 CORBA?我只看到少数项目中需要使用 CORBA，并且 CORBA 真的起到一定的作用。在其它大部分系统中，人们假定他们需要使用 CORBA，并且认为它会给系统带来更多的灵活性。但是结果却是系统比需要的更加复杂。

Mycluster2001：但是我必须更多地了解特定中间件系统，而不是 UML。如果我不使用 CORBA 的话，我可以利用其它方法。但是，它们当然不会简单。

Juttaeckstein：或许它们可以更简单…

Eurekahyj：面向对象语言、软件开发过程、代码重构和 XP 之间的关系是什么呢？

Juttaeckstein：代码重构是 XP 中的一项技术，但是也可以使用在除 XP 以外的软件开发过程中。通常意义上讲，你可以将任何语言与任何软件开发过程关联，但是，XP 经常是和面向对象的语言结合使用的。

Mypine：对不起，我觉得你们不应该问到 juttaeckstein 诸如像 Are you sure you need corba?这种问题，这样很浪费大家的时间。我建议大家将问题集中在 XP 的实施及其它的敏捷方法或重型方法的比较上，而不是技术上的选择！

Juttaeckstein：这正是我想说的。CORBA 并不是万金油，出于复杂性的考虑，你必须仔细斟酌。当你有所疑问的时候，不要使用它。

Mypine：你知道用户是上帝；但是实际上，用户是不讲情理的上帝。因此，用户的角色在 XP 中很难实现。

Juttaeckstein：但是如果你希望忽略客户，你为什么去开发一个系统？没有客户，没有项目；或者说，没有客户，就没有成功的项目。请你牢记：客户是每个项目所必须的，而不仅仅只是 XP 的项目。

Eurekahyj：XP 适合 CORBA 还是 COM？或者，它们根本就不是一回事？

Juttaeckstein：如果你使用 CORBA，你也必须编码。因此，你的问题是什么呢？

Eurekahyj：我还不清楚。我的意思是说在构建一个系统的方法之间必定存在每种关系。

Juttaeckstein：是的。但是，你可以为任何技术使用任何开发过程。如果你决定使用 XP，而系统需要使用 CORBA，没有问题。但是，请时常提醒自己是否真的需要复杂的技术。

Mypine：是的，没有客户就没有成功。当我们在项目中应用 XP 时，应该注意什么？能给我们一些建议吗？

Juttaeckstein：查看项目中最大的问题，选择一项 XP 的最佳实践去实施。请确认项目组中的每个人对每件事情都十分清楚。

Newdongkui：我阅读过一些关于 XP 的文档，其中有一句说道“一个项目应该被分成更小的部分，其中每部分的生命周期小于三个月”，您对此如何理解？

Juttaeckstein：实际上三个月的周期还是太长了。使得周期尽可能的短，从而可以尽快地获得关于系统的反馈，这总是一条好建议。

Eurekahyj：讨论总是将我们之间的距离缩短，但是更多的问题会产生。谢谢您的慷慨。

Juttaeckstein：如果我需要很长时间来回答，请不要失望…

Newdongkui：当然。我曾建议我的老板采用成对编程，同时每项工作的时间不应长于两周。

Juttaeckstein：确认你每天都有一个运行的系统，每周都有少量的新功能，每个月都有主要的大功能。

Ool：您能否谈论一些关于编码前测试的相关事宜?我对此不是很了解。

Juttaeckstein：如果你指的是实际编码前的测试，那么它可以促使你考虑需求以及为你希望实现的系统设计接口。

Eurekahyj：编码与某些生物技术，例如解剖或者 DNA 技术有何类似吗？

Juttaeckstein：没有。

Eurekahyj：我目前正在考虑建立一个系统处理环境生化学的问题，但是我的专业与计算机无关，这是否是个大问题呢？

Juttaeckstein：我不认为如此，除非你是软件开发的专家，而你的小组中没有任何其他人。但是，尤其对于 XP 而言，知识正是通过成对编程实现传递的。

Newdongkui：非常感谢。我想会有更多的问题发送到您的邮箱，请帮助我解答。

Juttaeckstein：你看到了我前面给出的电子信箱？jutta@jeckstein.com。

Xftang：在 User Story (XP) 和用例 (UML) 之间有任何区别吗？

Juttaeckstein：比较长的回答，请查看 Alistair Cockburn 的书——《Writing Effective Use Cases》，以及查看他的站点；比较简短的回答就是，没有主要的区别。但是，一个大的区别就是一个场景（UML）和沟通（XP）。

Eurekahyj：因此一个开发团队应该是有多种角色，有一些多面手？

Juttaeckstein：是的。

Xftang：您如何看待沟通在 XP 中的重要作用？

Juttaeckstein：我的观点是它是 XP 中最重要的价值。

Zhaowenbin：什么是 XUP?

Juttaeckstein：对不起，我不是很清楚。我相信你指的是 Craig Larman 对于 XP 的更改。

Xftang：如果没有文档，我们与其他人如何沟通？我认为在中国，我们与客户之间沟通是很难的。

Juttaeckstein：直接进行如何？只是与人交谈。就像 Jim Highsmith 说的那样，文档与理解是两回事。为什么与客户沟通很困难？为什么在中国比在其它地方更困难？

Eurekahyj：XP 的未来如何？它是否会转变为另一种技术？就象它从其它技术发展来一样。

Juttaeckstein：你永远不会知道。但是，XP 的价值和最佳实践已经存在很长时间，我相信它必将存在得更长久，或许以另外一个名字存在。

Founder_chen：我想要创建一个 XP 的团队，我首先应该做什么？

Juttaeckstein：小步骤开始。确认每个人都在开放、诚实地沟通；小组中的人拥有多项技能；寻找一个 XP 的指导人员；拥有现场客户。

Xftang：我可以打赌，我们不可能没有任何文档。

Juttaeckstein：如果你确实需要文档，为客户编写故事卡片，请客户为它们划分优先级，对它们进行评估，同时考虑在客户制定的时间表中撰写它们。

Ool：保持设计简单是一件很难的事情，是吧？能否谈论一下这方面的事情？

Juttaeckstein：是的，这需要很多原则。首先编写测试代码会有很大的帮助，只是实现需要的功能也是如此。

Founder_chen：-_-，您如何看待设计模式在 XP 中的作用？

Juttaeckstein：有时，代码重构会引导你到设计模式。但是，不要首先根据设计模式设计代码（或许它们会引入比需要更多的复杂性）。

Xftang：我们的客户认为构建一个软件系统与他们毫无关系。

Juttaeckstein：但是，你的客户需要的是系统。因此，不要向他们展示代码、模型，而是运行的系统。另外，请他们为故事卡片进行优先级评估，对他们而言，什么功能是最重要的？

Eurekahyj：简单是美，但是这个世界有很多丑陋的东西，因此简单就…

Juttaeckstein：是的，完全同意。简单是美，这也是它为什么这么难的原因。

Ool：“拥有现场客户”如何？

Juttaeckstein：你应该尽量保持拥有现场客户。有时，你需要通过市场部与匿名客户一起工作。如果有一组客户，请他们派一个代表。如果他们不能每天在这里，那么至少应该是 50%的时间在；其它的时间应该可以通过电子邮件或者电话找到他们。

Xftang：是的，我们能做到这一点。但是，每件事情都是通过电话联系，客户不能每天都参加我们的团队。我认为没有文档是实施 XP 中最大的障碍，管理部门是不会同意的。

Juttaeckstein：如何客户要求有文档的话，你就应该撰写文档。这是另外一种需求。要求客户每周至少在开发现场一至两天。最低的要求是每当有少量的功能开发完毕的时候，客户应该在那里，以便可以提出他们的反馈意见，同时决定下一个开发周期做什么。

Founder_chen：“现场客户”，难道需要一个 CRM 系统来处理客户的问题吗？

Juttaeckstein：你为什么不和他们直接交谈呢？你为什么需要一个系统呢？我自己的实践证明，如果你不直接与客户交流，你总是会遗漏一些东西。

Eurekahyj：向客户展示界面并告诉他们你的系统能够做什么，这就够了。客户需要做的就是——点击按钮。我听说也许日本的开发人员开发编写完美的文档。在日本公司里，文档的格式是固定的。

Juttaeckstein：重要的问题在于客户是否将撰写文档看得比开发一个运行的系统更加重要。对于一个运行的系统，我的意思当然是开发一些新的功能。

Luofat：为什么我在所有的代码重构工具中没有发现“Move Method”？

Juttaeckstein： Eclipse 有这个方法，IntelliJ 也有，Smalltalk browser 也支持这个方法。你查找的是什么工具呢？

Xftang：除了在小型的公司，XP 的流行程度如何呢？

Juttaeckstein：嗯，我不认为 Daimler Chrysler 是个小公司，而 XP 正是从这里开始运用的。

Founder_chen：一个团队也许同时维护两个或者更多的项目，而一个客户代理（也许是产品经理）不能获得足够的信息，有时甚至会丢掉一些信息。

Juttaeckstein：我从来不认为让开发者同时参与不同的项目是好主意。可以查看 Tom DeMarco 最新书籍《Slack》（编注：同样在 Tom DeMarco 的《人件》中提到，《人件》中译本将于 2003 年上半年出版）中的这个话题。

Myvrml：我可以在 GIS 中运用 XP 吗？

Juttaeckstein：当然。XP 与领域无关。

Xftang：C3 项目是什么？

Juttaeckstein： C3 是第一个应用 XP 的项目，由 Daimler Chrysler 公司完成。那是一个 payroll 系统。

Myvrml：您曾经看到有那些 GIS 公司运用 XP 吗？能否给我一些建议或者例子？

Juttaeckstein：我自己没有做过 GIS 系统。在最近的 5 年中，我所做的大部分项目是商业项目（财政方面），在那之前，我在制造领域工作。

Luofat：eclipse 的那个版本支持 jrefactory？或者您说的是个插件？

Juttaeckstein：我们正在使用最新版本的 eclipse，它就支持 Jrefactory。

Xftang：没有文档，没有注释，代码重构，持续集成，测试先于设计，所有这些在理论上都很简单，但是实践起来却很难。

Juttaeckstein：你认为其中那个部分是难以实践或者最难实践的？嗯，我认为如果你不尝试，你永远都不会知道这种方式是否比另一种更难。

Xftang：C3 代表什么？是缩写或者其它什么？

Juttaeckstein： Chrysler Comprehensive Compensation。

Myvrml：我们公司的大部分人都不是计算机专业毕业的，他们不喜欢写文档，甚至不懂 OO！我该怎么办？

Juttaeckstein：我认为他们有很多业务知识，这很好。因此，你的团队需要多样性。需要有一些软件开发的专家，将这些人与其他人“配对编程”。

Xftang：其它敏捷方法如何？例如水晶模型。

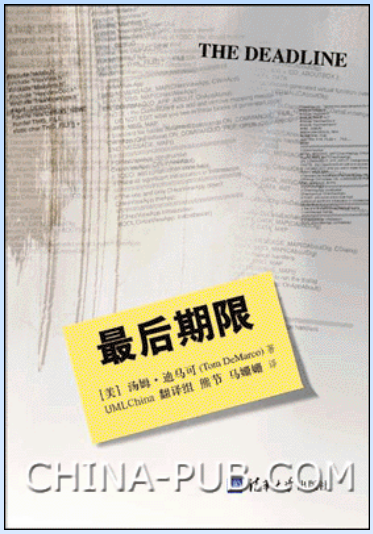
Juttaeckstein：所有的敏捷方法（当然包括水晶模型）都需要与客户的紧密关系，较短的工作周期，及时的反馈等等。

Xftang：开发人员平均每天撰写多少行代码？为什么这么强调 XP?

Juttaeckstein：这个问题的目的是什么呢？在重构的时候，有时他们删除的代码比添加的还多，但是系统的表现提升很多。对我来说，XP 是一个很好的起点。但是，你应该时常关注团队，判断 XP 是否是最适合他们的以及是否需要更改开发过程。哦，我知道一些人正在以 XP 的名义作其它毫不相关的事情（例如，写书），但是 XP 是用来支持软件开发的，它着重于如何建立系统，因此编码是中心点。另外，使用 XP 的方法学习编码比使用其它方法容易，因为结对编程的原因，你总是有一个伙伴在身边。

Juttaeckstein：谢谢各位。这非常有趣。祝你们愉快，同时祝愿你们的项目好运。

《最后期限》



Tom Demarco

翻译：UMLChina 翻译组 熊节

汤普金斯在飞机的座位上翻了一个身，把她的毛衣抓到脸上，贪婪地呼吸着它散发出的淡淡芬芳。文案，他对自己说。他试图回忆当他这样说时卡布福斯的表情。当时他惊讶得下巴都快掉下来了。是的，的确如此。文案……吃惊的卡布福斯……房间里的叹息声……汤普金斯大步走出教室……莱克莎重复那个词……汤普金斯重复那个词……两人微张的嘴唇触到了一起。再次重播。"文案。"他说道，转身，看着莱克莎，她微张的嘴唇，他……倒带，再次重播……

....

"我不想兜圈子，"汤普金斯看着面前的简报说，"实际上你们有一千五百名资格相当老的软件工程师。"

莱克莎点点头："这是最近的数字。他们都会在你的手下工作。"

"而且据你所说，他们都很优秀。"

"他们都通过了摩罗维亚软件工程学院的 CMM 2 级以上的认证。"

傻姑之路：通往职业初段—前言

think 著

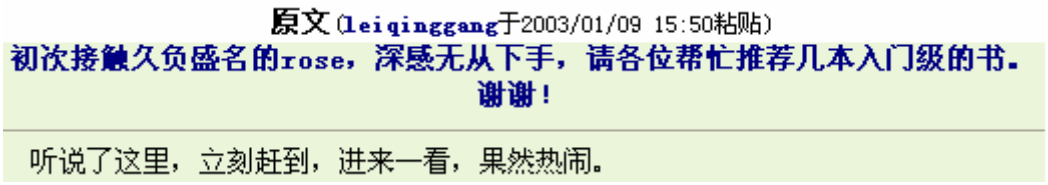
查看评论

当年黄药师后悔一时意气用事，迁怒无辜，累得弟子曲灵风命丧敌手，因此收养曲灵风这个女儿傻姑，发愿要把一身本事倾囊以授。可是傻姑当父亲被害之时大受惊吓，坏了脑子，不论黄药师花了多少心血来循循善诱，总是人力难以回天，.....黄药师知道甚么变化奇招她是决计记不住的，于是穷智竭虑，创出了三招掌法、三招叉法。.....傻姑只练六招，日久自然精纯，招数虽少，却也非同小可。

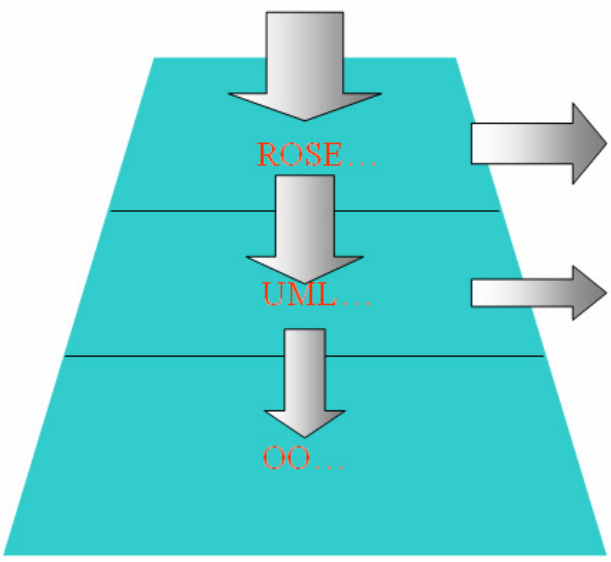
《神雕侠侣》，金庸

本系列文章每期《非程序员》一篇。之所以起名为“傻姑之路：通往职业初段”，原因来自于笔者在实践和培训经历中的深刻感受【1】：国内的软件开发团队，在开发方法上能称得上专业的很少，大多数属于狗刨式。注意我在这里说的是“开发方法”。现在很多公司在抓过程改进，但过程改进只是解决问题的一个方面，改进效果最终会受到开发人员技术水平的制约。也许一家公司在追 CMM，也有需求管理、配置管理、SQA...，但用什么方法获取和组织需求？分析、设计是如何做的？怎么判断工件质量如何？就像中国足球队，有主教练、助教、队医、领队等角色，和巴西队配置完全相同，比塞内加尔队完备，那又如何呢？教练的执教技术、球员的个人技术、队医的医疗技术决定了中国队的结局。“方法改进”和“过程改进”二者相辅相成，不能相互代替。在过程改进的同时，还需要改进开发的基本技术。

那么如何改进呢？

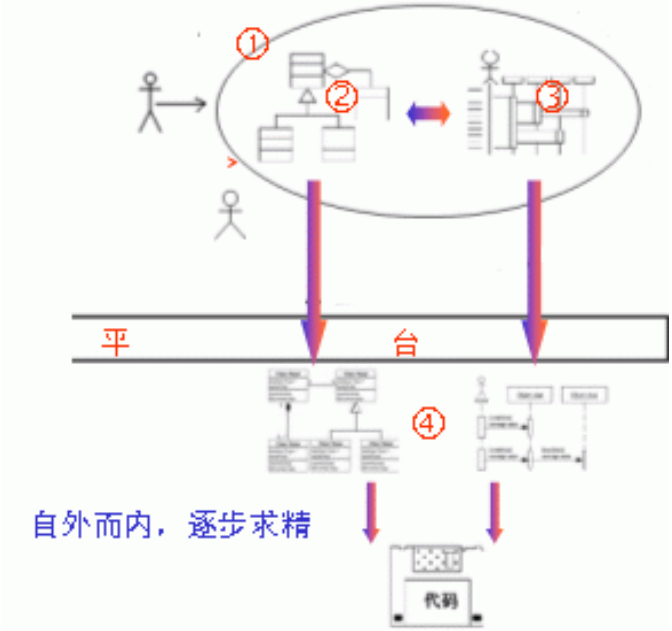


近几年由于 UML 的出现，使得面向对象软件工程的书籍资料迅速增加，为想要改进开发方法的团队起到了启蒙的作用。最开始人们操起 Rose，开始兴趣勃勃的探险，没几下就发现，不会用，用不好。这时，一部分人放弃了，一部分人继续向下探寻，探寻到 UML 这一级，一部分人发现无法理解，又放弃了，只有一部分人最终能探寻到 OO 方法这一级。而在这个过程中，还会存在大量的误读。【2】



我们可以从另一个角度来看：有一些东西本来就是软件开发的基本技术，不管有没有 UML/RUP/Rose…，都是我们平时一直在做的。我们做项目时，写不写需求？（不写；写），用什么方式写？（随意地写；有组织有层次地写），做不做分析设计？（不做；做），用什么方式做？（纸上划拉几下；不同视图模型）。把这些基本技术练好，自然能适应包括 RUP、XP…等各种开发过程。【3】

首先介绍一下笔者建议的“通往职业初段”的软件开发方法，此方法基于 UML，但只专注三个核心要素：用例、类、交互。通过这三个要素的协作，“自外而内，由粗至精”，就能解决软件开发的基本问题。就如金庸小说中的黄药师，只教给傻姑三招，让她练得精熟。开发人员只关注这三样东西，但对每一个细节中蕴含的道理细细体会。每个细节都要精熟，到位，不浪费一分力气。【4】工件的来源都有根据，工件有价值，而不是走过场。



开发方法共分四步，各步之间相互衔接：

1. 使用用例组织需求（需求的技术）

2. 从用例文档抽取类图（分析的技术）

3. 把用例文档和类图结合，通过顺序图分配责任（分析的技术）

4. 和具体平台结合，精化所得到的分析阶段类图和顺序图到合适程度，交给程序员编码（设计的技术）

上面的过程是不完备的。行家肯定能说出：业务建模呢？构件呢？版本控制呢？测试呢？增量迭代呢…？笔者无意美化这个开发过程，再次声明这只是傻姑三招。不过，开发团队要是能锤炼好这几招，就已经提高了一个层次了【5】。

傻姑三招可适合各种 UML 工具（严格地说，白纸和铅笔也算是一种 UML 工具），因为操作很简单。【6】

【1】UMLChina 为开发团队提供 UML/OOAD 培训。特点是：以受训团队的当前项目作为案例进行剖析，培训的过程就是指导团队做自己的项目的过程。

【2】从网上网下的情况看，目前讨论得最多的是用例（用例图）。当然不是因为用例最难理解，而是因为后面的概念还没有来得及讨论。一种需求技术被提出来进行比较详细的讨论，这已经是一种进步。

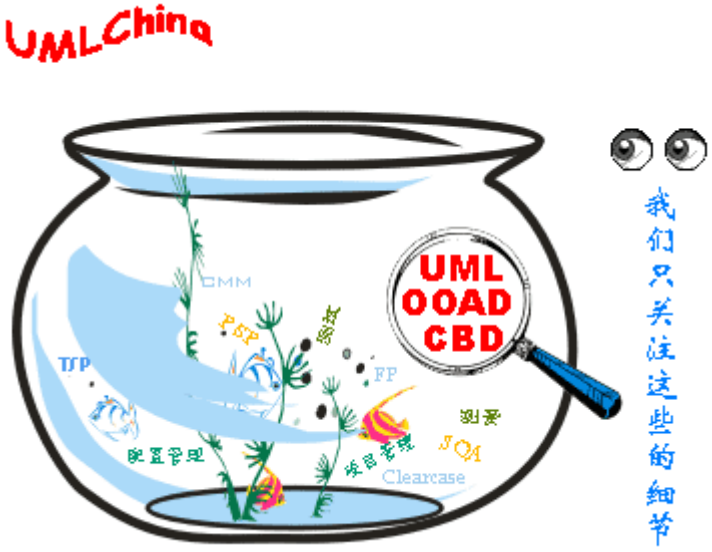
【3】目前成型的开发过程有很多种。重型方法如 RUP 有数十种角色和活动，上百种工件，常会引起“大型团队才适用”的误解。轻型方法如 XP 看起来更吸引人，“简单”（simplicity），但实际上对开发人员的技能要求更高。不曾拥有，何谈放弃？超一流棋手常能“超越定式”，但那是在“理解定式”之后。

【4】职业棋手和业余棋手的区别在于细节。业余棋手靠经验，好野战，但往往不注意细节。业余高段在通晓棋理的职业初段面前，往往不堪一击，因为职业棋手在细节上非常注意，每一步细节都蕴含着棋理。

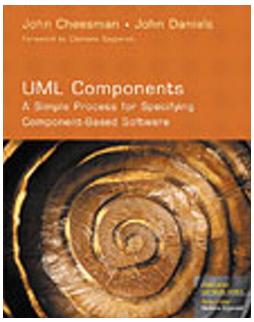
【5】其实不止。其他方面也是可以触类旁通的。例如：用例技术同样可以用来描述业务流程…

【6】目前国内占统治地位的工具是 Rose，但对于特定的项目，还是有很多别的选择的。有关 UML 工具的情况，请看 http://www.objectsbydesign.com/tools/umltools_byCompany.html，不过经过多次购并，现在的形势已经有了很多变化。

下一期将刊登：傻姑之路：通往职业初段—使用用例组织需求（上）



2003 年 1 月 UMLChina 专家交流预告

软件过程 1 月 16 日（星期四）上午 9:30-11:30	 Barry Boehm	嘉宾介绍>>
UML/J2EE/.NET 1/22 日（星期三）上午 10:30-12:30	 John Cheesman	嘉宾介绍>>
软件需求 1 月 28 日（星期二）上午 9:30-11:30	 Ellen Gottesdiener	嘉宾介绍>>

过程敏捷性和软件可用性：以使用为中心的轻量级设计

Larry L. Constantine 著，huang_shen 译

查看评论

关键词：可用性、用户界面设计、以使用为中心的设计、极限编程、轻量级方法、敏捷方法、迭代开发

现在的软件开发有时像一场奇怪的比赛。场地的一边，站着一帮身着各种颜色的破旧运动衫的没规矩的家伙，他们象征许多公司在程序设计时所呈现出的毫无章法的混乱状态。而在场地的另一边，我们看到了排着整齐方队的重量级选手们，每一个大汉的胸前都镶嵌着一个大大的“U”字。现代软件开发的重量级选手正是统一过程（Unified Process）和它的追随者以及统一建模语言（UML）。在近期管理论坛（Management Forum ， Constantine，2000）的一个专栏里，他们成为了被抨击的对象。

在极端自由散漫和迂腐教条之间，占据中场的是一群现代软件开发的热衷者。他们是一小群相对年轻、速度快的选手——一些轻量级的过程，诸如XP、Crystal、SCRUM和FDD。谁都明白这个道理，我们应该遵照教练组的战术安排。2001年2月，这些过程的倡导者们在犹他州的雪鸟城召开会议，对他们的现状与前景进行了讨论。并且一致同意用“敏捷（Agile）”过程来泛指他们的方法，以避免“轻量级”一词给他们带来负面影响。因为，谁都不愿意被别人误以为自己的软件开发方法很柔弱。

敏捷的选手

现在，那些加入到这场竞争中的管理者们不仅要了解占霸主地位RUP（Rational Unified Process），也要熟悉这些敏捷的选手。管理论坛没有对各种敏捷过程作全面地对比；有太多的敏捷过程，而且它们在细节上有很多的不同。但至少Martin Fowler在这方面提供了一个简明的概述（Fowler，2000），请到网页（留意下方的链接）上找相关的详细资料和新的可用版本。在本文中，我将强调一个关于管理的观点，并且我会将重点放在这些敏捷过程的一些共同优缺点上。

典型的敏捷软件开发过程是XP——极限编程Extreme Programming（Beck，2000）的缩写。勿庸置疑，它是敏捷过程中最为人所知的一个，在很多方面都是这个敏捷的典范。正像其它的轻、中、重量级的方法论一样，XP和其它敏捷成员就是一锅技巧、技能的大杂烩，里面包括方法、模型和管理方式，再加上工具、实践和一点点哲学。一些拥护者认为，所有这些都是互补的，你每次必须应用一整套的方法。但许多的从业者并不这样做，他们只挑选那些认为会对自己的工作有用的部分，将他们不喜欢、不相信的部分扔到一旁。

与大多数的敏捷过程一样，XP的重要特点是“注重人多于过程”和“灵活掌握胜过照抄照搬”。敏捷过程强调有效的团队合作，这就需要有良好的管理和领导才能。对于大多数敏捷方法来说，要想成功就要在运用过程的同时，对它进行裁减，将没用的部分去除。

说也奇怪，XP最富争议的方面也是最完善、最实用的方面——结对开发。就像二重唱的形式，它要求两位程序员在同一显示器前工作。由于可以瞬间对所编写的每行代码做检查，这种方法可以大大地减少错误的出现，从而降低了开发的整体成本。由C/C＋＋专家 P.J.Plauger开创的，被大家推崇的结对开发是一项被广为证实的技巧，它有许多成功的案例。

绝大多数敏捷方法都有大量的哲学成分在里头。XP就是围绕着一系列核心价值建立起来的：简单、交流、测试和无畏。没错，是无畏。在这个基础上， Scott Amlber——模型驱动敏捷过程（model-driven agile processes）的倡导者，又加上了谦逊。Jim Highsmith的适应性软件开发（Adaptive Software Development，Highsmith，2000）也被看作轻量级过程之一，与那些已经描述得很清楚的方法，如XP或Pete Coad的特性驱动开发（Feature Driven Development，简称FDD）相比，其中框架性和哲学性的东西要多于限定性的东西。要寻找相关的资料，请点击下方的链接。

敏捷的游戏规则相对简单。缩短版本周期；只做必须的工作，不做任何修饰；不要在分析和设计上浪费时间，马上开始编码；根据小而准确的片段简单地描述问题，然后用连续迭代的方法实现这些片段；通过在构建和测试时得到快速的反馈来开发出一套稳定的系统；由小而简单的工作入手，通过连续的迭代不断深入；程序员与用户之间、程序员与程序员之间保持密切的交流；测试系统的每一个细节，而且保持不断的回归测试。

虽然有一些程序员和他们的经理们把敏捷方法看作散漫的挡箭牌，但必须搞清楚的是真正的规矩必须依据哲学思想和实践来制订。我曾听到有一些管理把XP称作有纪律的散漫，这也许有些道理。

优点

也许敏捷过程的最大卖点就是它们的重量。它们要比那些占主导地位的重量级过程简单得多，这就预示着要学习和掌握的东西相对少一些。但这不意味着敏捷过程是容易的，除了敏捷过程可以代替编程技巧和开发规则这一层意思外，它更重要的意思是解释过程本身的东西相对少一些。

敏捷过程的另一优势是可以减少管理上的开销。不像那些重量级过程，它们更注重的是多种多样的交付产品和大量的过程产物，敏捷过程关心的是代码。设计只在有空和必须的时候才做。索引卡和白板上的草图代替了厚厚的设计文档，简短而又直接的谈话代替了冗长的会议。

早出结果也是敏捷过程的另一特点。缩短版本周期，每一次迭代都创建一套机能健全的系统，敏捷方法使用户在项目的早期就可以用上具备了部分关键功能的产品。

对于用户、管理者和开发者来说，主要能带来收益的是敏捷方法可以减少bug发生率和遗漏率，即出现bug的数量和遗留到下一个阶段的bug数量。更可靠的代码意味着更省时、更省钱的开发和更少的技术支持和维护。

敏捷过程的简单哲理很容易应用到实际工作中去。尽管它们是简单化的，其实也正是因为这一点，敏捷过程可以运用到各种各样的真实环境下的项目中。成功的案例要远远多于失败的（当然还是有的）。只要运用得当，那些受过良好训练的团队往往百战百胜。

存在的问题

敏捷方法有什么不足？它会给我们带来什么样的风险呢？那些有着丰富阅历的读者，应该对三十年前IBM鼓吹的主程序员制团队（chief programmer teams，Baker，1972） 还记忆犹新吧？小而敏捷的团队，由一个拥有雄厚技术背景的人带领。他能够在脑子里构建软件结构，而不需要其他形式的设计。通过细致的检查和测试，对程序不断细化来构建应用。目的是在整个开发过程中都有可运行的代码，通过不断地完善达到满意的规模。

像XP和其它的敏捷方法一样，主程序员制团队赢得了早期的短暂胜利。但最终这种模式还是走向了毁灭，它输在了领导力和规模上。“由有天分、有技术的管理者领导”和“将问题分割成合适的规模来解决”都是正确的方法。但这个世界上能有几个Terry Becker？而且不是所有的问题都能被分为几个部分来在短期增量开发中处理。

今天同样的问题困扰着敏捷方法。毕竟世界上没有几个能像Kent Beck那样领导团队的人。所有敏捷方法都非常依赖人的因素。即使是那些一流的、多才多艺的、纪律严明的并且拥有高超的技能和积极主动的态度的开发者们，也得做到最好才能成功。除了拥有这样的开发者之外，你还得寄希望于他们在被监视的情况下还能拼命干活。

规模是另一个问题。当我与那些经常与轻量级方法打交道的同事交流时，他们都认为敏捷过程对大到一定规模的开发活动并不适合。一个30人的XP团队开始工作时，就已注定了项目失败的命运。虽然Alistair Cockburn宣称他的水晶家族方法（留意下方的链接）已经应用于较为大型的项目，但舆论普遍认为12或15人是大多数敏捷过程的工作上限。作为敏捷方法成功的必要条件之一，紧密的团队合作在超过15或20人的时候变得很难实现。

所有的敏捷方法都是建立在某种形式的增量细化或迭代开发的基础上。这就是说，先写出代码再不断完善其功能，而不是一下子先将整个应用都设计出来。敏捷过程所规定的短周期迭代和它在团队规模上的限制，致使一个敏捷团队不能胜任太大的项目。一个25万行的系统可以做到这样的迭代，但对于百万行的系统就只能望洋兴叹了。再者说，有的项目和应用，要想实现所有功能，就不能简单、准确地分成30、60或90天的增量。

关于使用

在敏捷过程的圈子里，认为被询问者是那些在整个项目过程中，所涉及到的同事和用户。XP和其它轻量级方法对软件的用户端考虑欠周。它们看上去更适合那些不着重于GUI（图形用户界面）的应用。 正像Alistair Cockburn 在给我的email中说的，这“不是弱点，而是欠缺” 。用户界面设计和可用性普遍地被敏捷过程所忽略。除了DSDM和FDD（留意下方的链接）在这方面可能有所不同，用户和用户界面几乎都被忽略了。就像那些重量级过程一样，对用户和用户界面不够重视， 设计往往是很难成功的。尽管如此，值得注意的是持“人的因素”、“交互设计”或“可用性”观点的团体没有被邀请加入敏捷联盟——Agile Alliance（留意下方的链接）。

一些敏捷方法，如XP，明确规定了用户通过联合开发场景——用户素材的方法参与最初需求的敲定。值得注意的是，用户素材一般是由客户和客户代表编写，而不一定是真正的最终用户。任何一个熟悉以使用为中心的设计的人，都理解这一区别的重要性。用户在数量上要比客户多，而且是用户界面设计的主要信息来源。虽然是由客户来决定系统所涉及的范围和功能，但多数情况下他们并不能真正地理解用户需求。此外，用户素材是具体化的场景，就像通常创建用例一样，它倾向于假定用户界面的许多方面已被设计好。

在问题不太复杂、界面不太繁琐的情况下，如果想省一些事的话，可以用迭代原型法代替用户界面设计。然而当用户界面问题非常复杂、可用性又非常重要的时候，就必须运用成熟的、模型驱动方法来设计用户界面。以使用为中心的设计应运而生（Constantine & Lockwood, 1999）。

敏捷而又可用的过程

上世纪90年代初当Lucy Lockwood和我开始开发以使用为中心的设计时，并不准备将它创建成一个轻量级过程。我们的宗旨是通过运用某种工具或技术使我们能用更少的时间设计出更可用的软件。我们只有在必须画图或觉得画图能使开发更快的时候才画图。所以我们也不奇怪为什么后来当提到以使用为中心的设计时，总是与敏捷过程联系起来。

以使用为中心的设计与敏捷过程的区别在于我们注重建模，注重于用模型驱动用户界面的设计、开发。（Scott Ambler的敏捷建模（原名极限建模，留意下方的链接）是一个例外。它是一种注重建模的轻量级方法。）我们在系统中为用户安排的角色是亲自描述任务模型，通过任务模型我们就可以掌握系统任务的本质。任务模型是简单且抽象形式的相互联系的用例集合，它直接驱动了信息和功能在用户界面上如何收集和排序。

就像XP和其它敏捷方法，以使用为中心的设计应用于敏捷方法时，往往是利用卡片来工作。我们在卡片上创建任务实例（task cases）的模型，从而能保持他们的短小精悍和容易收集和分发的特点。我们可以按业务术语的顺序或对最终用户的重要程度来排列卡片的顺序。我们也可以将它们分为相关的几组，这样就可以帮助我们构思完整的、可用的场景，同时也帮助我们决定使用什么样的用户界面——屏幕显示、页面还是对话框。任务实例可以引导用户界面设计。打个比方说，任务实例就好比是谷物通过编程这个大磨房加工成软件的动力——对象和方法。和平常一样，我们关注用户目的和系统职责，使我们能更好地区分用户的真正需要和那些仅仅是理想、希望的区别。总之，虽然没人刻意这么做，但以使用为中心的设计已拥有敏捷过程资格。它弥补了其它方法的不足，为设计出高可用性的用户界面提供了一套有用且高效的方案。

作为一个敏捷过程，要想成功就必须与客户、领域专家以及客户保持紧密的合作与交流，尤其是那些应用领域的最终用户和资深专家。如果用户和专家确实不能参与设计过程，那他们至少也要审查设计的成果。JITR（Just-in-Time-Requirements， Constantine， 2000a）实时需求技术是另一种用于快速敏捷设计的方法。它要求用户或领域专家能随时回答和澄清问题。

从表面上看，当以使用为中心的设计方法应用于极限和敏捷的过程时，都通过简单的以卡片为基础的过程来建模和决策。卡片的运用不仅加速和简化了过程，同时由于卡片的限制迫使我们使用节俭的建模方式。以卡片为基础建模的主要缺点就是，在没有改编之前模型不具备软件兼容性。但这对于倡导“以直接交流和面对面讨论取代文档和图”的敏捷开发来说，并不是问题。当然用卡片记载信息的同时，也不排除用图的可能。

就像在Joint Essential Modeling（Constantine and Lockwood, 1999）中所说的，我们一直认为协作能使项目更快更好地完成。一个拥有设计者、开发者以及用户或相当于用户的人（如，领域专家）的团队，结合“以使用为中心”的思想，执行下面列出的敏捷活动就能做到最好：

1. 通过头脑风暴直接在卡片上构建出用户角色的清单。

2. 回顾和修整卡片上的清单，然后简短地描述出卡片上每一个角色最显著的特点。

3. 按着根据对项目成败的影响程度排列用户角色卡片。考虑推迟实现优先级低且不会引起争议的用户角色。

4. 由优先级高的用户角色开始，通过头脑风暴在卡片上构建出支持这些用户角色的任务实例（提炼后的用例）的清单。

5. 将任务实例卡片排序。先是按预期的发生频率和普遍性的顺序，然后是按照对项目成败的影响程度。

6. 将任务实例卡片分成三份：必需的（在这个版本中首要完成的），期望的（在这个版本中，有时间才做的），暂时不做的，并在卡片上做好标记。

7. 对于所有必需的任务实例和重要的、复杂的、不清晰的、感兴趣的那些期望的任务实例，在它们的卡片上写出描述性的文字。这段叙述文应该用简单、抽象、非技术性的形式记述所有的“正常情况”（事件的常规流），而且要采取两栏式，一边记述用户目的，一边记述系统职责。其它的一些附注性质的或描述“异常”实例的文字应写在每张卡片的背面。

8. 根据任务实例的相似性和相互依赖性，将所有卡片分成几组。

9. 将每组卡片都看作是用户界面中的一套由相互呼应上下文所支撑的任务。这之后就可以画出这部分用户界面的书面原型草图。我们应把注意力集中在必须的任务实例上，而不是其它的。（所有的任务实例都应看作是整个用户界面构建过程能顺利进行的保证。）

10. 和用户和客户一起用任务实例所产生的场景来检查原型。

11. 根据检查的结果修订和改进书面原型。

12. 根据书面原型，再结合任务实例，就可以开始用户界面或表示层的编程工作了。

在一个采用以使用为中心的敏捷设计方法的开发过程中，那些不依赖于用户界面设计的内部组件、后端组件的编程工作可以与以上的过程同时进行。这样，当任务实例确定的时候，那些独立于用户界面设计的编程就能有一定的结果了。当书面原型修订完，无论是使用极限方法还是“普通方法”，原型中包含的上下文以及原型所相关的任务实例就成了以后编程工作的依据了。

几乎所有的敏捷方法都认定开发是通过连续的版本循环来进行的。连续的迭代利用角色和任务实例来引导用户界面的改进和扩展。在第一次迭代前，通过书面原型的方法考虑这些角色和任务实例，将减少整个构建过程中用户界面的更改。

中场争夺

在浑浑噩噩的编程者与患了强迫症的文档狂之间是一块狭窄的中场腹地，这正是那些急得团团转的管理者、超负荷工作的程序员和得不到满足的用户的希望所在。出于这种考虑，我在软件开发管理论坛我的最后一期专栏中给读者写下了这样的忠告：如果你打算远离那些软件开发中的暴民和官僚的统治，那么就加入敏捷的行列吧。请记住奥卡姆的威廉说的话，“不应无必要地增加实体”。

参考文献

Baker, F. T. 1972. “Chief programmer team management of production programming,” *IBM Systems Journal, 11* (1).

Beck, K. 2000. *Extreme Programming Explained,* Reading, MA: Addison-Wesley.

Constantine, L. L. 1992. “The benefits of visibility,” *Computer Language Magazine,* September. Reprinted in *The Peopleware Papers,* Upper Saddle River, NJ: Prentice Hall, 2001.

Constantine, L. L. 2000a. “Cutting Corners: Shortcuts in Model-Driven Web Development,” *Software Development, 8,* (2), February. Reprinted in L. Constantine, ed. *Beyond Chaos: The Expert Edge in Managing Software Development.* Boston: Addison-Wesley, 2001.

Constantine, L. L. 2000b. “Unified Hegemony,” *Software Development, 8,* (11), November.Reprinted in L. Constantine, ed. *Beyond Chaos: The Expert Edge in Managing Software Development.* Boston: Addison-Wesley, 2001.

Constantine, L. L. and Lockwood, L. A. D. L. 1999. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,* Reading, MA: Addison-Wesley.

Fowler, M. 2000. "Put Your Process on a Diet," *Software Development, v. 8,* 12, December.

Jeffries, R. 2001. "Card magic for Managers," *Software Development, 8,* (12), December.Reprinted in L. Constantine, ed. *Beyond Chaos: The Expert Edge in Managing Software Development.* Boston: Addison-Wesley, 2001.

相关链接

想进一步了解以使用为中心的设计，以及与敏捷/轻量级方法相关的资讯，请登录http://www.forUse.com.

Agile Alliance

http://www.agilealliance.org/

Crystal

Alistair Cockburn: http://crystalmethodologies.org/

DSDM

http://www.dsdm.org/default1.asp

dX

Robert Martin: http://www.objectmentor.com/publications/RUPvsXP.pdf

Extreme Modeling

Scott Ambler, Modeling: http://www.extreme -modeling.com/

FDD

Peter Coad: http://www.togethercommunity.com/coad-letter/Coad-Letter-0070.html

Fowler, Agile methods overview: http://www.martinfowler.com/articles/newMethodology.html

SCRUM

Ken Schwaber: http://www.controlchaos.com/

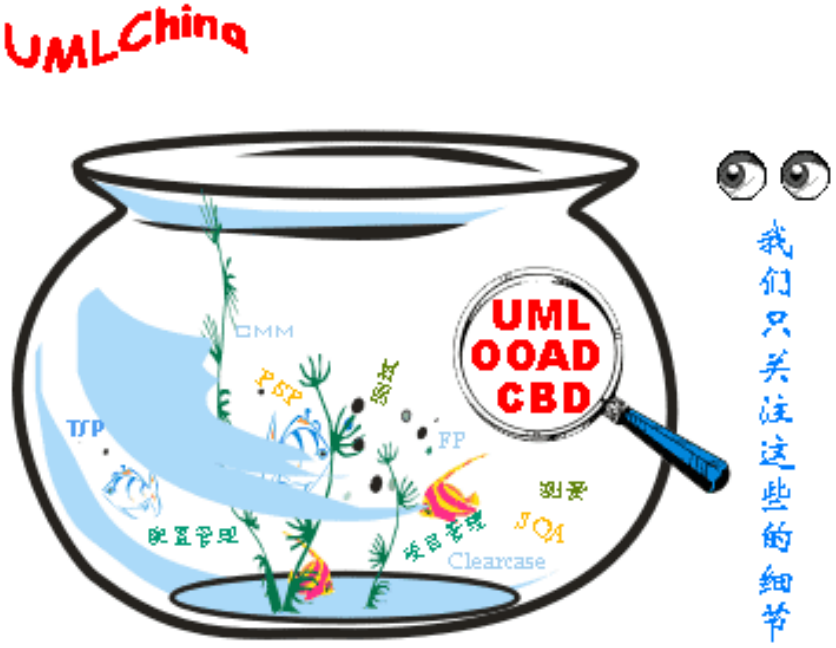
Jeff Sutherland: http://jeffsutherland.com/scrum/

XP

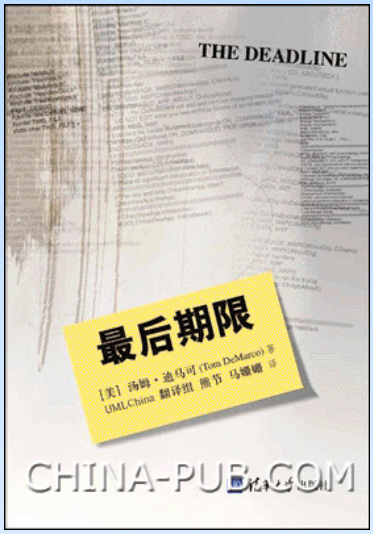
Jim Highsmith, J. “Extreme Programming.” http://www.cutter.com/ead/ead0002.html

Ron Jeffries. http://www.xprogramming.com/

Don Wells: http://www.extremeprogramming.org/



《最后期限》



Tom Demarco

翻译：UMLChina 翻译组 熊节

汤普金斯在飞机的座位上翻了一个身，把她的毛衣抓到脸上，贪婪地呼吸着它散发出的淡淡芬芳。文案，他对自己说。他试图回忆当他这样说时卡布福斯的表情。当时他惊讶得下巴都快掉下来了。是的，的确如此。文案……吃惊的卡布福斯……房间里的叹息声……汤普金斯大步走出教室……莱克莎重复那个词……汤普金斯重复那个词……两人微张的嘴唇触到了一起。再次重播。"文案。"他说道，转身，看着莱克莎，她微张的嘴唇，他……倒带，再次重播……

....

"我不想兜圈子，"汤普金斯看着面前的简报说，"实际上你们有一千五百名资格相当老的软件工程师。"

莱克莎点点头："这是最近的数字。他们都会在你的手下工作。"

"而且据你所说，他们都很优秀。"

"他们都通过了摩罗维亚软件工程学院的 CMM 2 级以上的认证。"

Web 应用模型中的抽象和复用机制

Gustavo Rossi 著，ydr 译

查看评论

摘要

我们在本文分析了各种抽象和复用机制，这些机制用于WEB应用，来改善其升级和维护工作。我们首先复习用于定义WEB应用模型的面向对象超媒体设计方法(OOHDM)，特别是导航模型和概念模型的分离。然后专门研究在这两个模型中的抽象和复合机制，以说明如何结合OOHDM观点和节点聚合概念。我们介绍了导航和接口模式，并说明了模式产生Web设计框架的方法。我们强烈主张在目前的WEB应用技术状况下，建设相似应用的系列模式，以改进设计复用。其次，我们展示了描述Web框架的符号，给出了在电子商务领域的一些例子。最后讨论了进一步的研究工作。

1. 介绍

建设复杂的WEB应用是一项耗时的任务，因为它们必须提供对重要信息资源的导航访问，不仅允许用户全面浏览潜在的信息世界，也允许对信息进行操作。在某些领域如电子商务中，客户的行为触发了那些必须要与核心业务软件集成的复杂工作流。这种集成也必须以另外一种方式进行；例如，电子商店中的营销软件应该监视顾客的行为，以便更有效地为顾客导航。第一个明显结果是，我们不仅必须仔细设计导航体系结构，而且要有效地与业务模型集成。

对于复杂事务，Web应用的开发应该是零缺陷的，而且不要花费部署和维护的时间。在这种上下文中，我们不仅应该使用系统化的工程技术，而且也能在整个开发周期内改善复用。

获得可复用设计或组件的关键是能以抽象方式表达可变性；即，我们需要改善我们的建模技术和实践，去建设可扩展的和可复用的概念模型。然而，在为传统应用广泛探讨复用技术时[Meyer94]，Web应用的本质似乎妨碍了设计者处理设计和实施复用。

我们利用OOHDM探索WEB应用中的抽象、组合和复用技术已经多年[Schwabe98]。OOHDM把Web应用当作对象模型的导航视图[Rossi99b]，并为导航（上下文、索引等）和用户接口设计提供了一些基本结构。我们研究在开发过程中最大化复用的各种方法，因为我们，还有其他人注意到，在一些相似应用域的解决方案中有很大程度的共性。

本文的目的是描述用于构建WEB应用模型的各种设计复用机制。我们用简单的例子说明描述每一种机制，并与在（面向对象的）概念建模中的现有技术进行比较。我们强调新机制（如导航模式WEB框架）和那些特别用于Web应用的新机制（上下文、索引等）。

虽然我们利用OOHDM作为基础设计方法，本文中的这种思想很容易用于其它建模方法；读者可以在[Schwabe98, Rossi99b, OOHDM00]中找到对OOHDM的详细描述。

本文的组织结构如下：在第2部分我们把Web应用模型刻画为概念模型和导航模型的结合体。在第3部分说明了OOHDM中的各种抽象和组合机制如何协同工作取得上好的和可复用的设计模型。在第4部分我们复习导航模式，从而简要强调了抽象设计复用。因为模式产生体系结构，我们在第5部分进一步把Web设计框架描述成获得整个域模型复用的一种方法。在第6部分我们描述了OOHDM框架，用于说明Web设计框架的符号方法。最后讨论了一些深层次研究工作。

2. Web 应用模型：概念+导航模型

OOHDM的关键概念是，Web应用模型包括概念模型和导航模型[Rossi99b]。这个概念模型旨在利用著名的面向对象原型（如类和关系）和抽象机制（如聚合和泛化/特化）去捕捉这个领域的语义。例如，在电子商店中，这个概念模型将包含核心类如产品、定单和顾客等，及其相应行为。我们用UML[UML97]作为说明这个概念模型的符号。因为这个概念模型是面向对象模型，我们可以利用面向对象[Johnson88]中的复用方法。

在OOHDM方法中，用户不导航概念对象而导航对象（节点）。节点被定义成概念对象的视图，利用与OODB视图定义方法[Wim90]相似的一种语言。节点用连接补充，而连接本身被作为概念关系的视图。这个导航模式说明了节点和那些包含这个应用的导航结构的连接类。

对于每个特定的用户配置，我们构建一个导航模型作为这个共享概念模型的视图。这样，我们在相似的应用系列中就可以复用这个概念模型。而且，正如在第3部分所说明的，我们能在一个应用的上下文中定义各种视图。

在图1中，我们显示了一个电子商店的部分概念设计。注意在这个模型中的一些类将被映射到导航模型中（即它们将被作为节点研究），而其它的类如定单则不是这样。

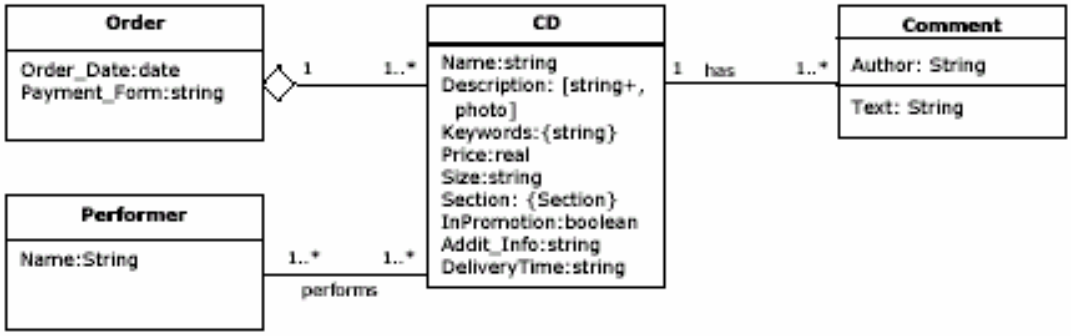
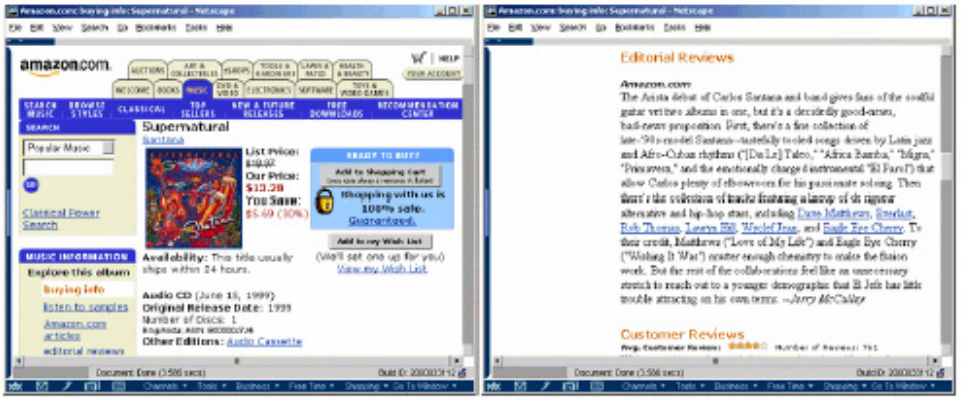


图 1: CD商店的概念设计

如果我们设计这个电子商店的顾客视图，我们将指明产品的节点类。如图2所示，这些节点可能结合概念类“CD”的属性和概念类“注释”和“表演者”的属性。注意在优秀的面向对象软件说明（如图1中的）中，“产品”、“注释”和“表演者”分别属于不同类，而节点实施概念类的视图（遵循观察者设计模式[Gamma95]）。定义视图的精确句法可在[Rossi99]中找到。



Node CD FROM CD:C
name: String
description: Photo
price: Number
performer: String SELECT name FROM Performer: P WHERE C *tsPerformed by* P
comments: Array[Text] SELECT text FROM Comment: R WHERE C *hasComment* R
....
other attributes and anchors

Figure 2: CDs including comments in Amazon.com and the OOHDM definition

图2：Amazon.com网站中的包含注释的CD及其OOHDM定义

这个导航模式在OOHDM中用应用中的显示导航上下文的上下文模式和访问结构（索引）补充。导航上下文是一套通常被连续探索的对象；例如，一个作者的著作，某摇滚乐队的CD等。在图3中，我们显示了这个电子商店的部分上下文模式。图3中的符号以紧凑的方式显示了用户将探索哪套对象和它们如何相关联。导航上下文是一个新颖的设计原型，用于简明指出集合，特别是为探索多维空间而开发的。

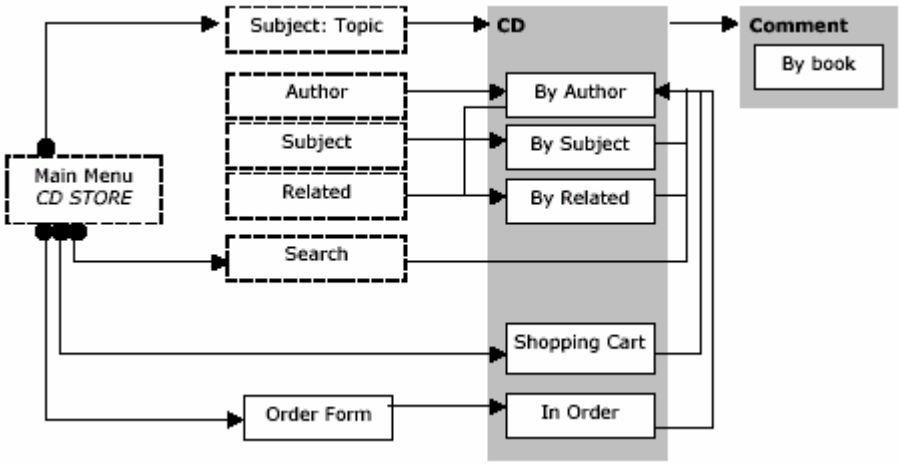


图3：电子商店的上下文模式

图3中虚线框显示了访问结构（索引），而类CD中的方框（和注释）说明了可能的上下文，在这些上下文中CD（和对应注释）可以被访问。一个节点可能出现在各种上下文中，根据它可以达到的上下文显示不同信息。在这种情形下，我们利用修饰器（Decorators [Gamma95]）减弱这个节点中的基本信息和节点展现的不同“外观”之间的耦合。因此，导航上下文结合两个导航模式、基于集合的导航和上下文中的节点[Rossi99a]。

在相同领域的应用系列中复用应用模型时，这种导航模式和上下文模式起着关键作用。我们将在第6部分讨论这种复用技术。

3. 结合视图与聚合节点

复杂的Web应用提供了获得它们所包含的信息的多种方法。例如，在电子商务应用中，顾客获得各种建议如热点列表、推荐和最新发布等。在图4中，我们显示了一个包含电子商店中产品的各种链接的主页例子。

图 4:描述主页的一个节点

在OOHDM中，我们可以用聚合节点详细说明这个主页。一个聚合允许在相同节点中粘合各种信息项（其它节点）和访问结构（如索引）。在图4中主页的节点部分的说明如下：

Node MusicHome

news: Array [CDView]
search: SearchTool
categories: IndexOfCategories
topSellers: IndexOfTop
landmarks: IndexOfStores
...
other attributes

Node CDView FROM CD: C
name: String
performer: String SELECT name FROM Performer: P WHERE C *tsPerformed by* P
description: Photo
shortComment: Text

注意上述类型CDView的说明取益于视图机制，它能在网站的其它部分被复用（例如“艺术家精华”部分利用了与每个CD相似的概述）。聚合允许以随机方式说明组合节点（正如通常情况它在主页中一样）。然而，聚合节点和视图机制以比面向对象中的简单组合机制复杂的方式协作。这种协作用链接机制补充，而链接机制允许同一对象的不同视图相互连接。例如，你可以轻易地从图4中的CD汇总导航到相应CD中。在图5中，我们在图表中说明了如何复用一个对象的视图和这个视图如何被链接到同一对象的其它视图上。

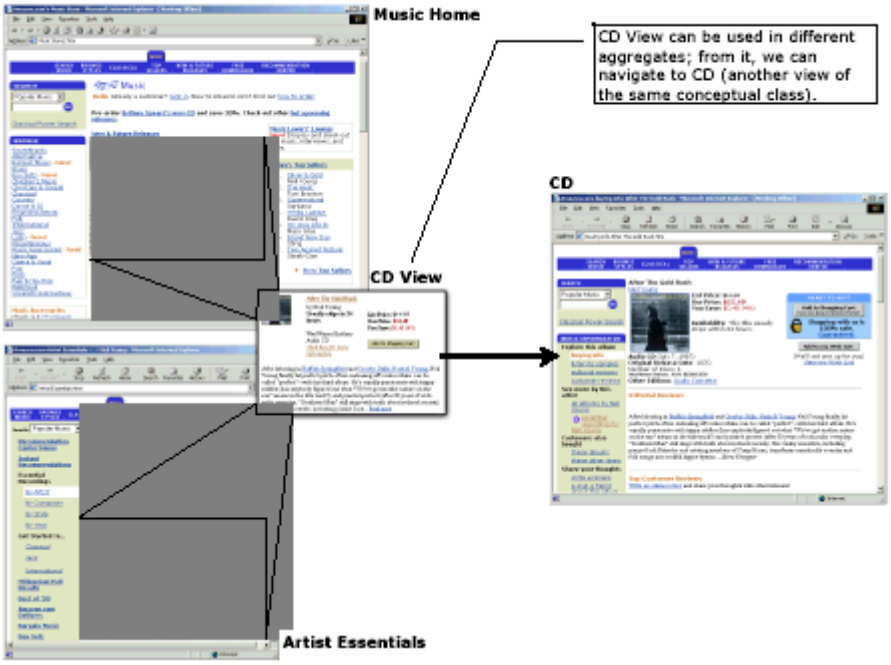


图 5：导航模式中的聚合和视图复用

这个简单例子引起了一些与设计复用有关的有趣论点和疑问：

1. 我们能归纳前面显示的主页背后的基本思想吗？在构建这种聚合节点时，我们解决了什么设计问题？
2. 这种应用的结构与相同领域中的其它应用相似吗？换言之，在设计相似应用的概念模型和导航模型时，我们能取益于我们的智力投资吗？

这些疑问显示了一些非平凡的设计复用问题。当组合、视图和继承允许改善一个应用中的复用和维护时，它们并不足以表达在应用系列中的复用方面。下面我们介绍在Web应用中的设计复用的两种新方法：导航模式和Web设计框架。

4. 用导航模式设计复用

模式思想原来是由Christopher Alexander在城市建设体系结构领域提出的[Alexander77]，几年前被采纳到面向对象的软件中[Gamma95]。依靠以抽象方法表示重复发生的问题和已经证明的方案，模式记录了设计经验。它们是捕获、表达和复用设计经验的奇妙工具。

模式用表示非平凡设计结构的术语丰富了设计词汇，模式改善了设计者之间的通信。著名的解决方案通过模式表达，新设计者就可以受益于专家的知识。

我们为Web应用挖掘了模式并利用一个与Alexander的相似的模板[Rossi99a]证明它们。事实上，超媒体和Web模式与原来的城市模式相似，因为它们都表示用于构建有用的导航空间的重复结构；它们显示了有助于用户通过超空间找到线路的设计方案。超媒体社区提出了十二种新模式[Garzotto99]，现在它正在从事一项工程以在一个共享目录中表示这些可复用的方案。[HypPatterns99].

继续用前面例子，我们为处理（部分）应用复杂性定义两个简单而有效的模式：门户和界标。我们简要地描述它们，阐述它们要解决的问题和（广泛使用的）方案。

4.1 门户

问题：

在许多Web应用中，特别在电子商务中，我们想给用户一个他将在包括新闻、建议和商机的网站找到什么的综合描述。如果我们遵循一个纯正的多媒体设计观点，这个“主”页应该影射一些概念对象，或只是服务和产品的索引。这种方法从设计观点看或许是正确的，但是根本不可行。用户必须深入层次索引去寻找他们需要的内容。

方案：

把主页设计成各种信息项、锚和访问结构的聚合体。把空间用于新闻，把暗示面向用户、通用索引和特殊提议等。这种主页甚至包含一些在语义上不相关的信息。门户是一种机会主义的设计方案，它允许增加网站的访问数量，因为他们容易找到他们想要的信息。门户广泛用于所有的商务网站，如amazon.com、 netgrocer.com，以及更一般的网站如netscape.com。门户概括了图4中的设计方案。

4.2 界标

问题：

许多Web应用包含子网点，子网点提供特殊功能（不同商店、搜索功能等）。当我们描述导航模式时，我们尽力紧紧跟随存在于目前的对象模型中的关系，例如我们能从作者导航到书、从CD导航到它包含的歌曲清单。我们能从一本书导航到以前读者所作的评论，阅读相关图书等。然而，我们想要读者在任何时刻跳转到音乐或书（下级）或他的购物车。如果我们构建这种导航模式，链接每个导航类（如书、注释、新闻歌曲等）到商店、书店和购物车，我们的结局如同意大利面条，难于理解。

方案：

定义一套界标并使之能从网络的每个节点访问。使对界标的链接接口看起来一致。用这种方法，用户将有一个有关

界标的协调的可视线索。根据我们正在访问的站点，我们可以有不同级别的界标。注意对界标的锚和链接不容易从概念模型中得到，因为它们不表示概念关系。界标不同于索引，因它们出现在应用的每个节点上。这种模式广泛用于Web应用中，以指示相关子站点和功能。

挖掘和归档模式是一项有益的工作。它们或许是一般的，如同前面显示的那些，或许是特指某个领域（见一些电子商务模式中的一些例子[Rossi00]）。

模式并不独立，它们被集成到开发方法中生效。它们必须结合起来，以产生更高级的抽象，允许表示富有的可复用体系结构。在OOHDM上下文中，我们为在上下文[Rossi99b]和界标[Rossi99a]中的一些导航模式如基于集合的导航和节点定义了符号。在图6我们通过显示一个集成界标的导航模型概括以前的例子。注意，我们不用混乱的图示而用一个简化的图示，其中对界标的链接被省略。图6中的CD商店、书店和玩具店是界标（用箭头显示，圆点表示来源）。注意，在CD商店中，“主题”、“搜索”“购物车”和“定单”都是二级界标。

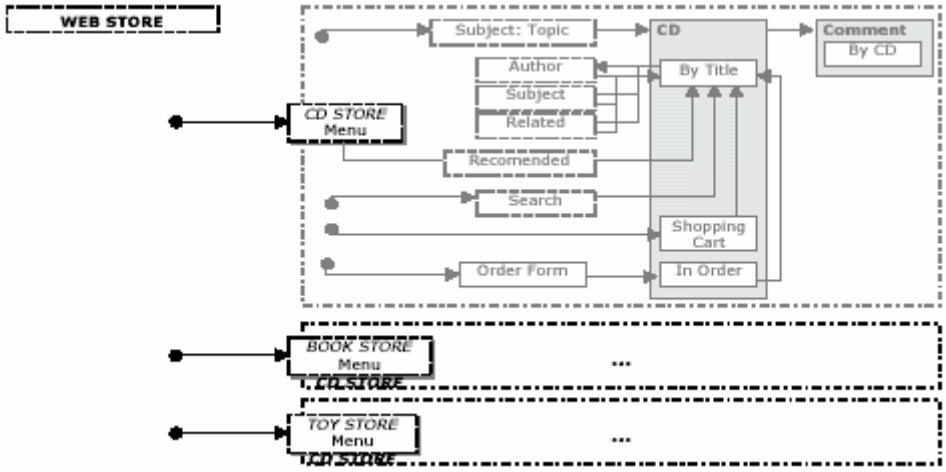


图6：在导航模式中使用界标

把这些模式合并成设计库有助于减少图表的复杂性，这样使得复用更加可行。然而，当我们设计复杂应用时，我们需要更有力的复用方法，以便依靠结合或特化我们在其它应用中用过的组件来构建新应用。

例如，在电子商务领域，我们可以容易地发现大多数虚拟商店给顾客提供相似的服务：多数允许依靠搜索或层次导航找出产品，都提供购物车保存选择结果等。而且，我们甚至能在核心应用行为上找到共性，例如，在顾客做出结帐操作时被触发的行为集合在本质上相同：检验用户数据、产生定单、发送确认邮件和在发货后再发邮件等。我们应该可以定义能抽象这些共性的和能平滑扩展到应付各个特殊应用中的变体的体系结构。我们随后介绍Web设计框架和说明它们如何与导航模式相关联。

5. 从 Web 模式到 Web 框架

虽然模式（特别是导航模式）允许在应用中记录和复用微体系结构，但是它们不足于表示较大的可复用设计结构。然而，它在面向对象的社区中是著名的，结合模式和其它设计原型是创建面向对象框架的有效技术[Johnson94]。

框架是针对特定领域中的应用系列的可复用设计。它们作为一套应用的骨架，可以被应用开发者定制。当在相同的领域中构建许多不同的应用时，应用框架为支持他们的共性提供“模板”，并适应个体异性（差异）。当模式提供设计经验的抽象复用时，框架允许在一个领域中复用具体的设计[Fayad99]。

框架由一些抽象和具体的类组成，这些类包含了特定域中的通用行为（通常用某种编程语言说明）规范。设计框架的关键是识别它的热点（即：在框架中变化出现的位置）。热点或许是抽象的类、钩或模板方法等。

接着上面的例子，我们可以概括（图4中的）概念模型，来表现虚拟商店中的抽象类和协作。这个模型应该包括抽象类产品、各种定单、评论等。开发某个商店的设计者需要定义新的具体类（如产品的子类）和特化某些行为如定单处理来适应特定应用（如利用不同的业务规则销售其它产品）。在虚拟商店中（如Amazon.com），这种方法用于定义新的子商店、各种购物和付款策略。

设计框架是一项困难而有价值的工作。我们需要理解领域和开发通用设计以被实例化成不同的应用。若已知某个领域的框架，我们能得到：

- 一个可复用的领域模型，框架包含业务实体、行为和规则。
- 对这个领域的应用的可复用设计，框架将包含有助于解决领域中的某个问题的设计抽象和
- 可复用的类和对象，它们的代码能用模板或钩方法定制[Gamma95]。

要把这种方法应用于Web模型，我们需要考虑各种变异：与领域模型相关的（即各种付款策略）和与导航体系结构（即各种索引、上下文等）相关的变体。除此之外，如果大量结合那些经常在Web应用开发和实施中的语言和工具，以编程语言为中心的方法（在应用框架上相同）难以用于Web。

Web设计框架可以被映射到将被实例化成运行的应用的应用框架或能被实例化成为改善Web应用模型的“纯”OOHDM模型、然后实现成一个Web应用[Schwabe00]。下面我们提出一种标记符，以改善带有Web设计框架中需要的抽象的Web应用模型。

6. OOHDM 框架：Web 框架的标记符

为了详细说明Web设计框架，我们定义了一种新符号，叫做OOHDM框架，它平滑地延伸了OOHDM。本文的目的不是罗列OOHDM框架的语法，而是分析如何改进概念建模中的抽象和组合机制，以表达通用的Web功能。我们将简要描述这种符号以强调每种建模特性。如前所说，OOHDM框架中的框架模型规范包括通用概念和导航的模型规范。我们下一步对每一个进行分析，并指出新的抽象机制。

6.1 概念模型中的抽象和一般性

Web应用中的可变性体现在概念模型上。在图7中，我们显示了电子商店的部分通用模型。注意我们包括了电子商店的部分抽象模型。我们也包括了一些抽象类如产品、特殊评论和付款方法。

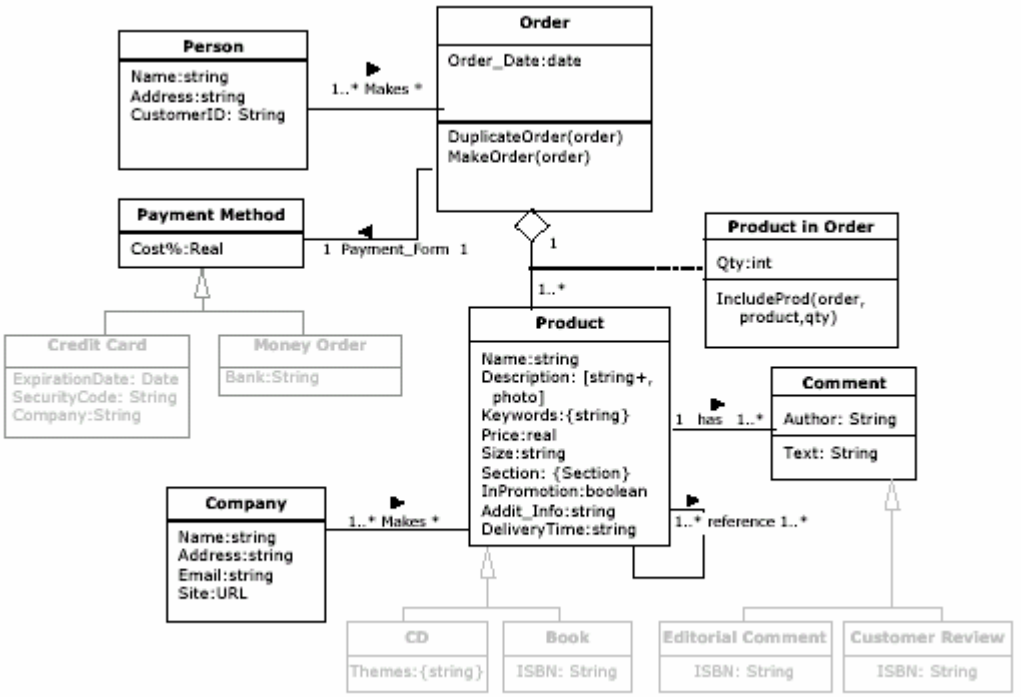


图7：虚拟商店的一个通用概念模型

模型对象模型中的通用性已经在面向对象的社区中进行了广泛讨论，可以用现有的符号表达通用类和行为[Pree94]，因此，我们不对它做进一步讨论。有趣的是，使用视图机制，概念模型中的抽象规范可以被映射到导航模式中的抽象规范。应该指出，当通用概念模型被特殊化为某个应用时，有必要采纳导航模型以与这种变化一致(参见 6.2)。

6.2 特化通用导航模型

OOHDM框架中的通用导航模型有通用导航模式、通用上下文图示和一套映射和实例化规则组成。通用导航模式概括了视图（或按[Gamma95]中的观察）理念，它与导航模式相似，除了节点属性（标记为“*”）和关系（画成虚线）是可选的，如图8所示。一个可选属性（对应链接）或者在或者不在一个实例化的应用中出现。注意，因为导航模型将经常被映射到一个非面向对象的实施中，我们并不限于“纯”符号，即我们总能用定义合适类层次指定可选的特征（属性或链接），虽然是以一种并不简练的方法。为简练起见，我们不包括图8中的子类。

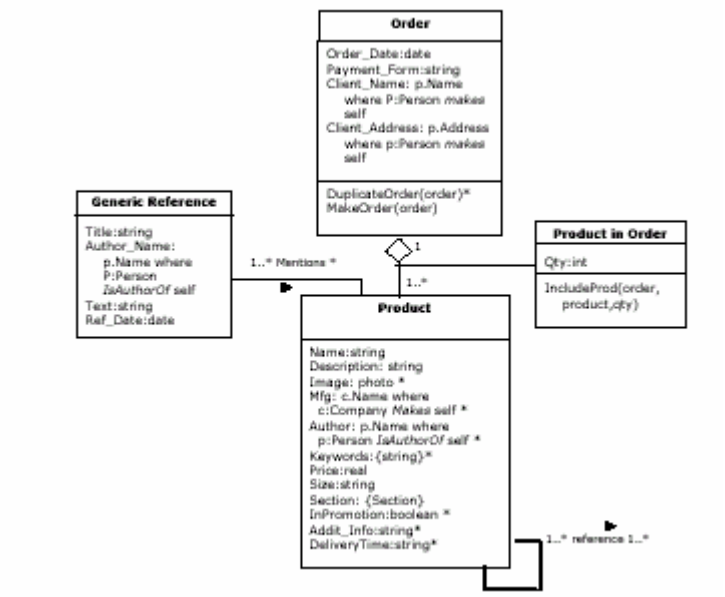


图8：通用导航模式中的可选属性和链接

通用导航模式中的子类允许一个获得通用性的更微妙的方法。在上述例子中，我们能创建产品子类，增加属性或锚，我们甚至需要为一个特定的属性说明视图规范，如下所述。

例如，假定我们有两个“评论”的子类（如图7中的通用概念模式所示），如果我们想把这个商店特化到图书和CD商店（在为虚拟商店的框架上下文中），我们需要一些导航产品子类显示只来自于一个（概念）子类型的评论。因此，我们显示了抽象节点类产品的部分规范，和如何为图书指定属性“评论”的定义。这个精化的操作符查询对应的超类和用子类的编辑评论代替评论。这样，我们在指出：图书只显示编辑评论。

```
Node Product from Product: P
...
comments: Array[Text] SELECT text FROM Comment: R WHERE P hasComment R
...
```

```
Node Book from Book:B
...
REFINE comments WITH EditorialComments
...
```

同时，通用上下文图示表示了Web应用中另一种热点，以一种抽象方法显示哪个上下文和访问结构在某个领域中出现。注意因为导航上下文是节点的集合，定义通用上下文相当于说明了通用集合。因此，用通常的面向对象的抽象机制，在上下文图示中获得一般性并非直接，即虽然上下文和索引最终会被映射成类，解释它们的可变性或许需要利用复杂图示。而且，我们更喜欢概括上下文图示和用一个通用上下文规范卡来补充，后者为实施指出了可能的限制。

在图9中，我们显示了一个针对我们的虚拟商店框架的简化通用上下文模式。虚线框和圆框指示通用访问结构和上下文。例如，通用上下文“按性质的产品”是一个简单的类获取上下文，典型地被实例化成一个或多个允许在产品中导航的上下文，根据某些性质（即“按价格的产品”、“按作者的产品”、“按颜色的产品”等）。一旦在这些之内，就可能导航到其它“相关产品”（即附件、配件等）。有几个访问结构，引导读者到这些上下文；典型地，有集中层次结构反映了真实商店中的产品部分（部门）。注意，我们已经说明了一些界标（像购物卡、定单和搜索）。

浏览产品的第二种方法是在随机获得的任何分组之内。通常，这些将响应一些（通常是著名的）人的（出版物）建议，或一些指导，如“纽约时报的畅销书清单”。这种分组通过通用上下文“按引用的产品”。

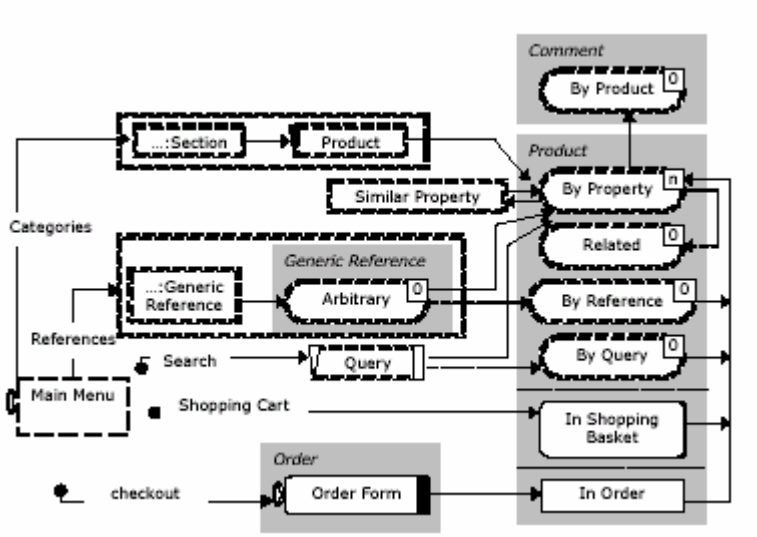


图9：虚拟商店的通用上下文模式

注意在图3中的上下文图示是图9中的通用图示的实例，在图9中的通用上下文“按性质的产品”被具体化成“按主题的CD”和“按作者的CD”。通用上下文模式简要地显示在Web应用中为某个领域提供基于集合的导航的不同方法。当用通用概念的和导航模式进行补充时，它们为在领域中的Web应用系列提供这种模型。这样，我们能得到结合通用导航模式（如集合和界标）和抽象和可复用的Web应用模型领域的规范。

7. 结论和将来的工作

我们已经在本文讨论了在Web环境中的各种抽象和复用机制。我们说明了最简单的技术如组合和继承为设计者提供了微妙的结合，在处理非平凡导航模型时。特别，OOHDM视图语言能用于与聚合体（子类），产生简练而可复用的导航设计。我们通过简要分析导航模式讨论了设计经验的复用。虽然设计模式在精细粒度上提供了设计经验复用，我们说明了如何结合它们去获得更大的可复用模型。我们介绍了Web设计框架，解释了通用的、可复用的概念和导航模型如何用OOHDM框架符号描述。Web设计框架显示了模式的结合（如基于集合的导航、界标和观察者）如何能为一个特定领域的应用系列产生通用设计。

Web应用框架难于设计，因为它们需要对应用领域的彻底研究；设计资料（概念和导航级）要用各种抽象和组合机制描述。概念和导航模式应该在被扩展到的领域的上下文中可复用。我们用扩展特殊到上下文的理念研究复用上下文图示的方法。

虽然本文的重点在于设计，但值得强调：以前描述的所有原型和机制，也能用目前的Web技术实现[Schwabe00]；而且，把设计框架映射成“纯”面向对象也是直观的。我们挖掘某个领域如电子商务的Web模式。研究一些方法来用这些新模式来丰富框架设计符号。仍然要研究几种实现外观，如在Web应用中实现视图和上下文的有效方法。

我们相信对Web应用日益增长的兴趣需要各种方法，以便容易地构建可扩展和可复用的概念模型。Web应用表现了全新的特性，需要认真考虑并把著名的抽象和组合机制应用到这个新领域。本文中的这种理念或许可以作为研究Web模型中的抽象和复用技术的背景知识。

8. 参考文献

[Alexander77] B. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, “A Pattern Language,” Oxford University Press, New York 1977.

[Fayad99] M. Fayad, D. Schmidt and R. Johnson (editors): “Building Application Frameworks”, Wiley 1999.

[Gamma95] E. Gamma, R. Helm, R. Johnson and J. Vlissides: "Design Patterns. Elements of reusable object-oriented software". Addison Wesley, 1995.

[Garzotto99] F. Garzotto, P. Paolini, D. Bolchini and S. Valenti: “Modelling by patterns of Web applications”. Proceedings of WWWCM99, Lectures Notes in Computer Science.

[HypPatterns99] Hypermedia Patterns repository: http://www.designpattern.lu.unisi.ch

[Johnson88] R. Johnson and B. Foote: “Designing reusable classes”. Journal of objectoriented programming” 1(2), 22-35, 1988.

[Johnson94] R. Johnson and K. Beck. “Patterns generate architecture”. In Proceedings of the European Conference on Object-Oriented Technology (ECOOP94).

[Kim90] W. Kim, "Advanced Database systems", ACM Press, 1994.

[Meyer94] Bertrand Meyer, “Reusable Software” - The base object-oriented component libraries. Prentice Hall 1994.

[OOHDM00] Daniel Schwabe and Patricia Vilain: “The OOHDM notation”, available at http://sol.info.unlp.edu.ar/notacaoOOHDM/

[Pree94] W. Pree: “Design Patterns for object-oriented software”, Addison Wesley, 1994.

[Rossi99a] G. Rossi, F. Lyardet and D. Schwabe: “Patterns for designing navigable spaces”To appear in Pattern Languages of Programs 4, Addison Wesley, 1999.

[Rossi99b] G. Rossi, D. Schwabe, F. Lyardet: “Web application models are more than conceptual models”. Proceedings of the First International Workshop on Conceptual Modeling and the WWW, Paris, France, November 1999.

[Rossi00] G. Rossi, D. Schwabe, F. Lyardet: “Patterns for E-commerce applications”. Submitted to EuroPLoP 2000, available at ...

[Schwabe98] D. Schwabe, G. Rossi: “An object-oriented approach to web-based application design”. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp.207-225, October, 1998.

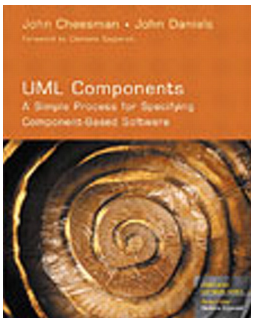
[Schwabe00] D. Schwabe, G. Rossi, L. Emeraldo, F. Lyardet: “Web Design Frameworks: An approach to improve reuse in Web applications. Proceedings of the WWW9 Web Engineering Workshop, Springer Verlag LNCS, forthcoming.

[UML97] UML Document Set. Version 1.013 January, 1997, Rational, 1997. (available at http://www.rational.com/uml/references/index.html)

征稿

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

2003 年 1 月 UMLChina 专家交流预告

软件过程 1 月 16 日（星期四）上午 9:30-11:30	 Barry Boehm	嘉宾介绍>>
UML/J2EE/.NET 1/22 日（星期三）上午 10:30-12:30	 John Cheesman	嘉宾介绍>>
软件需求 1 月 28 日（星期二）上午 9:30-11:30	 Ellen Gottesdiener	嘉宾介绍>>

GUI 设计精髓：交互

Weinschenk 著，Ma Yaohua 译

查看评论

交互

交互是指用户与计算机之间的交流，通过各种控件进行。看着菜谱就能够做菜，但控件的选择却不是这样。选择合适的控件意味着：设计者要满足行业标准、公司标准以及用户的需求。

命令按钮（Command Buttons）

在对话框中，命令按钮是用户进行操作的主要方式。它将对话框的主要操作展现给用户。根据命令按钮的名称及放置位置，用户在浏览过对话框后，应该能够知道做什么以及下一步该做什么。

仅对频繁或关键实时的操作设置命令按钮

只对那些频繁或关键的操作设置命令按钮（见图 1）。实际上，命令按钮扮演着醒目显示操作内容的角色。在一个窗体上，命令按钮不应当超过六个。同时命令按钮的操作内容也要出现在菜单项中。如果操作既不频繁也不关键，将其放到下拉菜单中即可。

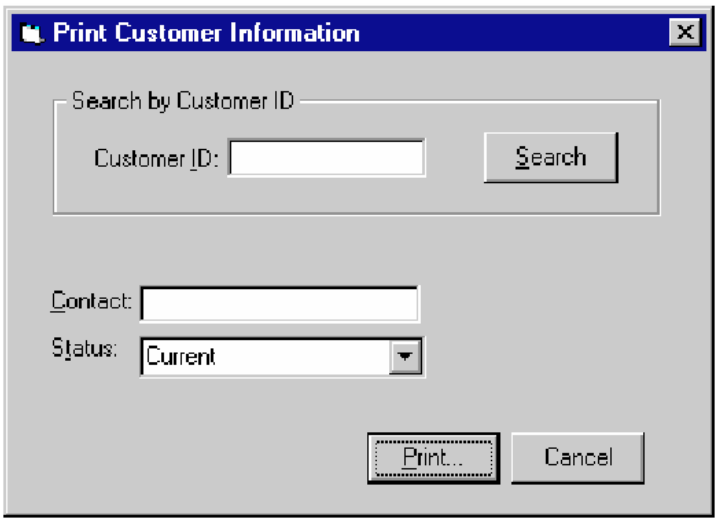


图 1 命令按钮用于频繁和关键的操作

谨慎标注按钮

确信命令按钮的标注清晰简明。例如，使用“Print Setup”（“打印设置”），而不是更多的词。当需要清楚地表达按钮的含义时可以采用多个词，例如，可以用“Print Current Order”（“打印当前定单”），而不能仅仅是“Current”（“当前”）。但，无论如何，要保证标注简洁，剔除冗余的词。同时，要遵循首字母大写规则，即所有主要单词的首字母大写。

保持按钮标注的一致性

为特定的功能选择特定的标注，并在整个应用程序及所有应用程序中都采用这些标注。例如，用“List”来显示表单的选择项，而不是有时用“List”，有时用“Search”。

标注应遵循行业标准

部分标注在图形用户界面中已经形成了标准。当存在表 1 中的操作时，应采用相应的标准标注。

Label	Action	Keyboard Equivalent
OK	Makes changes and closes the window	the Enter key
Cancel	Does not make changes and closes the window	the Escape key
Close	Closes the window when changes can't be made or canceled	C
Reset	Resets to defaults, leaves window open	R
Apply	Makes changes, leaves window open	A
Help	Opens online help	H

表 1 常用操作的标准标注

用术语替代 OK（确认）按钮

如果 OK 命令按钮实际上是像 Print（打印）或 Delete（删除）这些功能的话，考虑用术语代替常用的 OK 按钮，如图 2。

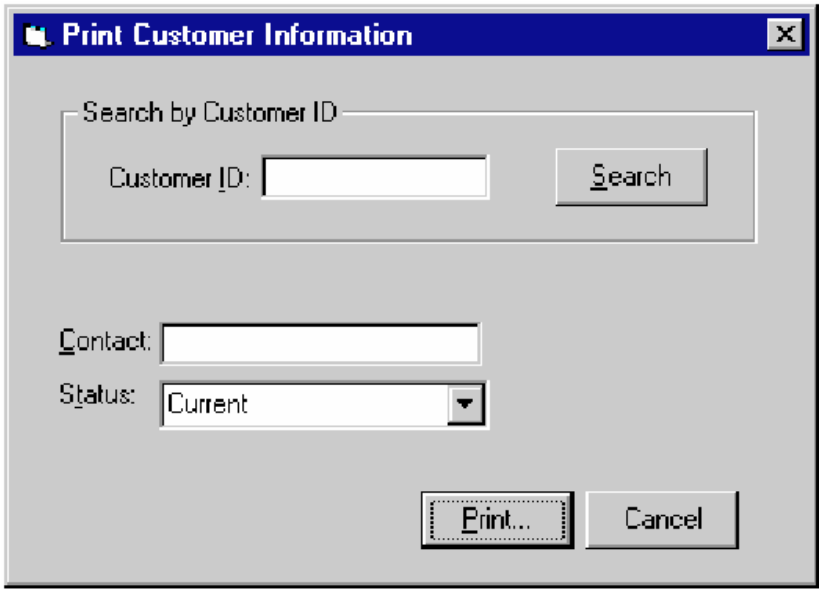


图 2 用 Print 代替常用的 OK

在其它按钮基础上调整按钮尺寸

在对话框中，如果一系列命令按钮的文本长度近似相等的话，调整对话框中所有按钮，使它们的尺寸与最大的按钮尺寸保持相同，如图 3 所示。

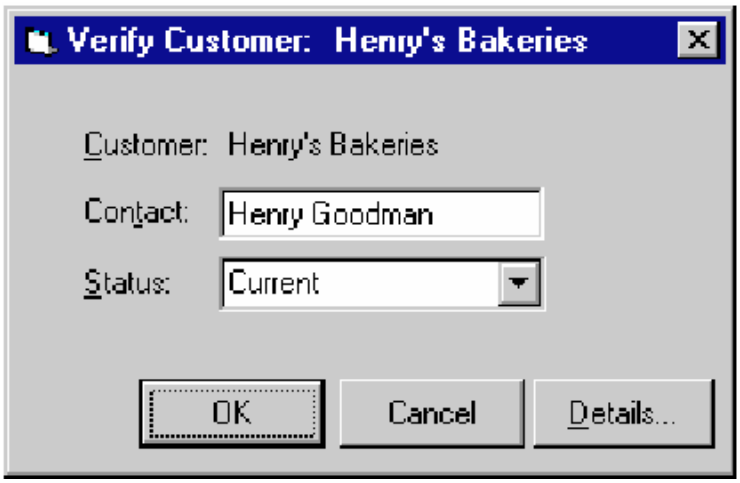


图 3 按钮尺寸与最大的按钮尺寸保持相同

如果对话框上的按钮尺寸相差较大，可以采用两种按钮尺寸——其中一种用于较短的文本，另一种用于较长的文本，如图 4 所示。这样，在避免具有太多不同尺寸规格的同时也保证了需要。在一个对话框中，不应有两种以上的按钮尺寸规格。

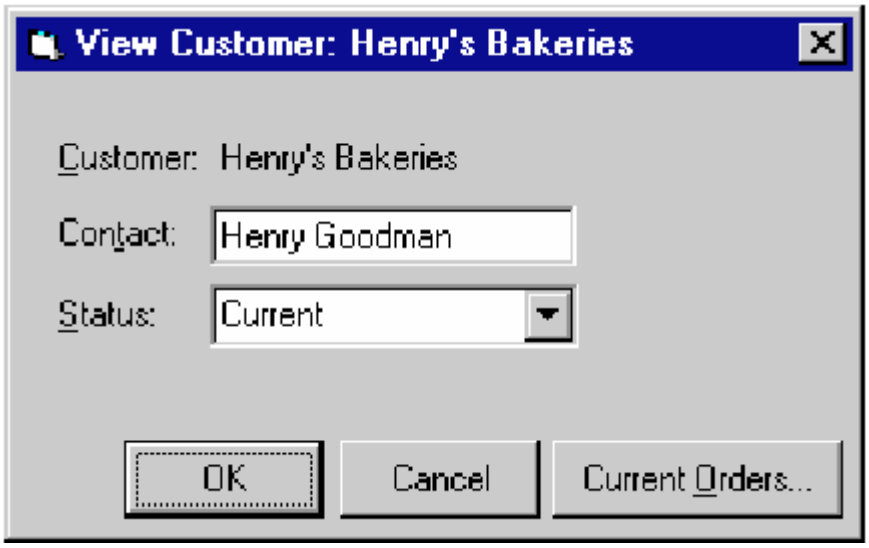


图 4 Current Orders 明显长于另外两个按钮的文字，所以使用不同尺寸的按钮

按钮与对话框其它部分应分开

用空白来分开整个对话框中的按钮，如图 5 所示。而不应当像图 6 中，按钮与对话框其它部分混在一起。

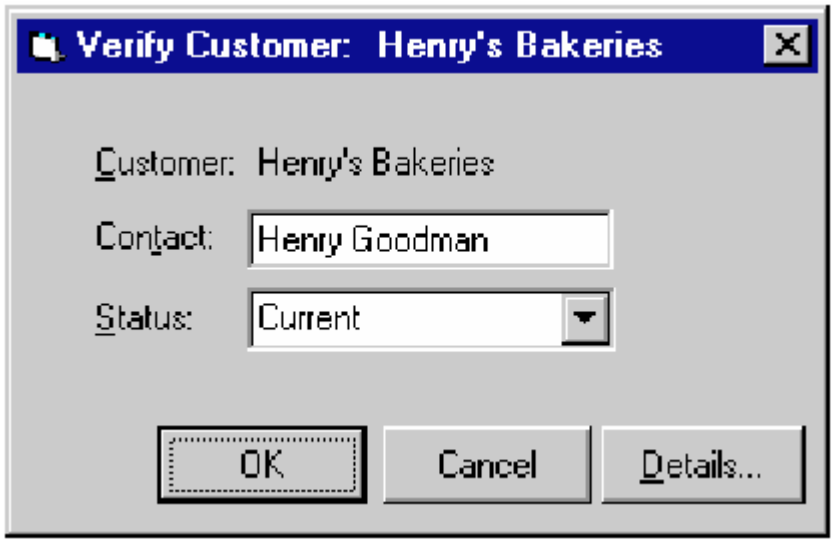


图 5 用空白把按钮和其他部分分开

图 6 不要把按钮和对话框中其他控件混在一起

按钮分组

如果你使用了三个以上的按钮，应用空白对按钮分组(见图 7)。通过分组，可以确定出：

- 具有类似功能的按钮
- 离开该窗口的按钮(OK，Cancel)
- 破坏性操作(删除)

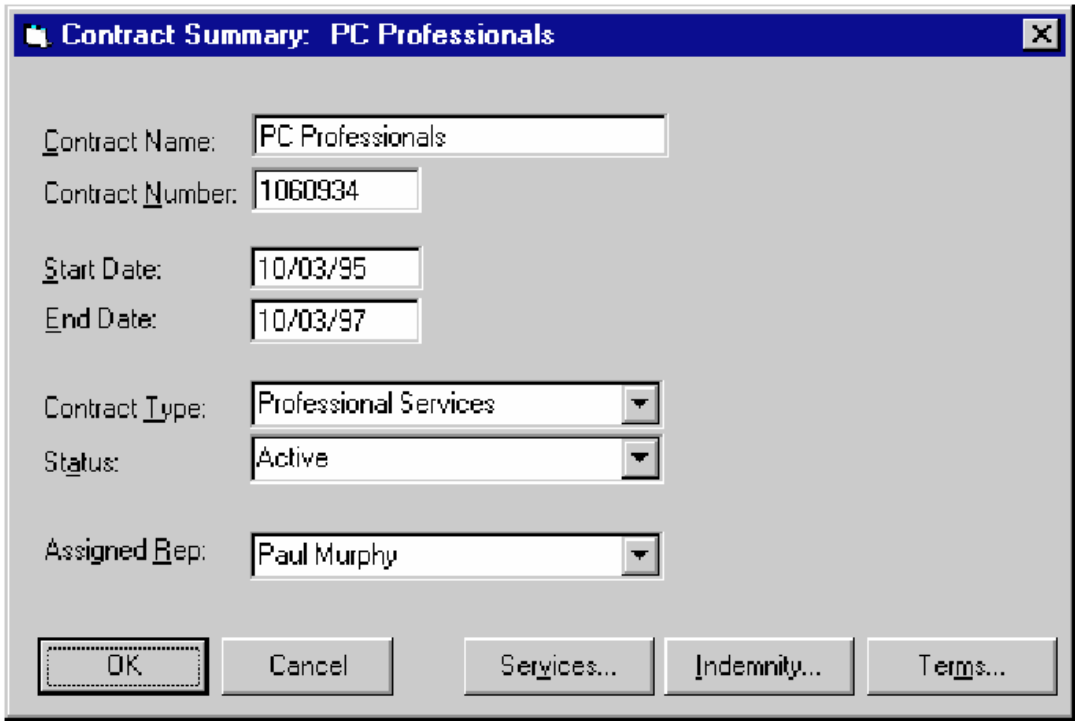


图 7 用空白将相近功能的按钮分组

用统一的方式放置按钮

用下列位置来放置按钮：

- 窗体的右上方(图8，9，10，11)
- 在 Windows95 中，窗体的右下方(图 12)
- 在 Windows3.1 和 Motif 中，窗体的下部居中(图 13 和 14)
- 在 OS/2 中，窗体的左下方(图 15)

在同一个窗体中，窗体的底部及右部不应同时出现按钮。

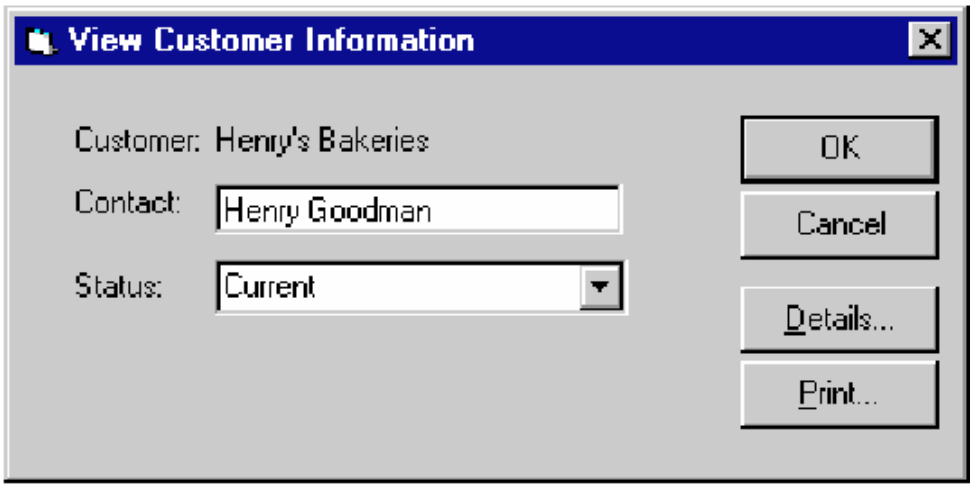


图 8 Windows 95 的右上角命令按钮

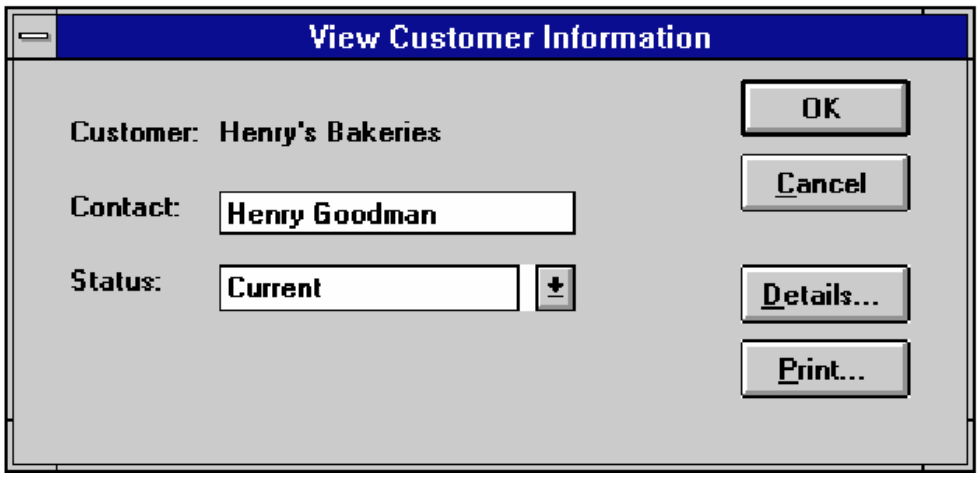


图 9 Windows 3.1 的右上角命令按钮

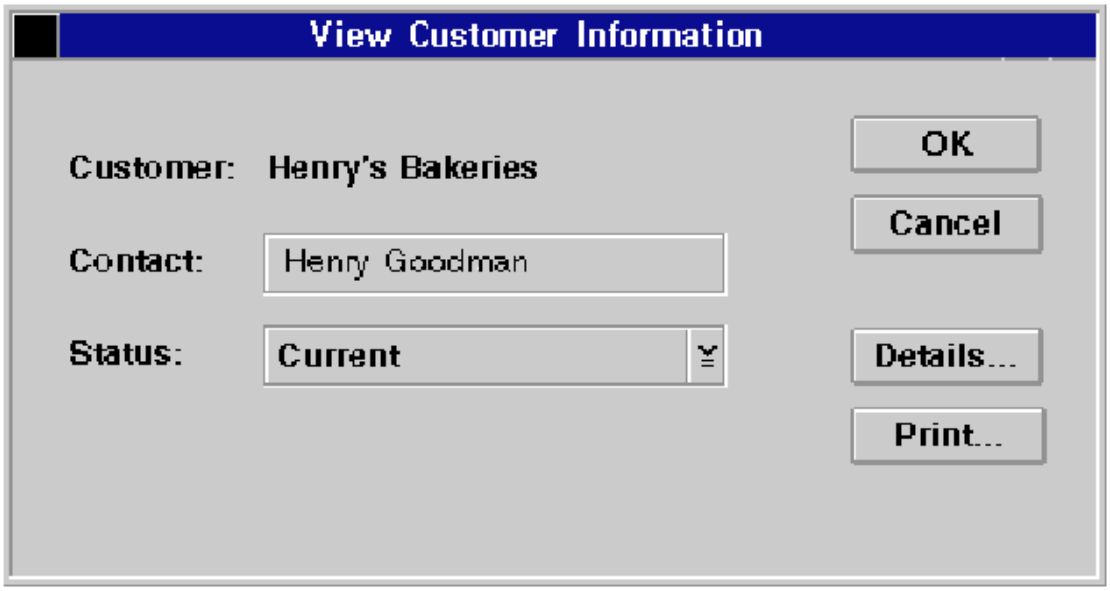


图 10 OS/2 的右上角命令按钮

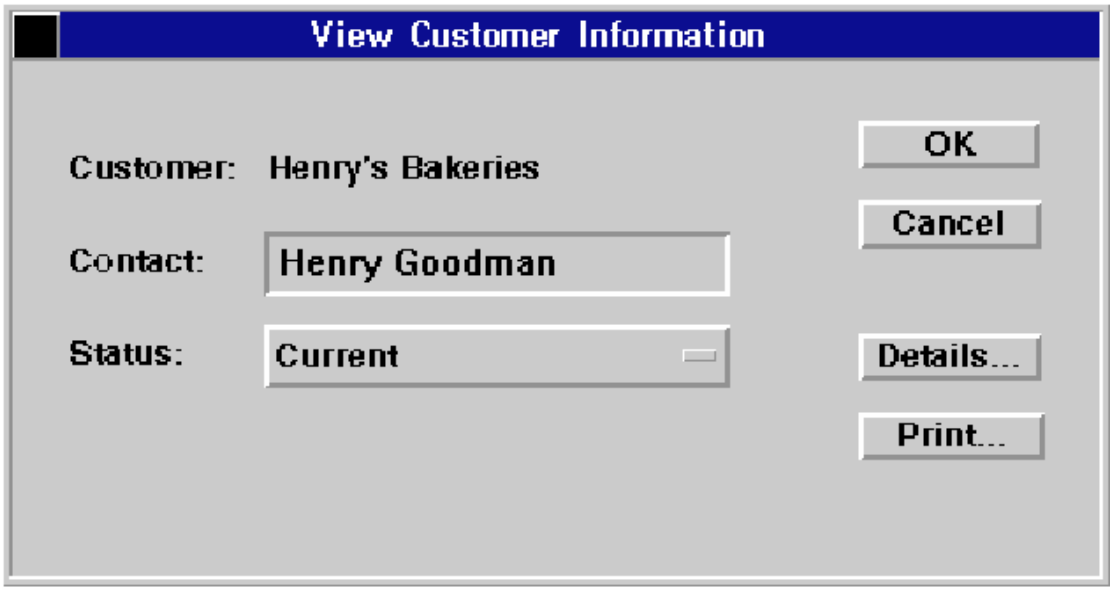


图 11 Motif 的右上角命令按钮

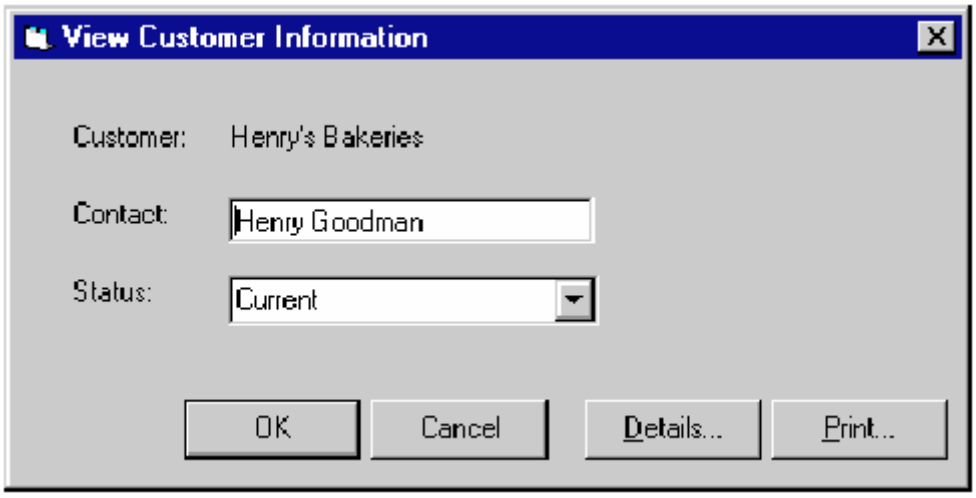


图 12 Windows 95 的底部居右命令按钮

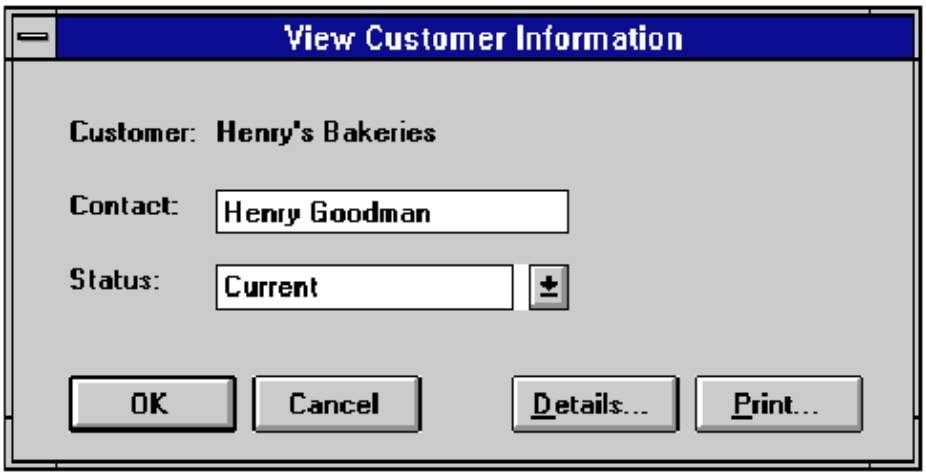


图 13 Windows 3.1 的底部居中命令按钮

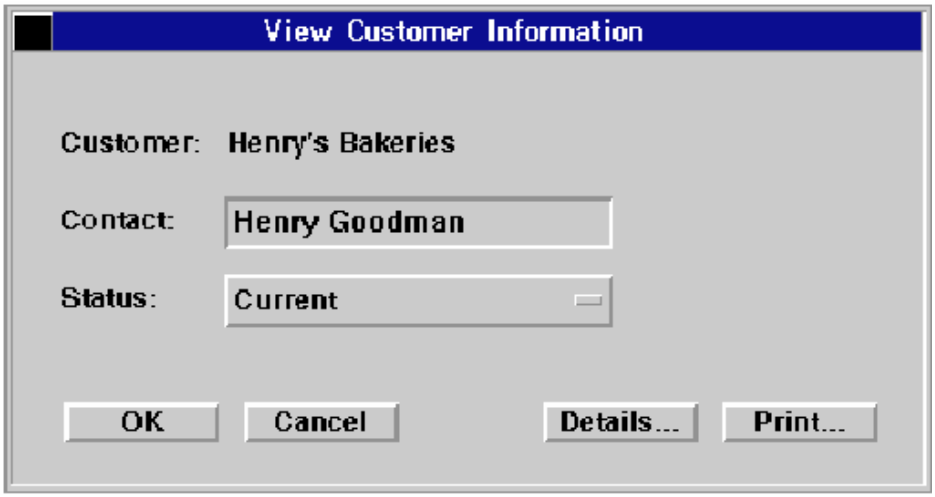


图 14 Motif 的底部居中命令按钮

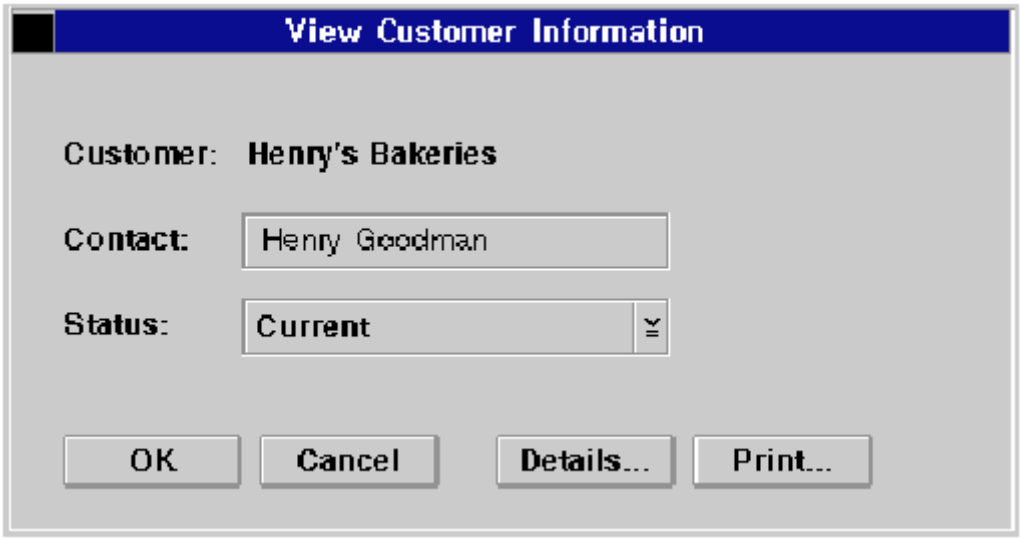


图 15 OS/2 的底部居左命令按钮

按钮位置应适应窗体

为某个特定的窗体选择设计方式：垂直或水平；按钮的放置应该适应窗体的设计方式。如果采用水平设计，按钮应当放在右上方，如图 16。采用垂直设计的话，应当放在底部，如图 17。

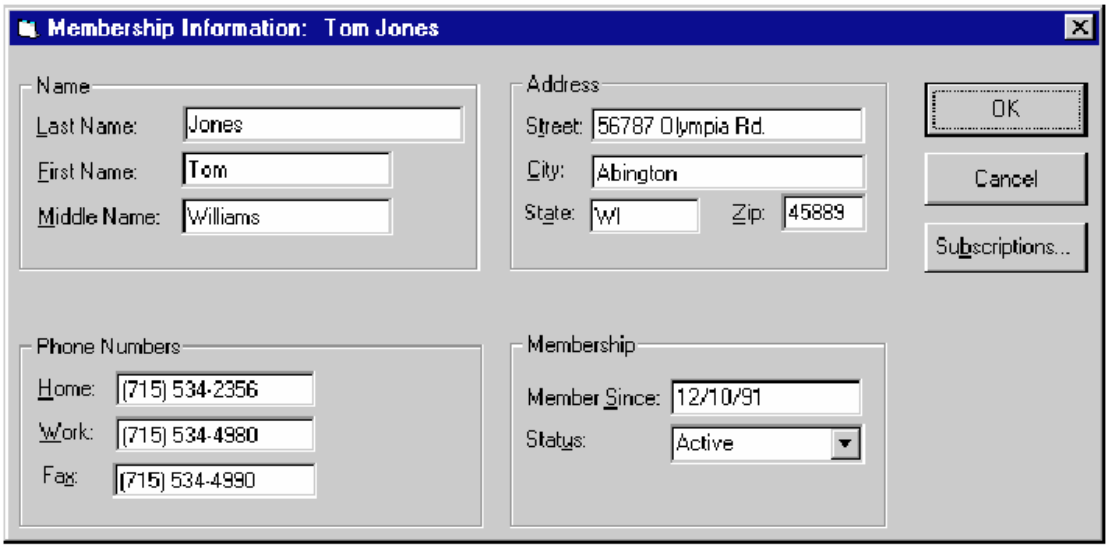


图 16 Windows 95 水平窗体的命令按钮摆放

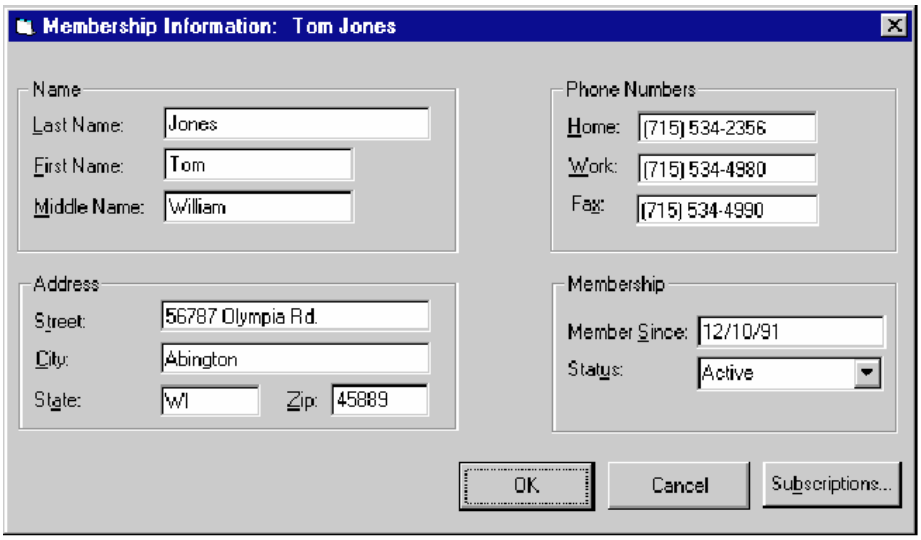


图 17 Windows 95 垂直窗体的命令按钮摆放

窗体上数据的分组及分布有着重要的作用，它决定了采用哪一种设计方式。按钮的长度和数量也是影响的因素。如果有长的按钮名称或者有很多按钮，你可能会采用将按钮放于右上方的水平设计。在基于窗体或对话框的风格中，你也可以采用这样的设计。窗体流（Window Flow）的设计并不需要在所有的窗体或对话框上都保持设计方式的一致性。

仅有有限操作的按钮应按需放置

如果命令按钮仅仅属于对话框的一部分，将按钮放在恰当的位置。图 18 中，“Search”（“搜索”）按钮被放在合适的位置。

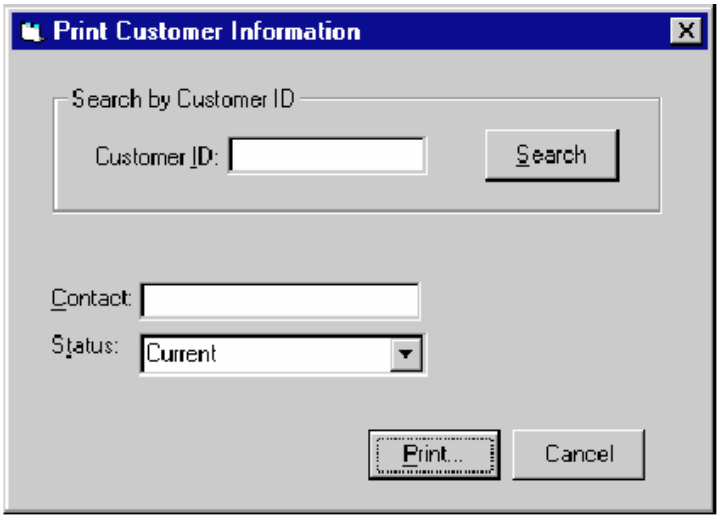


图 18 Search 按钮放在合适的位置

按钮放置顺序的一致性

按钮应尽可能以下列顺序排放：

1. 离开窗体的确认按钮(OK)；
2. 离开窗体的取消按钮(Cancel)；
3. 窗体的独有按钮。

当按钮放置于底部或右上方时，其顺序相同 (见图 19)。

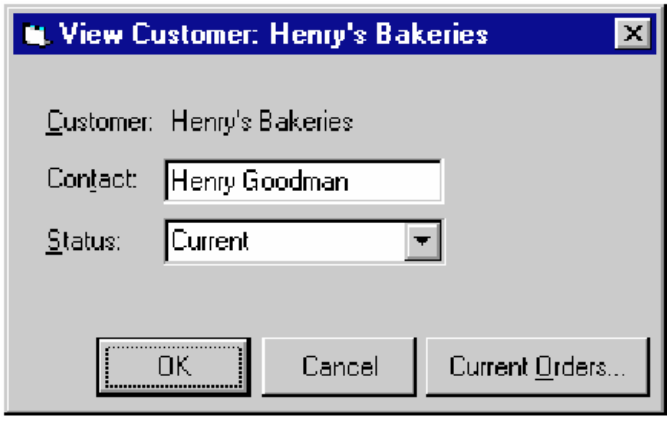


图 19 确认按钮(OK)在第一个，取消按钮(Cancel)随后，然后是该窗体的特定按钮（Current Orders）

用省略号（…）来表示需要输入数据

当完成一个按钮的操作需要更多的输入，应该在按钮名称后加省略号（…）（参见图 20）。

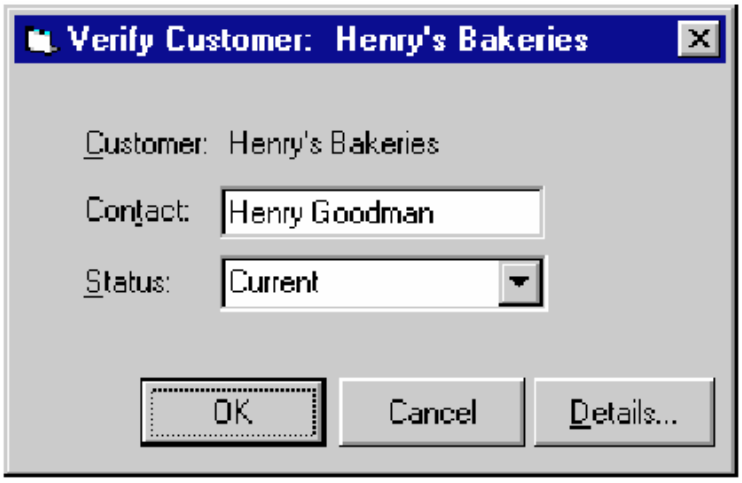


图 20 按钮文本后用省略号，当选择 Details 时，另一个对话框出现

灰掉不可用的按钮

当按钮不可用时，将其灰掉。例如，为了限制用户的操作，一些操作只能在其他操作完成后才能进行。在图 21 中，展示了“Search”操作可用时该按钮的外观。在图 22 中，“Search”操作不再可用，按钮被灰掉。

图 21 “Search”操作可用

图 22 “Search”操作不可用

选择非破坏性按钮作为缺省按钮

从窗体上选择一个按钮作为缺省按钮。当用户敲回车键时，激活该按钮。通常，以窗体上最常用的或重要的操作作为缺省值。例如，打印窗口上的“Print”（“打印”）按钮。像“Delete”（“删除”）这样的破坏性按钮，即使是窗体上最常用的或最重要的操作，也不要将其作为缺省按钮。

单选按钮（Option Buttons）

单选按钮（Option Buttons，也写做 Radio Buttons），通常起着数据输入操作的作用。

用单选按钮做唯一选择

在一组选项中，当用户要选择出排它性的选择项时，可以用单选按钮。例如，在关于人员管理的应用程序中发薪周期的选择。

标注单选按钮

为每一个单选按钮挑选一个简洁而又能表达清楚的标注，例如，使用“Send Course”而不是“Course”这样的标签。

集中放置单选按钮并标注

将单选按钮放置在一个组中，用一个框架（Frame）来展示。对于整个组选用一个能够表达清楚的标注（见图 23）。

图 23 单选按钮被分组和标注

竖直排列单选按钮

在竖直方向上排列单选按钮（图 24）。只要有空间，就不应当水平排列（图 25），以便使得用户能够清楚地看到所有项。

图 24 垂直对齐的单选按钮

图 25 不要把单选按钮水平对齐

单选按钮要限制在六个以内

限制单选按钮的数目，使其不要超过六个。如果有更多的选择项，可以考虑采用列表框（List Box）来替代。列表框将在以后的章节讨论。

顺序的选择

在确定单选按钮的最佳顺序时，可以采用如下排序方法：

- 根据频度－最常用的选项放在顶部
- 根据任务关系－假如存在任务的执行顺序
- 根据逻辑关系－假如存在逻辑关系，例如对于日期
- 根据字典顺序－仅仅在用户这么考虑时才采用

避免只有两个单选按钮的情形

如果用户要做出 yes/no 或 on/off 这样的选择，采用复选框（图 26），而不应当是单选按钮（图 27）。然而，对于一些截然不同、互斥的选项，如 male/female，请采用两个单选按钮。

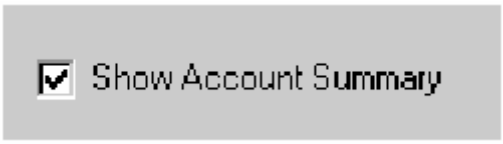


图 26 为 yes/no 选择采用复选框

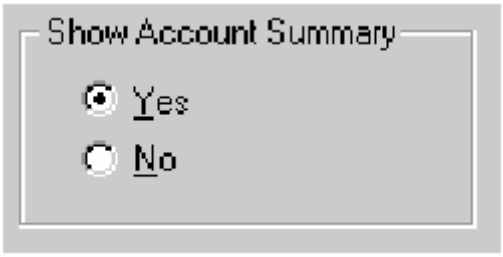


图 27 不要为 yes/no 选择采用两个单选按钮

复选框（Check Boxes）

复选框起到了数据输入操作的作用，提供了一个快速进行多选的办法。

复选框用于选择一个以上的选项

当用户要选择一个或多个选项时，应采用复选框。

复选框用于选项取值的切换

当用户在某个属性上进行切换，以决定取值与否时，应采用复选框，如图 28 所示。这时，一个复选框就足够了。

图 28 使用复选框打开和关闭某个特性

标注复选框

为每一个复选框挑选一个简洁而又能表达清楚的标注。例如，使用“Reverse Print Order”而不是“Reverse”这样的标签。

集中放置复选框并标注

将复选框放置在一个组中，用框架（Frame）来展示。对于整个组采用一个能够描述清楚的标注（见图 29 至图 32）。

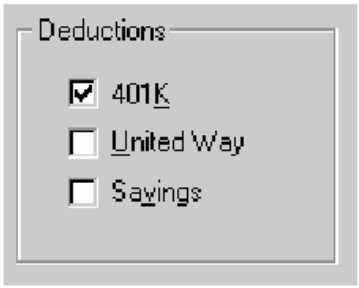


图 29 Windows 95 复选框

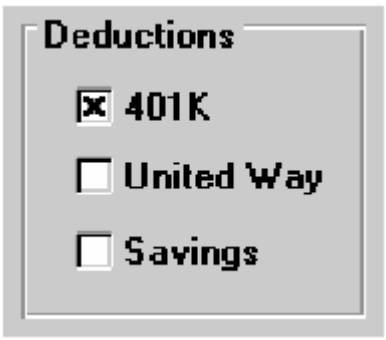


图 30 Windows 3.1 复选框

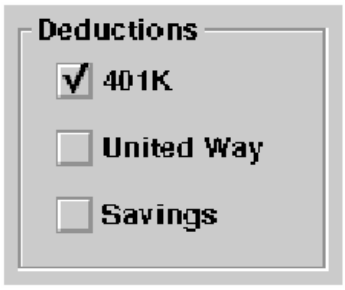


图 31 OS/2 复选框

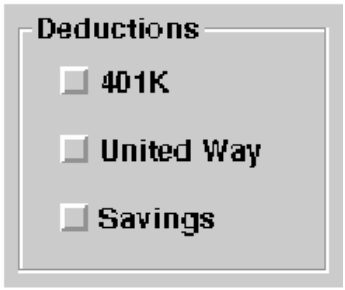


图 32 Motif 复选框

竖直排列复选框

在竖直方向上排列复选框（图 33），只要有空间，就不应当水平排列，以便使得用户能够清楚地看到所有项（图 34）。

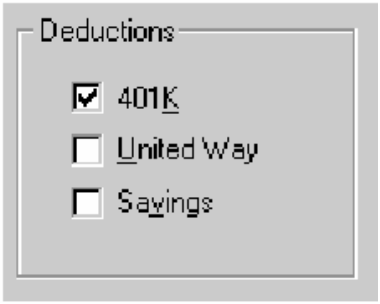


图 33 竖直排列复选框

图 34 不要水平排列复选框

复选框要限制在十个以内

限制复选框的数目，使其不要超过十个。如果有更多选择项，应当采用复合式选择列表框（Multiple Select List Box）来替代。

顺序的选择

在确定复选框的最佳顺序时，可以采用如下排序方法：

- 根据频度－最常用的选项放在顶部
- 根据任务关系－假如存在任务的执行顺序
- 根据逻辑关系－假如存在逻辑关系，例如对于日期
- 根据字典顺序－仅仅在用户做出这样的考虑时才采用

不要带有“选中所有”或“取消所有已选中”的复选框

如果你期望用户能够选中所有的复选框，或者一次性取消所有选中的复选框，应当采用一个带有“Select All”和“Deselect All”按钮的复合式选择列表框（图 35），而不是复选框（图 36）。复合式选择列表框将在以后讨论。

图 35 使用带有“Select All”和“Deselect All”按钮的复合式选择列表框

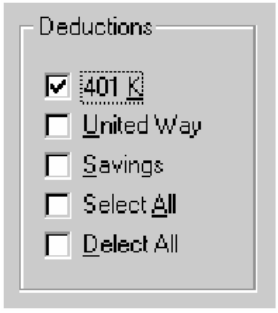


图 36 不使用“Select All”和“Deselect All”的复选框

文本框（Text Boxes）

文本框是用户输入数据的主要方式。

用边框表示数据输入

用带有边框的文本框来表示用户能够输入或编辑数据，如图 37。

图 37 在文本输入区域周围加边框

仅仅显示时文本框应无边框

如果数据仅仅为了显示，不能进行修改或增加，则不必在其周围放置边框（见图 38）。

图 38 仅为了显示数据时不要加边框

灰掉临时受保护的字段

当特定的文本框临时受保护时，灰掉该文本框，表明数据不能输入或修改。图 39 展示了一个数据能够修改的字段。而图 40 展示了一个数据不够修改的字段。

图 39 可以修改的数据

图 40 数据不可修改时变灰

用文本框的长度表明文本的近似长度

调整文本框的大小，使其表示字段的大致长度，如图 41。如果你有相近长度的文本框，使它们长度相同，除非必须表示出字段的精确长度（图 42）。

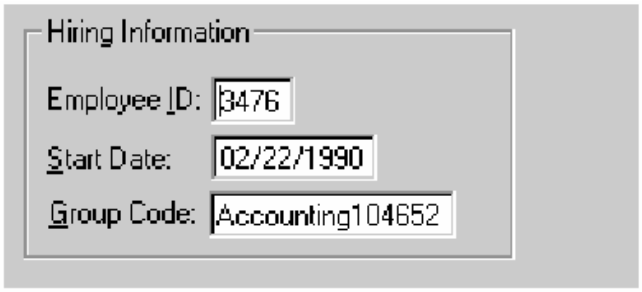


图 41 每个文本框有自己的长度，表明该域的确切长度

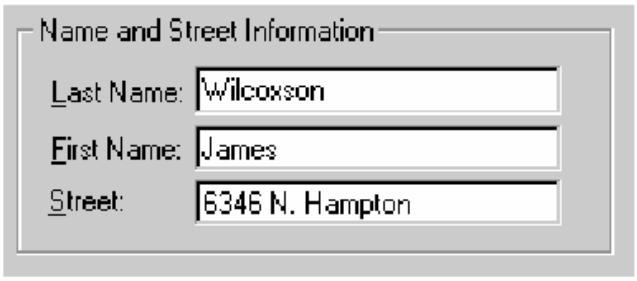


图 42 相同长度的文本框表示不同数据

排列文本框

左排列屏幕上的文本框，保证不同的窗体边缘数目最小（见图 43）。如果存在有着长标注的文本框，则对该文本框使用不同的窗体边缘。限制不同的窗体边缘数目，使其不超过两个。

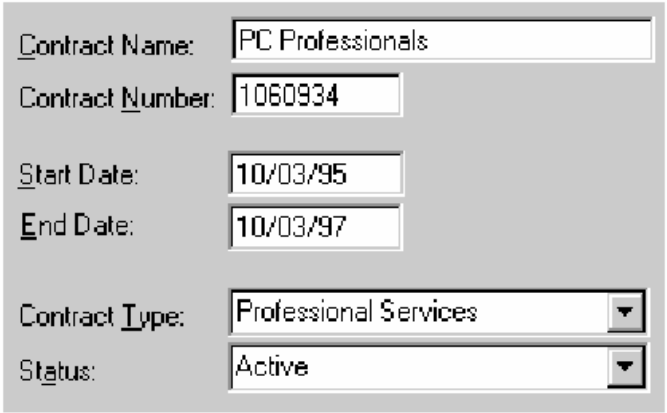


图 43 文本框左对齐

文本框的分组

当存在属于同一信息的文本框时，将它们集中放置在一个框架中，并标注整个组。

标注所有文本框

为每一个文本框增加一个能够描述清楚的标注。只有在所有用户都明白首字母缩写或简写时，才能使用它们。采用多个词进行标注是一个很好的选择。但是，要保证简洁。把标注的第一个词的首字母大写。

靠左放置标注

在文本框的左边放置其标注。不要将标注放在文本框的上方。

左对齐排列文本框标注

靠左排列文本框（见图 44）。不要使用右排列。右排列会造成窗体左边边缘的参差不齐，导致浏览的不便（见图 45）。

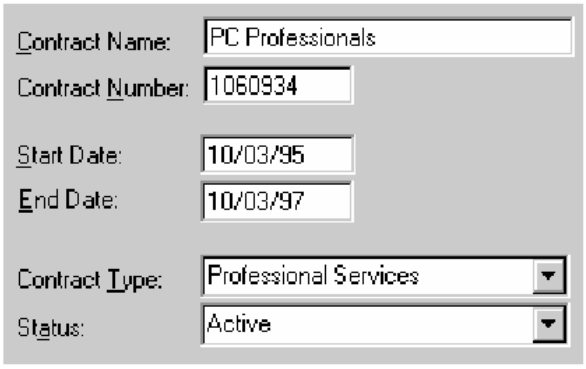


图 44 标注左对齐

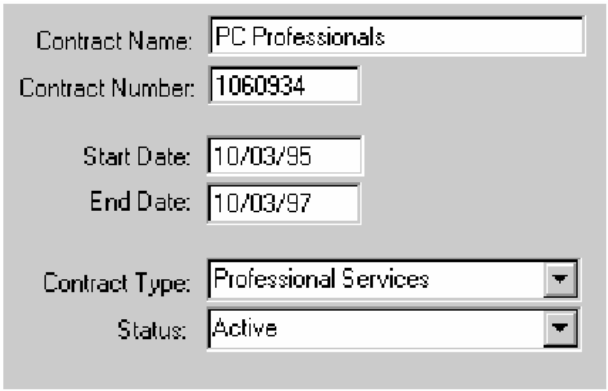


图 45 不要让标注的左边不平齐

在文本框标注后放置冒号

在文本框的标注后使用冒号，用于区分标注和接着的数据（见图 46）。在组框架和表格表头后，不要使用冒号。

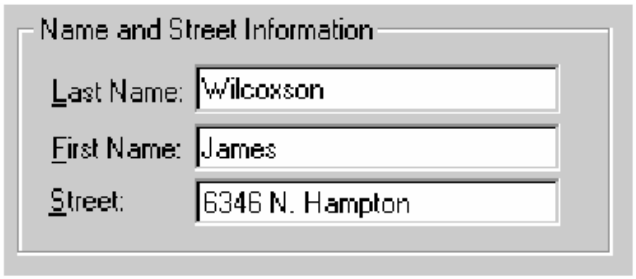


图 46 在文本框的标注后使用冒号，但不要在组的标注后使用冒号

列表框（List Boxes）

列表框是用来放置长的单选按钮列表的一种方法（图 47 至图 50）。它们也起到输入数据的作用，同时保证了数据完整性。

对长的列表采用列表框

当你有非常多的选项时，应采用列表框而不是单选按钮。当你有多于六个单选按钮时，应用一个单选的列表框。

动态数据应采用列表框

如果数据在时间上会发生变化，用列表框而不是单选按钮。这样，改变列表框中的选项相对更容易。

同时显示三到八项

在同一时间内，一个列表框至少应显示三项，但不要超过八项。当存在更多的项时，可以用滚动条来查看其它项。可参见本章后面的下拉列表框的使用指南。

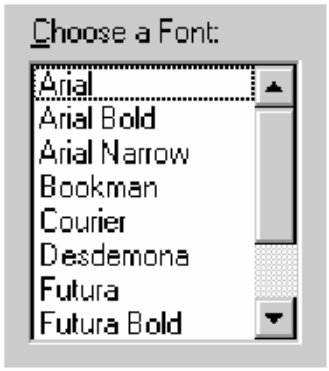


图 47 Windows 95 列表框

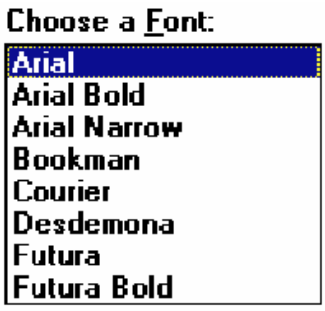


图 48 Windows 3.1 列表框

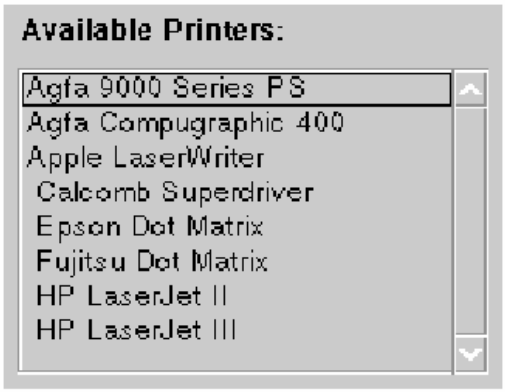


图 49 OS/2 列表框

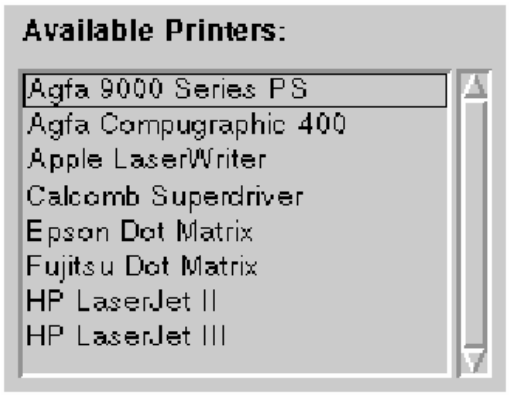


图 50 Motif 列表框

标记每一个列表框

为整个列表框挑选一个标注，并且该标注能够描述列表框中的所有项，例如“Available Printers“。将标注放置在列表框的顶部，靠左放置，并在标注后加上冒号。

对于长的列表采用过滤器

如果操作超过 40 个选项的列表，应提供一种过滤的方法，减少选项数，如图 51。

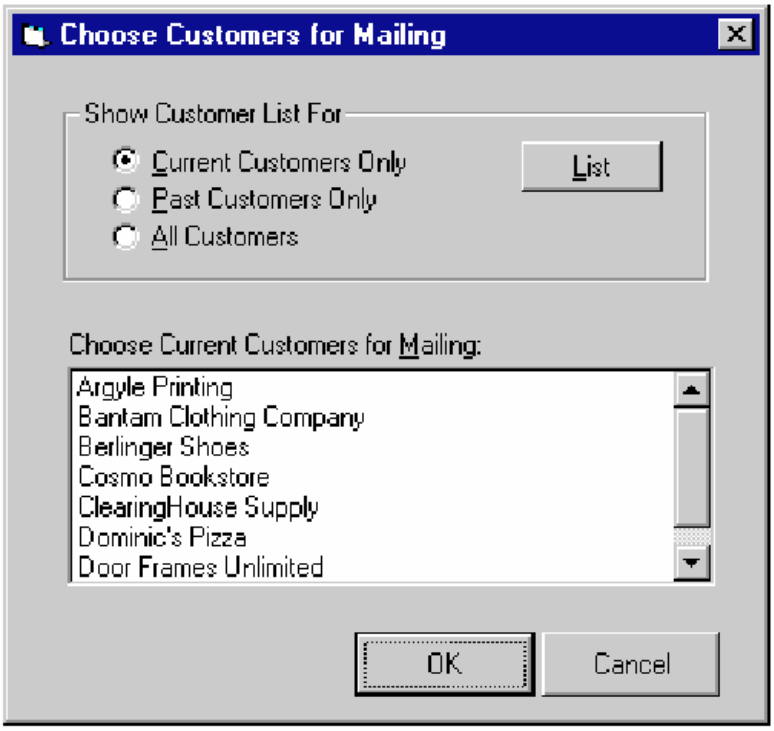


图 51 对长列表框使用过滤器

用下拉列表框节省空间

下拉列表框能够节省窗体空间。如果多数用户都会单选第一项时请用下拉列表框。然而，列表框除第一项外其它项都对用户隐藏。用户需要增加一个额外的步骤来得到列表的其它项。当用户需要同时看到所有的项时，不应当用下拉列表框。

用组合列表框允许用户输入

组合列表框允许用户像从列表中选择一样，输入选择项。当大部分用户已知他们所想要的，并且更愿意自己输入时，应采用组合列表框。如果列表很长，用户可以通过键入一个或多个字母来减少候选项时，组合列表框也很有用。禁止用户通过组合列表框在列表中增加项。

复合式选择列表框（Multiple Selection List Boxes）

复合式选择列表框以另外一种方式实现了长的复选框列表。然而，复合式选择列表框比较难以使用。你可以遵循下面的帮助来解决这个问题。

用复合式选择列表框替代复选框

如果存在十个以上的选项，或者列表会发生变化，此时，应当用复合式选择列表框替代复选框。

为复合式选择列表框增加使用说明

许多用户对复合式选择列表框并不熟悉。他们可能不知道一次能够选取一个以上的选项，或者不知道如何做到这一点。当一个窗体包含一个单选列表框和一个复合式选择列表框时，使用说明显得尤为重要。

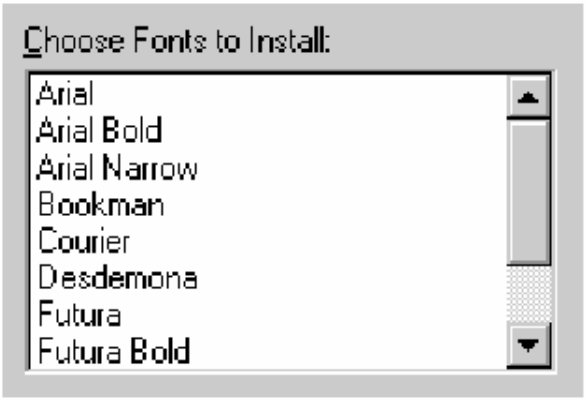


图 52 标注中要包含说明

使用选择项汇总框

当使用具有滚动条的复合式选择列表框时，考虑增加一个框，用于显示用户选择的项（见图 53）。通过这种方法，用户不必在列表上移来移去查看已经选了哪些项。

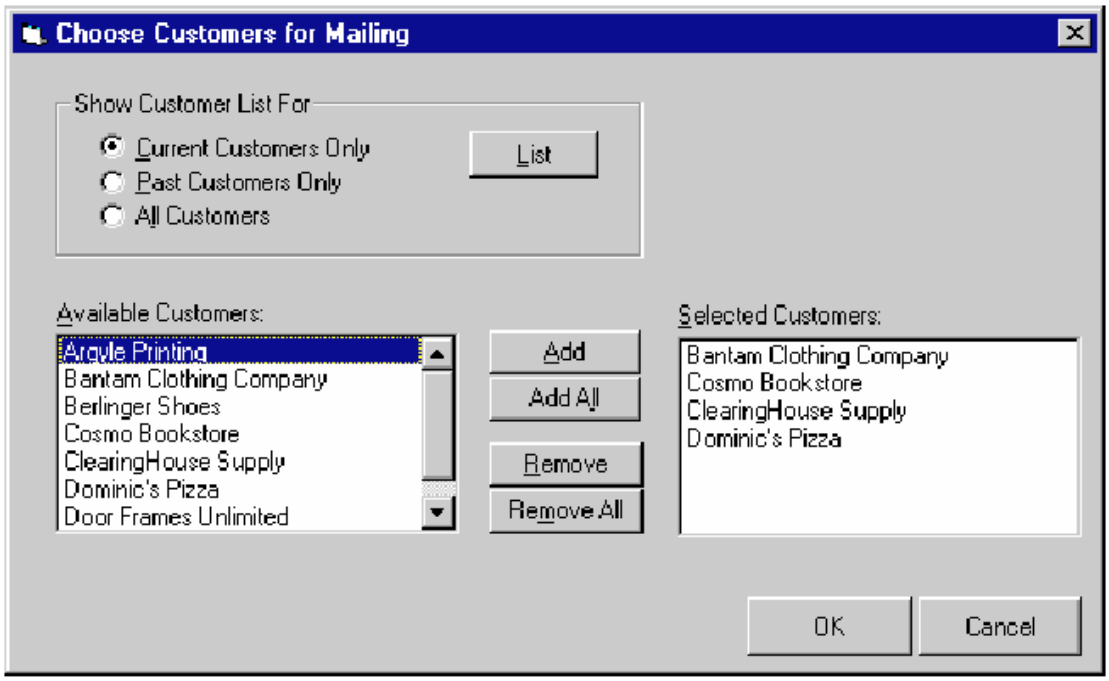


图 53 用选择项汇总框表示用户已经选择了哪些

复合式选择复选列表

展示用户所选选项的另一种方法是多选复选列表。它综合了复合式选择列表框与用户能选择的复选框。

使用选中所有和取消所有已选项按钮

如果有一列选项，你希望能够同时选中所有的选项，或取消所有已选项，不应该使用复选框，而是使用带有选中所有和去除所有已选项按钮的复合式选择列表框。

表和网格（Tables and Grids）

表和网格能够让用户同时输入或查看大量的信息。

表可用来进行数据之间的比较

当用户需要比较数据，而你又不能提前预知用户需求的话，可以用表来展示（见图 54）。

	1st Qtr	2nd Qtr	3rd Qtr	4th Qtr
East	20.4	27.4	90	20.4
West	30.6	38.6	34.6	31.6
North	45.9	46.9	45	43.9

图 54 使用表来比较数据

网格用来进行多数据输入

网格能够使得用户同时输入多个数据。

标注列

为列选择能够反映数据的标注。

必要时使有行标注

如果行包含不同的数据，标注每一行。

左调整标注

左调整列和行标注。在标注后不要用冒号。

微调框（Spin Boxes）

微调框提供了另一种让用户输入数据的方式。

微调框用于有限循环

当选择列表小于十个，其顺序可预期时，可以用微调框在可能的选择值上进行循环轮转，如图 55。

图 55 微调框

微调框与文本框结合使用

当使用微调框时，可以将其与文本框结合使用，这样，用户除了可以轮转循环外还能够输入指定的值。

滑动条（Sliders）

当无需精确值时，滑动条提供一种快速调整值的方法。

滑动条以可见方式调整值

可以用滑动条控件来增大或减小连续值。如果你将结果一并显示的话，该控件更为有效（见图 56 至 59）。

图 56 Windows 95 滑动条

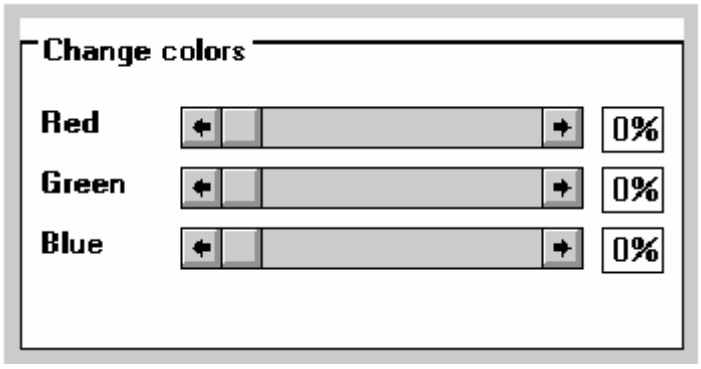


图 57 Windows 3.1 滑动条

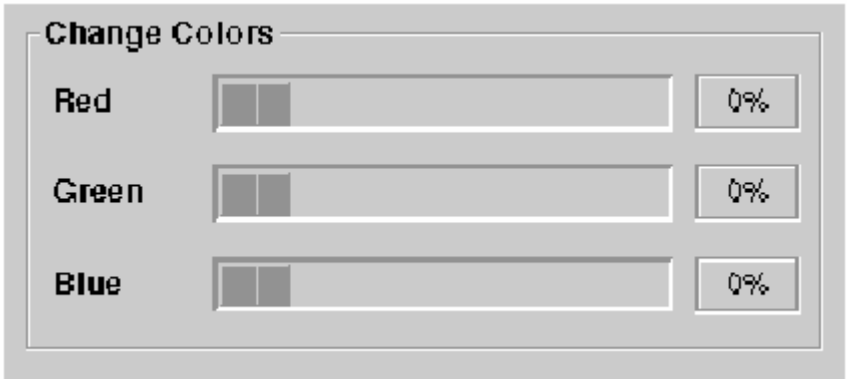


图 58 OS/2 滑动条

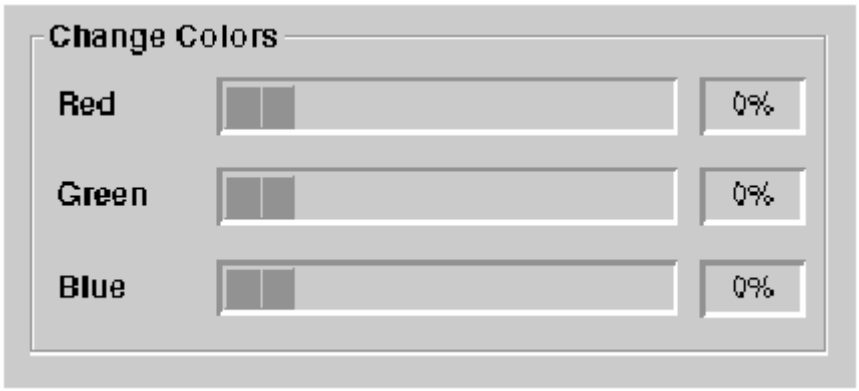


图 59 Motif 滑动条

滑动条适用于大的数据区间

在数据的选择范围小于十的情况下，不要采用滑动条。小的数据区间可以采用其它的控件，例如，微调框或文本框进行输入。微调框及文本框已在前面讨论过。

显示结果

显示滑动条所在位置所代表的实际值。

允许输入数据

在用户知道精确值的情况下，可以让他们之间输入值，而不是使用滑动条。

用箭头符号表示小的改变

当用户接近所要的值时，用滑动条两端的箭头进行小的调整。

树（TreeViews）

树用来展示列表中各项之间的关系。

树用于层次等级重要的情况

树能够表达出元素之间的层次等级关系（见图 60）。当用户需要从一列中选择其中一项，并且该项的位置依赖于所在的层次等级时，树非常有效，诸如文件及目录系统。当层次等级中各项之间的关系能够帮助用户确定所要寻找的项时，树也非常有用。展开（Expanding）及收缩（Collasping）操作允许用户在查找时改变详细内容的数目。

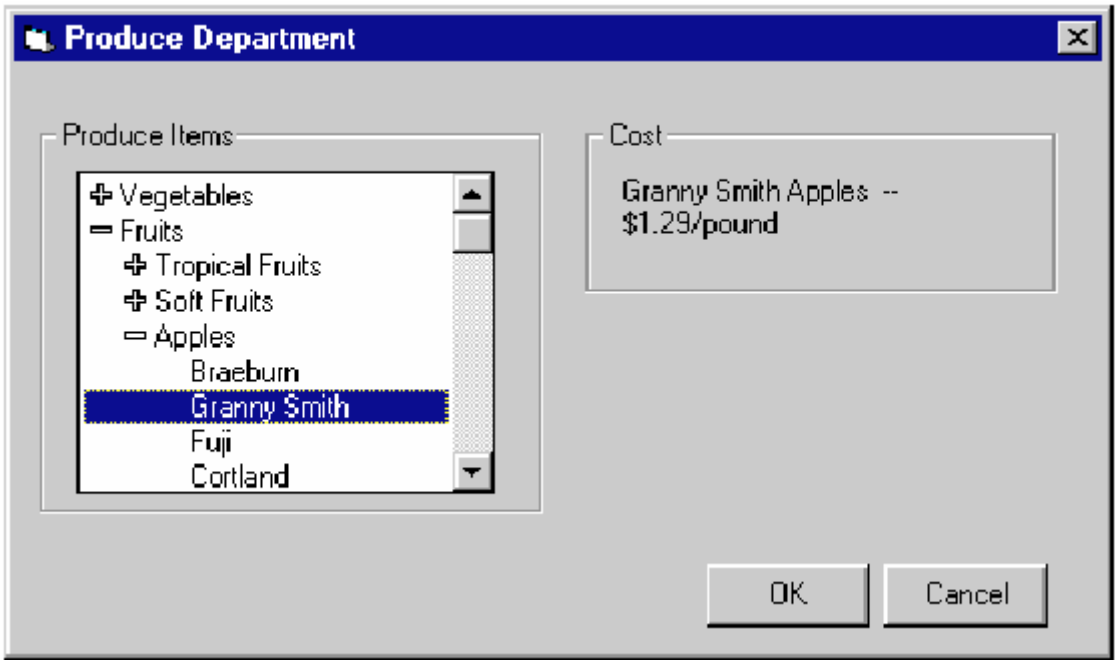


图 60 树表示层次

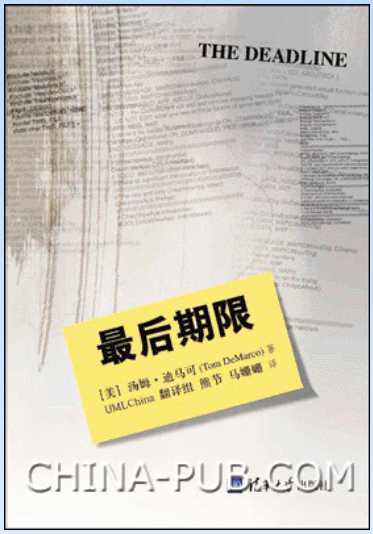
树：高级用户的选择

大部分用户属于新手，没有在开发环境中使用过树，对它们不很熟悉。如果各项之间的关系不能正确理解的话，树就很难使用，用户可能没有意识到展开（Expanding）及收缩（Collasping）的功能特性。

树不能替代菜单（Menus），启动栏（Home Bases）和快捷栏（Launch Pads）

不要用树替代菜单系统，启动栏和快捷栏。它们仅仅在搜索和选择控制项时有效。

《最后期限》



Tom Demarco

翻译：UMLChina 翻译组 熊节

汤普金斯在飞机的座位上翻了一个身，把她的毛衣抓到脸上，贪婪地呼吸着它散发出的淡淡芬芳。文案，他对自己说。他试图回忆当他这样说时卡布福斯的表情。当时他惊讶得下巴都快掉下来了。是的，的确如此。文案……吃惊的卡布福斯……房间里的叹息声……汤普金斯大步走出教室……莱克莎重复那个词……汤普金斯重复那个词……两人微张的嘴唇触到了一起。再次重播。"文案。"他说道，转身，看着莱克莎，她微张的嘴唇，他……倒带，再次重播……

....

"我不想兜圈子，"汤普金斯看着面前的简报说，"实际上你们有一千五百名资格相当老的软件工程师。"

莱克莎点点头："这是最近的数字。他们都会在你的手下工作。"

"而且据你所说，他们都很优秀。"

"他们都通过了摩罗维亚软件工程学院的 CMM 2 级以上的认证。"

总把新桃换旧符——写在《最后期限》出版之前

透明 著

查看评论

上个星期，《最后期限》的编辑给我一个电话，告诉我这本书在本周内就会生产出来了，让我颇感欣慰。从 8 月以来，心情一直不好，在上海与 CSDN 网友见面之后也没有什么特别值得高兴的事情，几个月火气都很大，常常在网上网下跟人吵架。最近北京的雪很漂亮，Adams 的《人月神话》读得心旷神怡，我的书终于快要出版。在圣诞节之前，心情有了一点亮起来的理由。这篇文章是早就酝酿要写的，直到今天，终于有了一点感觉要写出来。

既然《最后期限》很快就要出版，就先从这本书开始说起吧。总体来说，《最后期限》是一个有趣的故事。因为有趣，所以它可以帮助读者杀时间。但是，除了杀时间之外，能不能对读者的工作有所帮助，我就不敢断定了。对于一本小说，本来也不应该对它有太多的期望；但抱着“没有期望”的心态去读一本小说，或许就有了收获也未可知。

是的，作为读者，你不能（也不该）希望一本小说能对你的工作有多大帮助。你很有可能遇到像卡布福斯那样不知天高地厚的培训讲师，也很可能遇到像贝洛克这样可恶的上司，如果运气好的话还能遇到像“元首”这样英明的老板。不过，当你的项目面临最后期限的挑战时，你无法希望会有那么多优秀的顾问来帮你——这种事情只会发生在小说里。

总之，整本《最后期限》透着一种令人愉悦的快乐氛围。问题尽管很麻烦，总能一个一个地被解决；坏人尽管很嚣张，终于还是被赶走了；最后期限实在太紧迫，但和汤姆·克鲁斯一样，“不可能的任务”最后还是顺利完成了。还有，尽管被工作累得晕头转向，男主角最后还是能抱得美人归。但是，这种快乐的气氛往往会给读者一种不切实际的幻想，让他们以为小说中的故事也能发生在自己身上。如果半年后有读者来骂这本书的话，我猜他一定是因为南柯一梦醒来之后觉得落寞。

Tom DeMarco 似乎总有一脸和蔼的笑容，配上银白的头发，形象很不错，就像《最后期限》一样能令人心情舒畅。但是，作为第一代的软件科学家，他写作的态度难免让不明就里的人心里犯嘀咕。《最后期限》这样似是而非的作品不用说，还有那本被称为“无产者的呓语”的 *Slack*，就连最著名的《人件》也透着一种过度乐观和“加料过火的呆伯特式的幽默”。Jacques Lebrun 的批评一针见血：

> 如果你恰好处于作者们设想的位置，并也走在相应的方向上，作为一个殷勤的侍者，它（指《人件》这本书）会为你打开一扇特定的门。但似乎不应指望，它能领你一路回家。
>
> ----Jacques Lebrun，《人的问题：关于 *Peopleware*》，《程序员》杂志，2002 第 12 期。

如果有可能的话，在一个星期之内读《人月神话》和《最后期限》这两本书，绝对是一个绝妙的阅读体验。你能看到 Brooks 和 DeMarco 在几乎相同的知识背景下得到的两种截然不同的态度："人月是神话、银弹无处求"的悲观，和"逢山开路、遇水架桥"的乐观。如果读者再带着项目的紧迫压力希望在这些武林秘籍中找到一些灵丹妙药，被这一阴一阳的两股内力搅和搅和，怕是脑子会被弄晕掉。

或者，你可以把这两本书看成是对"软件的科学"的两种哲学思辩——假如软件工程中的确有科学成分的话。1978 年，Tom DeMarco 那本名字大得吓死人的书——*Structured Analysis and System Specification*——出版，看得出当时他对结构化方法和软件工程是很有信心的。但是，当软件从军队、政府广泛地转向民用之后，情况越来越多地转向了《人月神话》所指的方向：软件工程可以解决（很少的）一些问题；某些问题似乎永远都无法解决。于是，Brooks 更加坚定了自己的观点，并相继推出了《没有银弹》和《再论没有银弹》两篇文章，确立了自己的主张：在一个相当小的范围内，软件工程有效；对于相当大的其他范围，软件工程是否有效尚属未知；并且总有些问题无法用软件工程的方法解决，即"没有银弹"。Brooks 的论述，可算是第一次明确地为软件工程科学（假如软件工程的确是科学的话）划界，而不再期望软件工程能够解决一切问题。所以，这种态度可以被称为"科学的悲观"。

而在划界完成之后，DeMarco 似乎变得更加乐观了。1995 年，他推出了 *Why Does Software Cost So Much* 这本书；1997 年，《最后期限》；1999 年，《人件》；2002 年，*Slack*。从这几本书中，可以看到他的思维已经完全转到了软件工程科学之外的范畴。用他自己的话来概括就是：

> 优质管理的四大要素：
>
> - 选择正确的人。
> - 为他们分配正确的工作。
> - 保持他们得到激励。
> - 帮助团队凝聚起来并保持凝聚力。
>
> ----Tom DeMarco，《最后期限》，清华大学出版社，2003 年 1 月。

很明显，DeMarco 此时所关注的问题是：在系统的、可重复的过程之外，重视对人的关怀，并期望这类“自由”的管理方式能够解决一些问题，从而发现其中的科学原理。对于无法用软件工程的方法解决的问题，他回归到软件开发的最初形态——依靠个人的才华获得成功。这种探索，即使有一定的成果，也无法形成对别人的指导，但毕竟是一种有益的、对于软件哲学的探索。

从事这方面探索的，不仅仅是 Tom DeMarco。Pete McBreen 在他的 *Software Craftsmanship* 中也回归到工业革命之前的手工业时代，一改沿用多年的“工程”的比喻，而改用“工艺”来比喻软件开发，并提出了一整套发展软件工艺的办法。承认科学的范畴狭窄、承认科学并非万能之后，人们开始向科学之外的领域做大胆而谨慎的探索，并且保持冷静的乐观态度，不再把哲学的领悟妄自纳入科学范畴。这是对于科学和哲学的发展都有利的态度。自然科学的大发展，是从消除了“科学”二字头顶的光环之后开始的。软件科学如果能够有发展，必定是在消除了“软件工程能够解决一切问题”的幻想之后。

所以，当你捧起《最后期限》、《人件》、*Slack* 或者 *Software Craftsmanship* 这样一本书的时候，实在不应该期望它能够对你的项目有所帮助。达尔文的进化论作为一种形而上学指导时，能够让人对自然产生一种敬畏；但若把它当作一种科学理论抄起来随手乱舞，祸害也是无穷。对于形而上学的探索，读者只能期望它能够激起自己心灵的共鸣，而无法对它有现实的要求。或者可以换个角度来说：当你抛开现实的问题，忘记世俗的压力，用快乐的欣赏眼光去阅读它们时，它们或许能给你一些意想不到的收获。

到此，我想你也应该能对出版社的一些“不实宣传”表示谅解了。没有人能告诉你“这本书不好”或者别的什么东西，因为好与不好本来只是一个唯心的评判。而且，不能分清科学和非科学的人总是那么多，所以至今还有人认为熟读《射雕》就能学会降龙十八掌。作为腰包里钞票的主人，在希望别人帮助你之前，先给自己一双明亮的眼睛，然后给那些理智的乐观者和并不那么理智的乐观者一个宽容的微笑吧。

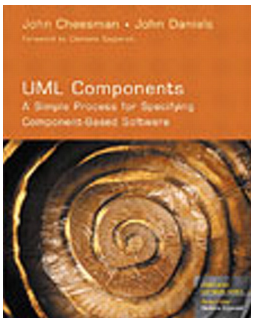
很快就要过年了。明年的软件工程该换上哪张符，我也不敢妄言。对于那些披上科学外衣的伪科学和那些戴上真理光环的废话，我仍然将给它最严厉的批驳。与此同时，也许我也会去看看那些“软件的哲学”。不过，在此之前，仍然保持着冷静的乐观态度，祝所有人新年快乐吧。

（抱着这样“无所求”的心情和快乐的欣赏眼光，圣诞夜去看了《英雄》，倒也觉得满愉快。虽然张艺谋仍然不懂如何拍出一部让人想看第二遍的电影，但在圣诞夜和三五好友一起花三个小时吃饭看电影，本身就是一件让我开心的事情了。）

透明

2002 年 12 月 26 日，北京

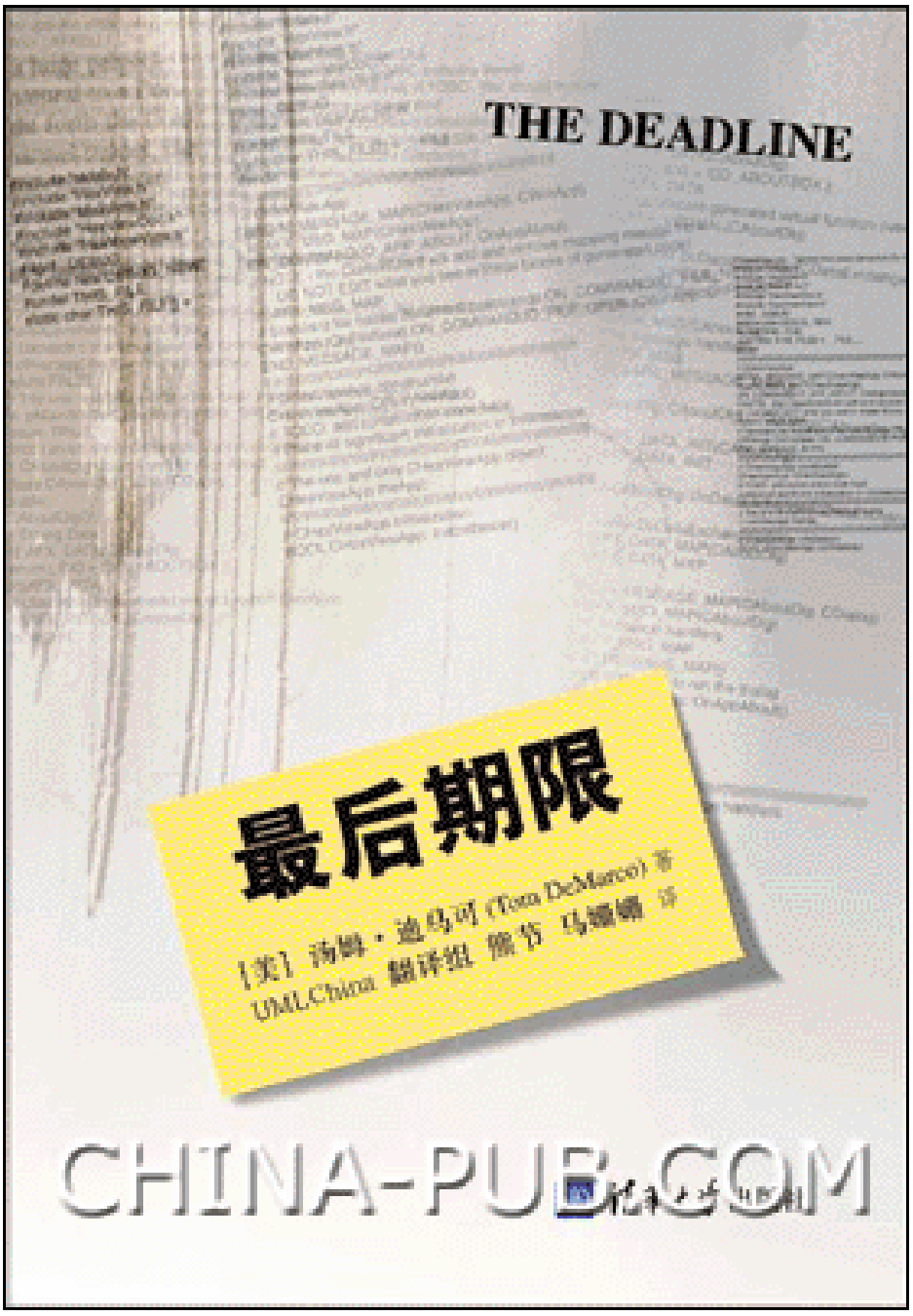
2003 年 1 月 UMLChina 专家交流预告

软件过程 1 月 16 日（星期四）上午 9:30-11:30	 Barry Boehm	嘉宾介绍>>
UML/J2EE/.NET 1/22 日（星期三）上午 10:30-12:30	 John Cheesman	嘉宾介绍>>
软件需求 1 月 28 日（星期二）上午 9:30-11:30	 Ellen Gottesdiener	嘉宾介绍>>

《最后期限》：一本软件开发小说

Huang Yin 编译

查看评论



John Scullery

“这是一本非常妙趣横生的管理书籍。《最后期限》是一个具有非凡创意并且有趣的故事，每个章节的最后都能基于团队的项目管理给出一些具备深刻见解的业务原则。”

关于本书

本书来自多产而有影响力的咨询顾问和作家 Tom DeMarco，是项目管理的小说，它生动阐明了影响软件开发团队生产率的原则，驳斥了完全荒谬的理论。

利用他在这种小说形式所释放的特有智慧，DeMarco 围绕六个软件产品的开发构造情节。Mr. Tompkins 是一个从某巨型电信公司精简出来的管理人员，他要将一个他所管理的大型开发团队分解为 18 个小组——每三个小组负责一个产品。这些小组规模不一，工作方法各异，相互竞争……并且要与不可能的最后期限进行抗争。通过这些小组，以及众多给予他很多帮助的资讯顾问，Mr. Tompkins 试验他在一个生命周期里所获取的项目管理原理。在这篇有趣的小说中，关键章节都用日志形式结尾，这些日志组成了本书中针对管理阐述启发性方法的核心。

Bruce Taylor

"Tom DeMarco 一再充满人性和洞察力地把管理的理念层层剥离， 这样的方式如同将软件项目和团队的管理转化为企业通用管理一样容易。在《最后期限》中，常让我们掩卷而笑， 其中富含大量日常工作生活中的荒唐事，以及通过度量设备有助于更好地管理和执行。 “是功能点，笨蛋!”（编注：和《人件评论集》中的“是人，笨蛋!”一样，都来自克林顿的竞选口号“是经济，笨蛋!”）

Dr. Patricia McQuaid

我曾经在该书 1997 年出版的时候拜读过，即被 DeMarco 所使用的将项目管理和软件质量连接起来的迷人方式所吸引。我可以领会其中对软件专业人员以及在这个领域热情工作着的人们的首肯。作为一个教授，我决定将其结合到我的软件质量管理班级的教学中，这个班主要是由来自加利福尼亚州立工艺大学管理信息系统系（商学院）和计算机科学系（工学院）的高年级的研究生所组成。

这本书虚构了一个项目经理 Tompkins 先生，他被绑架到一个岛上，在这里他可以实现他的梦想：对不同小组规模和具有不同专业水平的开发人员的进行实验。这些小组通过相互竞争来产生最好的产品，这样让他对自己的项目管理理论进行试验。当然，还有一个不可能的最后期限需要面对。除了围绕他的理论编造一个有趣的故事以外，他在每一章节的结尾还用条目形式将关键的原则记录在日记里。

与众不同的是，这本书的评论大部分来自于我的学生的注释。这个班级每年开一次，我第二年将这本书引入到这个班。我的学生阅读该书之后，在我们的课堂讨论前，进行了一次课堂闭卷测验，最后一个题目就是对《最后期限》进行评论。（为了澄清意图，我对学生的评论进行了轻微的修改并得到他们的允许将他们的注解加到评论中。）

这是一个对学生提出的问题：我想知道你们关于这本书对这堂课的价值的真实看法。这没有一个准确的答案，但是为了得到完全的信任，你需要对你的答案作出证明。（如果你不喜欢，要对你的这种反应做出证明。）以下是他们的回答。

“这本书很有价值，他给出了来自于大项目问题的真实情况。尽管这是一个虚构的故事，其中所提及的问题困扰着大多数的项目，而且没有得到很好的解决。DeMarco 作了一项很奇妙的工作来告诉读者如何解决他们。他涵盖了很多方面，例如团队的大小，争议的解决，设计，情绪，病态的见解，度量指标等等，以此教给读者项合理进行目最有效的方式。这本书教会我很多，而且读起来很有趣。”

“我相信这本书对于这堂课很有价值，它给出了许多在软件开发过程中通常会遇到的现实问题，以及事情会这样进行的原因。它揭示了成功项目的很多要素以及为什么项目通常都不能符合其最终期限。”

“我想这是一本好书。Tompkins 在日志中让我记住很多重要的东西。这本书让我知道一些重要的概念，也让我思考了很多。不只是阅读本书，在阅读的时候，我更想到如何在一些给定的环境下作出反应。”

“我真的很喜欢这本书，并且认为它极具价值。通用项目管理技能的知识非常有用，尤其对于这个班级而言。我计划保存这本书，并且将 Mr. Tompkins 的日志的每一章节中有价值的观点形成一个清晰简明的大纲。他对从风险管理到设计，到冲突的每一件事进行仔细检查。对于这个班级以及工作场所都是不可估量的。”

“这本书具有教育意义且充满趣味，我不时发现这个故事是令人兴奋的，而且用尽可能不枯燥的方式来表述管理原则（虽然有时是非常技术性的），谢谢！”

“在虚构和非虚构之间的明确交错使这本书更有趣。我最喜欢的就是日志（形式）。他总结了所有 DeMarco 试图解说的所有关键点，给出了大量有价值的观点并用一种很容易理解的方式来解说。”

“我认为这本书是很好的学习工具。我们读到的一些东西相当枯燥，这是一个好的改变步调。它给出了一些真实世界的问题，并展示了应该如何处理，且能从其中学到什么。尽管结尾有些牵强附会，但是我仍然喜欢这本书。”

“我真的很喜欢这本书。我可能还想再读一遍以完全吸收其中的概念，因为里面有如此多的教训值得学习。我喜欢他将我们所学的所有知识联系在一本书里：从人的问题和基本团队特性到风险管理。我认为这本书有很多东西可以传授，我也很有幸能有机会拜读此书。它也是一本易懂的读物。我喜欢的一个主要的观点就是，Tompkins 被恭维成事实上不是人们喜欢他，而是他喜欢人们。我希望能将这本书的概念结合到成为未来管理者的希望中。”

“这是一本浅显随意的读物，它指明许多我们所学的项目管理理念。我真的希望对于每一个科目都能有更多类似的书籍，也许我们就可以抛开那些令人厌烦的，枯燥的课本，用一种有乐趣的、令人愉快的方式学习。”

“我感觉这本书尽管有一点牵强附会，但是他把我们这个课程的大多数要点都作了一个精彩的总结。它将这些要点都放进现实环境当中，因此我们可以看到他们如何影响到一个真正的项目。它明确指出了这些概念如何工作，而不仅仅给出学生在演讲类型课堂中得到的理论。像这样在实际中看理论简直就是锦上添花。明智的选择！”

“我确实很喜欢这本书。于我而言，这绝不是作为一项家庭作业被强迫完成的。每天晚上睡觉前我都拿这本书随便翻翻，这是本有趣的书。我从中学到了很多关于该如何管理项目（的知识），但是我最喜欢的就是 DeMarco 做的更普遍的提示和论点。例如，每个人感到比其他人不够聪明的一种感觉就是对钱的态度。每天我都能在班上看到此类事情的发生，我知道并不是每个人都明白怎么回事，但是他们并不想说什么。我也喜欢发怒＝畏惧的观点。下一次我再看到愤怒的项目经理不再感到种胁迫。我想，在这样情况下，项目经理很多是因为畏惧而发泄在其手下身上。这本书是进行项目管理的一个创新的方式，DeMarco 改变了一个非常枯燥的科目。”

“我认识到这本书是我们这堂课最有意义的一部分。这本书是有趣的。但最重要的是，它对我们的学习有帮助。它运用了很多我以前听过的词并将他们都放在一个故事当中。它给予我一个与这些关键词组关联的方式，并且指出他们在业务以及项目管理中起到什么重要作用。它也是一个用于说明项目管理如何艰难的很好方式。我通常认为项目管理仅仅是需要画甘特图并组建一个团队，实际它包括比这更多的内容。项目管理是每个项目成功和失败的关键。这本书是将学生置身于所学的极好途径。”

“我认为这本书有一些关于项目管理真正优秀的观点以及一些实际的建议。最好的章节之一就是结尾关于处理愚笨的或是可怕的上层管理部分。它说，为了这里或那里的机会，等待奇迹出现的时机，但是不要指望这个。我认为这本书用这样让人感到好笑的方式来让他的观点得到理解真是棒极了。我认为这本书的真正的价值将在以后得到体现，当我们真正管理一个项目或者是至少参与一个项目的时候，这样的情况下我们可以打个响指说，啊，这个 DeMarco 曾经说过……”

Kathy E. Gill

从第一章可以清楚看到这不是本死板的书。DeMarco 在书前的内容简介中以不那么文绉绉的诗来开头（“我们的公司正在裁员，因此剩下的人会更富足”以及“ReSON 并为她而骄傲！”），它撇开了科学的管理工具继续前进并用对于软件技能管理的胜利来结尾。

DeMarco 轻描淡写地设置了一个剧情：从“大型的电话和通讯公司”裁掉的项目经理（Mr. Tompkins）被驱赶到一个以前的共产主义村庄里去充当完成一个雄心勃勃的计划的先锋：使摩拉维亚在 2000 年成为世界上第一个出口用收缩膜包装的软件的地区。

最幽默的是，这本书将 Mr. Tompkins 和几种业务原型的关联载入了编年史，从类似一个暴君的 CEO（Mr. Bill，一个非常富有的商人）到压力很大的商人的财务/计划经理（Belok）。或好或坏，作者所举的相信直觉的经理（贝琳达）的例子，保持一个女性的传统形象。

短小的章节，快速的阅读，大量的笑料，没有 Dilbert 和 serfriendly 的玩世不恭的教训，以及用于回顾的总结。其中还要推介的是，更多用于任何管理而不仅仅只是用于软件开发的原则实践。

实际上这里没有什么真正新的东西，至少对于一个学习过“软件资源”的人而言是如此（正如 DeMarco 很清楚指出的这是“硬件资源”）。从各种方面来说，这本书是《人月神话》的姐妹篇。例如，Mr. Tompins 测试“使人员服从于加速的进度”原理（有关于结论的猜测？）。就像 MMM 一样，《最后期限》被定位为一本技术书籍，同时包括了对于所有管理者的经验教训。包括对计划的重要性的倡导。

然而，《最后期限》专门关注于人际关系的问题：“管理中最难的是什么？”“人。” Tompkins 机械地回答。他清楚的知道他在这个项目中所处的位置。 “将合适的人放到合适的位置上去，这就是一个优秀的项目经理和糊涂的项目经理的区别。”

新颖的是 DeMarco 将幽默和叙事的方式用于宣讲好的管理概念（和回报）。你记得你什么时候能在阅读一本管理书籍时放声大笑吗？单独要提及的是，这本书对于任何管理者的藏书而言，都极具价值。

Jason Bennett

DeMarco 以一种有趣的方式从人间角度扩展了他的作品：寓言的小说。Tom Clancu 没有动摇他的观点，但是作为一个教学工具，这本小说真的很好。一些优秀的观点使得项目管理得到重视并解决了恶性管理的问题。

背景

再向各位致意。这个星期我们结合上个星期 DeMarco 对于项目管理的话题进行了艰苦跋涉。寓言以前在其他领域已经做得很好（我个人很满意《爱丽思梦游奇境》。）但我不认为以前在软件这个领域曾有这样的做法。在我将需要一个停顿下来，回忆以前我曾经消化过的另外一或两本书。现在，《最后期限》…..

这本书讲述了什么？

从技术角度而言，这是一本关于一个从 BT&T 裁员的经理 Webster Tompkins 的传奇故事。为了 NNL（国家高级领导）他被雇佣到摩拉维亚负责将这个地区变为软件生产地。DeMarco 编造了很多情节让 Tompkins 忍受软件项目开发中大多数的典型的问题。他有个非常好的大脑使他相信自己能够继续下去并找到新的解决方案。让人抱怨的是，由于几乎所有的这个地区的软件工程师都被召集在一起，其数量远远超过他们所需要的人数。因此他们在软件工程实验室为每个产品建立了多个团队来测试不同的理念。充斥全篇的都是差劲的老板，差劲的下属，员工管理问题，矩阵，有趣的新技术，各种复杂的问题还有大量的 DeMarco 个人的观点。

好处是什么？

幻想是一个有趣的观点，这样的叙述导致剧情的流畅。这本书里面有大量的情节，很多你几乎没有机会尝试的情节也被提及。为了避免读者在阅读的时候遗漏掉一些要点，DeMarco 还利用 Mr. Tompkins 的“日志”形式在每一个章节后面都做了要点的总结。如果仅仅因为 DeMarco 希望达到为那些人们对定论的信任进行一个总结，这本参考书对于很多有名的人来说都是很有用的。总的说来，这本书非常流畅，如果你是一个项目经理或是正在某个开发小组当中，你可以就这本书学习。通常，要记住好的管理的要点：“任用正确的人员，让他们到合适的岗位上去，不断鼓励他们，还要使整个团队凝聚在一起，并一直保持这样。所有的一切也都是按部就班的。”

坏处是什么？

不幸的是，幻想有时是会破灭的。DeMarco 不是一个非常好的传记作者（那些浪漫的章节写得都不是很好），虽然无可否认这些不是本书的要点。有时这本书里的情节有的就是根据那些 DeMarco 想要放置其中的经验教训来安排。我也并不赞成 DeMarco 的少数观点，（尤其是与 CMM 有关的），但是至少他往往有自己观点。

从中我们可以得到什么？

这本书是我以前曾经拜读过的两本书，DeMarco 的《人件》（编注：中译本将于 2003 年 3 月出版，由 UMLChina 方春旭翻译）和 McConnell 的《软件项目生存法则》的优秀姐妹篇。这本书相比较那两本书老练程度有所不同，当然，还有综合的体验。McConnell 关注进程，**《人件》更多关注雇员，《最后期限》关注从管理者角度出发的解决方案。**任何一个管理（或是参与管理！）一个软件项目的人，不管是商业的或是开放源代码的，都会遇到《最后期限》中讲述的问题。有备无患总是最好的。

Randy Rice

几年以前，我读过 Eliahu Golden 的目标和思想，“哇，如果有个人能够写一本象工程技术方面的《目标》那样的关于软件项目管理的小说来取代那些软件工程教科书，该有多好！”

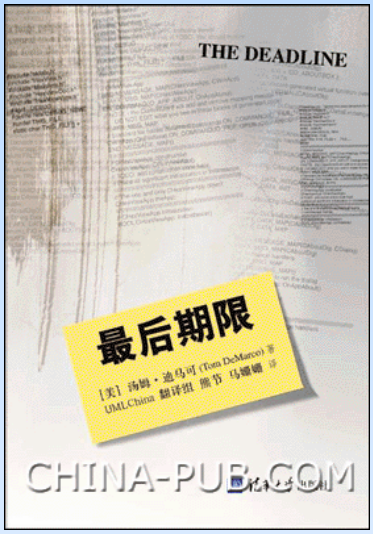
在《最后期限》中，梦想成真了。DeMarco 用一种幽默而有趣的方式为软件工程教学作了一项伟大的工作。实际上，有时我发现自己总是在没有认识到一个项目管理教训的价值之前学习它。

我发现一个这本小说的前提就是我在很多机会中曾经深思过。也就是说，如果一个外国的软件公司要开发一个先进的产品来与美国已有的市场领先者进行竞争的时候会发生什么？

这本小说就是一个外包项目经理的经历，Mr. Tompkins，他被绑架并得到一个唯一的机会去管理一个有最后期限的项目，并受制于一个暴君似的人物，Morovia 的老板。

为了确定这本书的要点不被遗漏，Tompkins 用一些日志来记录下他学习到的经验教训。读完这本独一无二的书，你会发现你将学习到许多项目管理的东西并乐在其中。

《最后期限》



Tom Demarco

翻译：UMLChina 翻译组 熊节

汤普金斯在飞机的座位上翻了一个身，把她的毛衣抓到脸上，贪婪地呼吸着它散发出的淡淡芬芳。文案，他对自己说。他试图回忆当他这样说时卡布福斯的表情。当时他惊讶得下巴都快掉下来了。是的，的确如此。文案……吃惊的卡布福斯……房间里的叹息声……汤普金斯大步走出教室……莱克莎重复那个词……汤普金斯重复那个词……两人微张的嘴唇触到了一起。再次重播。"文案。"他说道，转身，看着莱克莎，她微张的嘴唇，他……倒带，再次重播……

....

"我不想兜圈子，"汤普金斯看着面前的简报说，"实际上你们有一千五百名资格相当老的软件工程师。"

莱克莎点点头："这是最近的数字。他们都会在你的手下工作。"

"而且据你所说，他们都很优秀。"

"他们都通过了摩罗维亚软件工程学院的 CMM 2 级以上的认证。"

《最后期限》各章精选

第 01 章 新的机会

汤普金斯先生轻松地坐到了最后一排的座位上。这是新泽西，佩内洛普电话和通信公司的主会场。过去的几周里，为了参加各种演讲，他已经在这个礼堂里呆过很长的时间了。现在，汤普金斯先生以及其他几千位专家和中层管理员工都被解雇了。噢，他们通常不会用“解雇”这个词。他们更喜欢说“解决多余的人”或者“缩减企业”或者“调节企业配置”或者“精简”或者“减少管理”。或者，用他们最喜欢用的话来说，“离开到别处去寻找机会”。他们甚至给这种说法一个简写：ReSOE。汤普金斯就是一个 ReSOE——“离开到别处去寻找机会”的人。

第 02 章 对抗卡布福斯

有一个人在他的右边，一只有力的手扶着他的手臂，很温暖。还有一种淡淡的、让人舒服的气味。气味很淡，明显是一个女人身上的味道，有点象玫瑰或者是姜花。这种气味让他感到很满足，还有那种温暖的感觉。

……

“我们将制订出冈特图，”卡布福斯说道，“以及珀特图、状态报告、给人力资源部门提供的接口、每周会议的计划、电子邮件的使用规定、时间卡、进度跟踪记录、项目里程碑报告，还有——这是我们特别感兴趣的部分——制订一个质量管理程序。就是这样，各位还有什么问题吗？”

汤普金斯先生站了起来。“我有问题——我叫汤普金斯。我的问题是：这就是全部了吗？这就是整个日程安排？”

第 03 章 “硅谷”

他的衣服摆在梳妆台上，一条牛仔裤、一件棉布衬衣、内衣和内裤。他直盯着胡利安女士，给她最明显的暗示：他想要一点隐私。她调皮的笑了起来。汤普金斯抓起衣服，走进浴室，关了门，插上锁。

……

“我不想兜圈子，”汤普金斯看着面前的简报说，“实际上你们有一千五百名资格相当老的软件工程师。”

莱克莎点点头：“这是最近的数字。他们都会在你的手下工作。”

“而且据你所说，他们都很优秀。”

“他们都通过了摩罗维亚软件工程学院的 CMM 2 级以上的认证。”

“太厉害了！你们怎么做到的？你们从哪儿找到这么多高级工程师？我是说，在摩罗维亚这种弹丸小国，谁能想到……”

“亲爱的，你的偏见又开始作怪了。你其实是想说‘你们这样一个第三世界国家怎么会有这么强的技术实力’吧？”

第 04 章 CD-ROM 工厂

好了，在这一天中，他干了不少好事。他把一两个不可能完成的工程揽到了自己身上，启动了一些真正大型的系统开发计划。他觉得自己在这个领域里面也算是成就卓著了，以后在他死去的时候，应该会有一小段文字发表在《IEEE 软件》或《计算历史记录》上。不管登在哪里吧，他一想到末尾处会有一句“死于摩罗维亚的肉钩子上”这样的话，就觉得郁闷不已。

第 05 章 元首

“没问题。唔……”元首似乎有点走神。他懊悔的回头看着屏幕，好象是想找回刚才做的事情。现在汤普金斯可以看到屏幕了，上面似乎是一页程序代码。C++，他想。

一个声音从房间的后面传过来，莱恩小姐端来一盘软饮料和点心。元首稍微愉快了些。“噢，好。”他说道，然后拿起一块奶油蛋糕，塞进了嘴里。

第 06 章 世界上最伟大的项目经理

事实是：他从来没有管理过这么多的人。他曾经管理过的一个项目有两百五十人，其中大约有三十五个中层经理。但是这次是一千五百人！在摩罗维亚，要向他汇报工作的管理人员数目几乎就相当于他以前管理过的最大项目中的全部人数。而且他们还都是未知数。就象瓦尔多一直提醒他的，他需要立刻去安置职员、分配项目。元首已经安排了六个项目，以此开始摩罗维亚在软件世界中强大的新形象。六个项目，这不算太坏，但是汤普金斯

还是希望按照莱克莎的建议，让多个团队在不同的条件下做同样的工作，他们的项目管理实验室。假如在每个任务上面安排三个竞争团队，那就意味着他必须组成十八支项目团队，挑选十八名经理。

第 07 章 雇人

贝琳达穿着一件光艳照人的太阳裙出现在汤普金斯先生的办公室里，全身上下干净得闪闪发光。但是她仍然赤着脚，穿鞋是她唯一不肯做的改变。“鞋子？我再也不会穿。”当他问起的时候，她就这样回答。这样也就可以接受了，他想道。不管怎么说，她比他高了六英寸，尽管还光着脚。

“好吧，”他说，“我想我们可以开始了。”他从桌子旁边拉过一把椅子给她，指指桌上的一堆简历：“这就是我们要看的。”

……

她做了个鬼脸：“你真的看过那部电影吗，韦伯斯特？《巴顿将军》？你看过吗？还记得里面的情节吗？”

“当然了。”

“片子开始的那一幕，巴顿‘指挥’战役的那一幕，我们那位年轻的朋友所说的那一幕：呵呵，巴顿在那一幕里根本连一条命令都没下过。他只是从望远镜里看着整个战役。德军坦克分队象他预料的那样穿过山谷，而他就看着那些坦克。在那儿，在第一辆坦克上，站着一位军官，手里拿着马鞭。巴顿盯着他，说道：‘隆美尔，你这个高贵的杂种，我读过你的书。’他读过隆美尔的书，所以他很清楚隆美尔会怎么做。然后，战斗打响了。进攻，侧翼机动，佯装撤退，再次进攻，空中支援，后备队到达。巴顿只是看着，根本没有下达哪怕一条命令。”

第 08 章 大名鼎鼎的尼佐利博士

“但愿如此。不管怎么说，现在我和大名鼎鼎的尼佐利博士单独呆在一起。如果我不向您请求一些建议的话，我一定会疯掉的。告诉我，赫克特，我应该怎样做才能让这些项目有最大的可能获得成功？如果你在我的位置上，你会怎么做？如果只能做一件事？”

赫克特的目光越过楼梯，游离在远处：“一件事。这是个难题。”

“我是否应该关注过程改进？你知道，软件工程学院的人一直在试图说服我。他们告诉我：立刻执行一次过程改进计划，将整个团队从 CMM 2 级提高到 3 级，这就是我能对组织所做的最大帮助。我应该这么做吗？”

第 09 章 马可夫准将

“一万三千五百七十一人，由摩罗维亚陆军第一军、第二军和空军共同组建的团队。”准将立刻答道，“每年的预算是一亿九千一百万美元，资金总额是八亿五千三百万；六百八十八名军官（包括九位将军），三百六十二名支持人员；七万两千平方米的室内场地，超过一千一百平方公里的基地和研发中心；五百零九名技术人员，包括三百八十八名程序员、系统分析员和设计师。”

……

有七个人直接向他报告：马可夫准将和六个产品经理，这六个经理负责交付与 Quicken、QuarkXpress、Photoshop、Painter、Lotus Notes 和 PageMill 分别直接竞争的产品。在每个产品经理下面，都有一支 A 团队、一支 B 团队和一支 C 团队，这三个独立的单位将彼此竞争，力争以最快的速度制造出最好的产品。

第 10 章 阿布杜尔.贾米德

“那么，你们就可以一起研究这些模型，借此互相学习。没有这个模型，你就只有胃里面那一点点模糊的感觉，比如说，‘过快地增加人员会使项目变得效率低下’。这完全是主观的。你感觉到了，也许我或者贝琳达也感觉到了，但是我们没有办法讨论它。在某种意义上，也许贝琳达感到不舒服的程度是你的两倍，但是我们甚至不可能知道这一点，因为我们并没有量化这种胃里面的模糊感觉。而另一方面，在建造好一个直觉模型之后，我们就有了一种有说服力的理论，它清楚地表达出了人员增加对有用生产率的影响。”

第 11 章 可恶的贝洛克部长

汤普金斯先生坐在椅子上，做了一个深呼吸，全身上下感到舒服极了。他到摩罗维亚已经有三个多月了，这是他人生中最快乐的一段时光。现在看来，被公司解雇反倒成了最幸运的事情。呃，这种说法也不太对，无数被解雇的人没有象他这么好的境遇。他幸运的转折点不是被解雇，而是被莱克莎.胡利安绑架。

……

刚才他感到那么满意，这并不意味着就没有问题了。当然，问题总是有的。当你手下有一群天才的开发者时，就不可避免地会有些问题。在这一点上，摩罗维亚与他工作过的任何地方没有任何不同。优秀的人才了解自己的工作，并且努力让你知道他们了解自己的工作。有时候，他们会非常固执，而汤普金斯就不得不让出自己主角的

位置。但是在很久以前，他就学会了赞赏——至少试着赞赏——哪怕他们中最讨厌的人。

第 12 章 数字人

就在莱克莎突然消失后第三周的一个早上，汤普金斯先生在信件里发现了一张轻松活泼的明信片，是她寄来的。明信片的图案是一群学生围着一位街头音乐家，他几乎可以肯定那是在哈佛广场。在背面是她那娟秀的字体：

我亲爱的韦伯斯特：

我在这儿，在马萨诸塞偶然碰到了一个有趣的小公司。他们的职业就是度量，度量所有东西，特别是软件。他们有一种特殊的手段，可以完全从外部判断软件产品的大小，度量的结果用他们所说的“功能点”来表示。当然，我立刻就想到了你。我已经与他们的顾问 T. 约翰. 卡波诺斯先生联系过了，他会来拜访你。

祝一切都好。

莱克莎

第 13 章 QUICKERSTILL，再快一些

“呵呵，你知道，背离标准过程的人总是有好理由的。去年，软件工程学院把你的团队评定为 CMM 2 级，这就是说，你的过程和每个项目中人员遵守的规程都是可重复的。这就是 CMM 2 级：可重复级。不管你的方法是好是坏，是完美还是不完美 ，起码你每次都使用同样的方法。现在，爱德里沃利一号楼里大多数的人都在这样做。我们看了这六个项目，六个项目都还处在需求分析阶段，其中五个都在按照一贯的标准过程进行需求获取和文档化，只有 QuickStill 项目除外。格罗斯先生似乎完全抛弃了可重复的过程。他的团队还没结束文档工作就终止了需求分析，直接进入了设计阶段！”他说话的口气，好象这是违反人道的犯罪行为一样。

“我敢肯定他有自己的理由。”汤普金斯先生又重复了一遍。

“我的确有。”格罗斯说道，“我有非常充分的理由。你们看，这个项目跟我们以前做的项目根本不一样。这个项目是要模仿一个理解得很充分、文档做得很到位的商业产品，Quicken。我们有那个产品所有的文档，根本没必要再去建立一份需求规格……”

“没必要建立需求规格！”检查组长气急败坏地叫道，“我从来没有见过哪个项目不需要建立需求规格的。永远不会有。每个项目都必须建立良好、全面的需求文档。每个通过软件工程学院 CMM 2 级认证的项目都必须使

用与以前一样的方法和符号来建立需求文档。这正是拥有可重复过程的意义。”

第 14 章 摩罗维亚的第一个程序员

“是的，但是不花时间调试就意味着我们需要……”

“没有错误。对，你说对了，你学得蛮快的。”

“没有错误！”

“这是你说的。”

“我们怎么可能没有错误呢？！？”

“你看，假如你刚刚在某个模块中找到一个错误，它应该在哪儿？”

“在模块内部。”

“不，它应该在模块的边界上，在最边缘的地方。噢，当然，模块内部也会有一些很简单的错误，它们只影响这一个模块，在检查的时候，这些错误都很容易找到。真正的错误，会浪费你大量时间的错误，是那些与模块和系统其他部分之间的接口有关的错误。”

……

“当然，但是他们不是在设计阶段做。在设计阶段，团队只拿出一份文档。他们有一些空洞的‘哲学’，可能有一两份文件上的设计，然后复审只是走走形式。他们做这些只是为了应付管理者，让他们可以开始编码。最后，经理说‘好，你们可以进入下一阶段了’，团队就会欢呼，把所谓的设计束之高阁，再也不去管它。这种设计完全是废物。

第 15 章 快点思考！

“感谢上帝。现在，我看到你已经很好地完成了过程改进程序，我希望你保持下去。现在，我把目标改成：到年底的时候达到 4 级。然后到明年，我要……”

“请原谅，阿莱尔，你知道 4 级的条件吗？我是说，这要求员工获得特定的技能，你有把握吗？”

第 16 章 筹备夏季运动会

“呃，我想可以从输入操作得到。如果是这样，我们可以假设系统中有一个命令行输入设备；或者也可能在初始化的时候，跟着软件一起加载进来；或者也可能从上位系统中传过来；或者还可能由软件来检查硬件连接情况，然后自己构造出配置数据库。”

“对。按你的说法，有四种可能性。文档中描述的，可能是四种完全不同的系统中的一种，这取决于它的选择。但是它什么都没选，文档根本就没有说明这些数据到底从哪儿来。或者，让我们再看看这个：哪些配置数据必须包含在数据库中？系统可以重新配置吗？重新配置的规则是什么？怎么分配无线电频率？怎么改变无线电频率？消息采用什么交换方式？有没有多接收方连接？……文档中都没有说明。”

“它什么都没说。”格列佛点点头，“她是对的，韦伯斯特。这份规范细则什么都没有规定。这就是三百页含糊不清的废话。”

第 17 章 解决冲突的权威

“是啊。”贝琳达笑着说，“完全没有，因为冲突到处都是。在我们和贝洛克之间，在我们和软件工程学院之间，在我们和某些团队之间，在团队与团队之间，在团队内部，到处都有冲突。而且这还只是在爱德里沃利，在这个小小的校园里。韦伯斯特和我认为，联邦航空局的项目，我们的空中交通控制系统规格文档所出的这些项目，在所有的层面上都充满了冲突。”

“我们的业务中到处都是冲突。”汤普金斯继续说道，“如果不面对一些严重的冲突，你就根本无法建立任何规模的系统。系统开发中总有很多的分工，所以会有很多的冲突。我们的业务中到处都是冲突，但是我们对冲突的掌握却几乎为零。”

第 18 章 麦斯特罗.迪耶尼亚尔

“当一个小孩摔破了膝盖的时候，他的妈妈会怎么做？一个吻会让孩子感觉好些，然后她会把孩子的注意力转移到伤口之外的其他东西上面。在孩子意识到之前，伤口其实已经好了，然后他就把这回事全忘记了。”

“比如说，她会讲个故事来转移孩子的注意力？”

“也许就是，或者别的什么办法。但是，别忘了那个吻，妈妈那特别的吻，就在伤口上。”

“你说，在我们的业务中，那个吻应该是什么？”

第 19 章 部分和整体

“F 的意思通常是指：项目制造出了行政性的文档，并把它叫做设计。一般来说，这些文档只是用文字描述了对软件内部结构的一些早期设想而已。”

“用你的话来说，不是真正的设计。”

“对。当然，以后会有一个设计出现，作为编码的副产品。但是被叫做‘设计’的行为却没有制造出真正的设计，所以只能得到一个 F。”

“嗯。所有的小团队都得了 A 和 B，大团队则都得了 F，这是怎么回事？”

“我还想问你呢。这是个难题。”

“首先，我注意到：如果没有一个好的设计，那么‘先贤’所说的‘最后一分钟实现’就没有可能。”

“非常好，韦伯斯特，你开始进入状态了。”

“我还不敢肯定为什么 A 团队都这么惨。我可以肯定的是：他们根本不可能象我们希望的那样进行编码之前的设计。”

“完全正确。实际上，他们根本不会实施‘最后一分钟实现’。这六个团队都已经开始编码很久了，我没能说服他们推迟实现。我试过了，但是没有成功。”

第 20 章 在典礼上

这天早上，走过接待处的时候，汤普金斯先生发现传真机正在接收一份传真。他看看已经打印出来的第一页，看见开头处写着“我亲爱的韦伯斯特”。他的心“砰”地一跳：只有一个人会象这样称呼他。他给自己倒了一杯咖啡，整份传真也打印好了。把传真拿在手上，他走进自己的办公室，关上了门。也许，传真里会说，她就快回家了。

第 21 章 决战开始

悄悄地，她回来了，就象当初她悄悄地走。她办公室的门，那扇紧闭了快十个月的门，再次打开了。她就在里面，坐在窗边常坐的那张安乐椅上，静静地望着窗外的雨。

他的第一个念头就是……噢，他无法肯定第一个念头到底是什么。紧接着，他的第二个念头冒了出来，那就是要指责她。“见鬼，你到底跑到哪儿去了？”他的声音比原来打算的还要响。

她抬起头来，羞涩地微笑着：“韦伯斯特。”

……

“建构计划给我们一种一切尽在掌握中的感觉。”她继续说道，“这儿，来看这个。”她带着他走进 QuickerStill-C 的作战室，让他看墙上的一幅彩色图表。“一开始，我们计划在 60 次建构之后发布产品。每次建构都是整体的一部分，都在前一次的基础上添加新的特性。你看，今天得到的是第 24 号构件。从这张工作表上就可以看出，24 号构件有 409 个模块。上周完成的 23 号构件……”她找到了与 23 号构件相关的工作表，“……有 392 个模块。所以，在这次新建构中，我们增加了 17 个模块。还有这儿，你看这些模块的编号和这 17 个模块的规模。”

第 22 章 年度最热的首募日

五月二十四日，QuickerStill-B 团队交付了他们的产品。五月二十九日，QuickerStill-C 也交付了产品。五月三十日，PMill-C 交付。

汤普金斯先生欣喜若狂：“你能相信吗，贝琳达？我们竟然成功交付了三个产品，在贝洛克定的那个白痴一样的日期之前。”

……..

也许他真的能做到，只要有足够的时间。汤普金斯开始第二次考虑自己的建议。噢，天啊，也许他应该闭上嘴的。对于他的国家来说，一次融资收购也许并不是最好的事情。汤普金斯先生还是有一颗爱国之心的：“唔，如果你真的得到了美国，你不会干涉公民的人权吧？”

“噢，天，当然不会。我只管做生意。”

“也不干涉政府？”

“不会太多。难道我连做一些小改动的权利也没有吗？”

“呵呵，我想没问题。你想干什么？”

“我想把反对派赶到阿拉斯加的诺姆城去。”

“啊。呵呵，我想美国人民能够接受这个。”

第 23 章 经过里加回家

（主人公买下了保加利亚……）

他停了一会儿，盯着她说：“请等一分钟。”这一次，如果得不到他想要的，他绝不再让步。他换上最严厉的表情：“告诉我，莱克莎，你给自己选择的角色是什么？”

第一次，她显得不自信了。她咬着嘴唇：“我……”

“告诉我，莱克莎。”

她低头看着地板。“我还没想过。”她轻声说道。

“好吧，那就让我来告诉你。我给你准备了一个角色，你必须接受——否则我转身就走。”

她还是不看他：“你给我什么角色，韦伯斯特？”

他让自己平静一下，然后说道：“元首助理和夫人。”

“噢，韦伯斯特，我还以为你永远不会问呢。”

“嗯？”

“我同意，我愿意，我接受。”

“好。”他四下看看，“那么，我们现在该去哪儿？”

“王室套间。”她说道，朝着一扇华美的门点头示意，“就在那边。”

他把她抱起来，朝他们的新生活走去。