国外计算机科学经典教材

SOFTWARE QUALITY AND SOFTWARE TESTING IN INTERNET TIMES

软件质量和软件测试

(德) Dirk Huberty 等著
马 博 赵云龙 译

清华大学出版社

国外计算机科学经典教材

软件质量和软件测试

(德) Dirk Huberty 等著
马 博 赵云龙 译

清华大学出版社
北 京

内 容 简 介

软件质量和软件测试在过去的几十年中一直是软件开发的重要话题，而 Internet 时代的到来又给软件质量和软件测试带来了新的要求和挑战。

本书收集了这个领域知名专家的多篇论文，讨论了他们在大型组织的 IT 项目中的不同实践和经验。本书的第 I 部分讲述了项目管理，第 II 部分介绍极限编程和为 Internet 应用程序设计的过程，第 III 部分讨论从用户的角度进行的测试，第 IV 部分阐明了技术性测试的挑战，第 V 部分介绍了测试自动化的经验。

本书集中讲述了在软件质量管理方面的挑战和实用技术与经验，适合对此有兴趣的读者阅读。

EISBN：3-540-42632-9

Translation from the English language edition:

Software Quality and Software Testing in Internet Times edited by Dirk Meyerhoff,

Begona Laibarra, Rob van der Pouw Kraan, and Alan Wallet.

图书在版编目(CIP)数据

软件质量和软件测试/(德)胡伯蒂(Dirk Huberty)等著；马博，赵云龙译. —北京：清华大学出版社，2003

书名原文：Software Quality and Software Testing in Internet Times

ISBN 7-302-07444-5

I. 软… II. ①胡… ②马… ③赵… III. ①软件质量②软件-测试 IV. TP311.5

中国版本图书馆 CIP 数据核字(2003)第 094371 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：曹康

文稿编辑：侯彧

封面设计：康博

版式设计：康博

印 刷 者：北京昌平环球印刷厂

装 订 者：三河市化甲屯小学装订二厂

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印张**：16.5　**字数**：342 千字

版　　次：2003 年 11 月第 1 版　2003 年 11 月第 1 次印刷

书　　号：ISBN 7-302-07444-5/TP・5500

印　　数：1～4000

定　　价：30.00 元

前　言

Internet 时代系统开发的回顾和展望

摘要：

本篇论文对 Internet 时代系统开发的重要方面进行了回顾，包括质量保证和测试。系统开发和 Internet 这两个领域在发生着巨大的变化，且紧密地相互依赖。由于 Internet 的出现，系统开发面临着新的挑战，特别是新的响应时间和质量要求。像面向对象、面向组件和代理、应用程序挖掘、开放式系统开发和新的轻便开发模型，这些方法都必须证明它们适合于解决新型的问题。本文介绍了在此环境中出现的两个重要的技术发展，即对等式计算与所谓的 Web 服务的实现，并且分析了它们对系统开发潜在的影响。本文在结束时探讨了软件质量和测试由此而受到的影响。

关键字：系统开发、Internet、面向对象、基于组件的软件工程、基于代理的系统、重用、应用程序挖掘、开放式系统开发、极限编程、对等式计算、Web 服务、.NET、软件质量、测试

0.1　介绍

系统开发和 Internet 的发展在很多方面是相互依赖的。本文将重点放在了 Internet 的发展给系统开发带来的要求、机会和某些特殊的问题上。

首先，很明显地，像其他任何应用程序一样，我们必须开发 Internet 应用程序或基于 Internet 的应用程序。即使在 Web 环境下，系统开发也经常是“随意的”，没有任何正式的方法，较大的、尤其是战略性的应用程序需要一个支持过程和保证质量的可靠的方法基础。对 Internet 应用程序的新要求需要系统开发研究，例如在方法应用、上市时间、质量特征等领域中[参见例如[HOWC00],[BAUE00]]。把新的 Internet 应用程序集成在现存的 IT 解决方案和基础结构中，这些观念我们不应忽略。

其次，Internet 本身可以启动新的系统开发方法。在此，特别重要的是关于开放源代码软件和免费软件开发的讨论，这个讨论是在一些特别成功的项目之后而引起的，像免费的操作系统 Linux 的开发和开放式 Web 服务器 Apache 的开发。主要问题是这种类型的项目是否基于一种可推广的软件开发模型。

再次，系统开发是一门现有的学科，有或没有 Internet 的需求，它都是在逐步发展

的。然而，Web 几乎总是和系统开发环境相关的一个变数，至少是有间接的影响。系统开发的相对新的“轻便模型”(特别是一些原型开发方法和极限编程[BECK99])都提供了很理想的 Web 环境(可见本书中的第 5 篇论文)。

最后，系统开发和 Internet 这两个领域都受到科技发展的影响，而且它们的发展是相互促进的。Internet 是一个快速发展的环境，几乎每天都有新的技术产生，任何支持 Internet 应用程序开发的方法都必须随着 Web 而发展。在近期有两个最重要的科技进步(两个都没有完成)，一个是所谓的 Web 服务的平台概念(特别是 Microsoft 公司的.NET 平台和 Sun Microsystems 公司的 ONE——Open NET Enviroment)，另外一个就是对等式计算的复兴。

0.1.1 Internet 的发展

自从 20 世纪 90 年代开始，Internet 就已经飞速发展，并且正成为越来越多的人检索信息、通信交流和处理事务的媒介。尽管最近面向商务的 Internet 热情有所减退，但毫无疑问的是，Internet 在社会和经济上的重要性是在不断增长的。

在 Web 的早期，主要是被动地使用静态信息，而现在的焦点是基于 Web 的事务处理，例如座位预订、票务、购物等。Web 越来越成为应用程序开发的平台[CLAR01,p.16]。在这种环境中的时髦用语是客户关系管理、供应链管理或提供应用服务(见 0.3.2 节)。用户看到的一切都是用户界面和新应用程序的易用功能——一般是通过他们的浏览器窗口。但看一看幕后，特别是从系统开发的角度来看，事情并不总是展现出积极的方面。

在万维网开始后的一个短暂时期，静态 HTML 页面足以令人满意。因此，不存在正式的开发方法或质量保证方面的问题。但是，动态生成 Web 内容的观念一出现，就出现了对用户输入、数据库访问和编程逻辑的需求，这些或者是基于客户端的(在 Web 浏览器中)，或者是基于服务器端的。

如今，传统的应用程序与 Internet 应用程序之间的界线正在变得模糊。对 Internet 应用程序处理逻辑的要求至少与其他任何应用程序的要求一样重要——也许甚至更加重要，因为电子商务时代的系统复杂性再次提高了。Internet 应用程序必须能够随时(时间方面)随地(地区分布方面)被数量不明确的用户访问，并且对于质量不能有妥协的余地，特别是在安全性和可靠性方面——只要想想一个出错的在线银行应用程序所产生的后果！只要编程受限于图形用户界面(graphical user interface，简称 GUI)和 CGI 脚本，质量就不那么重要了，因为不管怎样，代码都会很快地过时[FISC01,p.65]。但是，一旦通过 Web 重新实现或提供具有战略重要性的应用程序，定义一个质量标准就变得绝对必要了。

遗憾的是，近期 Internet 和电子商务的大肆宣传导致了大量的电子污染：完成一半

的 C++、Java、JavaScript、VBS、Perl 和其他代码堆积在一起。对 Web 应用程序的大量需求迫使公司开发适合的应用程序。但是，有限的合格的软件开发人员或不愿高价聘请软件工程师使达成妥协成为必要。通常情况下，没有足够的时间进行详细的分析和考虑体系结构的概念。结果反应出了上述情况。

由于开发人员总是面对关于 Internet 的新技术、新的编程语言(Java、C#、Ruby、VB.NET)、通信协议、中间件结构(COM、DCOM、COM+、CORBA)或系统开发的整个构架(Microsoft .NET、Sun ONE)，以及其他方面，所以问题就变得更加严重。所有这些就其开发周期和未来的存在性而言都是不可预知的。如今，在 Java 和 XML 之后，Web 服务又成了全世界开发人员追求的新潮流。科技能否对 Internet 的发展产生巨大的影响，这个问题仍然悬而未决。在 Internet 环境下，每个开发项目都表现出高度的不确定性。

0.1.2　系统开发的挑战

在系统开发领域，实践者和研究者每天都面临着新的挑战。就像前面提到的那样，这些挑战大部分是由于 Internet 环境的发展和电子商务的引入，以及近期的移动业务带来的：

- 日益提高的重要性

如今，信息技术在组织和社会中表现出更高的重要性。系统不仅仅用于使已相当确定的任务和过程自动化，还用于重新设计业务过程、改造组织，甚至产生全新的企业和行业[YU01,p.124]。

- 日益增长的复杂性

系统和软件项目开发的复杂性已经提高，并且由于多方面原因仍在提高。用于开发过程的工具和技术的数量也在增加——很多时候它们是互不兼容的。例如，除了新的语言(像 Java、VB.NET 或 C#)，传统的编程语言(像 COBOL、C 和 C++)也一直在被使用。众多的可用平台为系统体系结构带来了更多的设计选择。这样提高了集成异构系统(术语是 EAI，Enterprise Application Integration，企业应用集成)的重要性，和管理与它们相关的接口的重要性。最后也是很重要的一点是，由于新的和额外的应用领域，以及解决更加复杂的问题，所以系统的复杂性正在不断地提高。复杂的任务需要复杂的和更加全面的软件[BALZ98.P.29]。

- 日益增长的质量要求

因为 IT 系统应用于比以前更加重要和关键的领域，所以这样的系统出现故障将会导致严重的经济和物质损失，甚至危害到健康和生命[BALZ98,p.30]。因此，IT 系统质量的重要性有所提高。特别是像安全性(访问控制、身份验证、加密)和可靠性(可用性、故障自动防护、正确性)这样的质量特性，必须是 Web 服务环境的重要

组成部分。

- 需求积压和瓶颈

对更加灵活、可适应的、可扩展的和健壮的基于 Web 的企业应用系统的需求加快了[GRIS01,p.37]。大多数公司正感觉到更快发布软件的压力，不管它们在市场中所处的位置如何。对于那些与 Internet 无关的公司来说，在过去的 10 年中的开发周期一般为 24 到 36 个月，现在已经压缩到 12 或 18 个月。对那些涉及电子商务和 Internet 环境的公司来说，这个周期可能要短到 3 至 6 个月[BASK01,p.51]。直到最近，一个主要问题是缺少优秀的开发人员，这不可避免地导致了需求的积压。由于很多 dot-com 公司的倒闭，所以现在有更多的开发人员可用。但是从长远来看，最近很多开发人员的跳槽能否成为很多公司和组织解决开发资源有限的方法，仍然令人怀疑。一方面，研究表明只有 30%到 40%的软件开发人员拥有足够的技术来开发成功的应用程序[CLAR01,p.17]。另外一个方面，技术更新的速度仍然在加快，特别是在 Internet 领域，这导致了对掌握最新知识的专家的更大需求。而且，当前软件和服务行业的衰落不会持续很长时间，这是因为高科技产业仍然是经济增长的驱动力。

- 承担传统系统

在最近的 30 年里，开发人员编写了数以百万计的程序。现在的大部分生产系统已经用了很多年，并且是用 Assembler、FORTRAN、COBOL 或 C 言语编写的。随着系统应用程序环境的变化，系统必须要升级、修改或重新开发[STAH99,p.326]。软件开发人员和程序员的三分之二以上的活动都在进行维护(故障修复和优化)和服务(改动和扩展)已有的系统[BALZ98,p.969f]。如今，很多公司为了便于从 Internet 访问，都在努力开放已有的应用系统。

0.2 系统开发中解决问题的方法

系统开发作为一门学科所面临的任务是找到解决上述问题的方法。以下各个问题之间存在着矛盾：

- 对应用系统的过量需求，特别是基于 Internet 的应用程序
- 应用程序开发的巨大时间压力(上市时间)
- 日益增长的质量要求
- 有限的开发资源

解决的途径是什么？

在大多数情况下，有限的开发资源从总体上来说是一个无法改变的瓶颈，至少在短期内是如此。有限的资源对特定问题的其他方面来说是个限制性因素。这个表现可以

用一句话表述为“目前资源决定要求”[MELL01,p.4]。

短时间内上市和必需的质量要求这两个目标之间存在的冲突如何解决？

在美国，研究表明在以 Internet 为中心的开发项目中时间因素是首位的：时间是主要的驱动力，相比较而言，时间使得诸如性能、成本，甚至质量的优先程度都降低了[Baskerville et al.2001]。在由此生成的框架中，有不同的解决策略可以缩短开发时间，与此同时，而不会低于一定的质量标准：

(1) 在系统开发中上升到更高级别的抽象，如通过应用强大的开发工具和程序生成器。

(2) 应用面向对象的方法或甚至是基于组件的开发方法，以确保所开发系统的模块化和重用性。

(3) 使用各种方法，以重用现存的、创建好的业务逻辑，例如应用程序挖掘。

(4) 引入社会的和组织的方法，像轻便模型的应用，例如极限编程，或是开放式系统开发的分布式形式。

(5) 在没有任何内部开发的情况下解决问题。取而代之的是，修改一个标准的软件包或者委托一个专业服务公司(外包)。当然，核心问题仍然存在，仅仅是从一个公司转移到另外一个公司。

0.2.1　从面向对象到基于代理的系统

应用面向对象的方法已经逐渐成为系统开发的标准[BALZ98,p.328f.]。甚至传统的面向过程的编程语言(像 COBOL 或 Pascal)，现在也已经增强了面向对象的特性，还有那些经常用于 Web 应用程序编程的脚本语言(像 Perl、PHP 或 Ruby)现在也支持面向对象[WYSS01,p.78]。

面向对象的随后应用带来了更好的模块化，因此有更强的灵活性和可维护性，以及更好的程序模块重用性[FISC01,p.62]。由于软件项目变得更加复杂，所以这些优点很重要。构建模块化软件结构的下一个更高级别是通过面向组件来完成。

基于组件的软件工程(component-based software engineering，简称 CBSE)为构建基于 Web 的企业应用系统提供了一个诱人的选择。它的工作原理是从选定的可重用的软件组件中开发和发展软件，然后把它们集成在合适的软件体系结构中[GRIS01,p.37]。组件是独立部署单位，通过良好定义的接口与环境交互，同时封装它的实现[GRIS01,p.37]。

在这个方法中，不是进行逐句的编程，而是可以把软件开发描述成一些可重用组件的编译。对于由此生成的系统的质量至关重要的是底层的体系结构的概念，在开发过程的开始就应该对这个概念达成一致。理想情况下，每个组件的质量已经提前分别测试过了。

在使用基于组件的软件工程的方法中，我们希望能达到以下目标，这些目标尤其在 Internet 环境中是很重要的[GRIS01,p.37]:

(1) 通过从一系列可重用的组件中集成应用系统，而不是从头进行编程，以减少开发成本和上市时间。

(2) 应用系统可靠性的加强，因为每个可重用的组件都经历了几个审查和检查阶段。

(3) 通过用高质量的组件代替已有的低质量组件，使应用系统的可维护性得到了提高。

(4) 整体的应用系统质量的提高，因为每个组件和所有组件结合生成的产品，都可以被每个专家在他们各自的领域内开发出来。

沿着这个方法，下一个发展阶段是面向代理的软件开发[YU01,p.123]。虽然在上面提到的方法中的组件有能力封装大量的功能，但它们原则上不是智能的，而必须依靠某些中心部件的智能控制。

相反地，代理是智能的、自治的、主动的、与环境和其他代理动态交互的软件组件。因此它提供了比传统组件更强的灵活性和可适应性[YU01,p.123]，[GRIS01,p.38]。中央控制在这个环境中已经被每个代理的分散的自我控制所替代。

由于这个原因，代理可以被看作是下一代组件，并且面向代理的软件工程可看作是传统 CBSE 的扩展[GRIS01,p.38]。基于代理的系统提供了大量的灵活性并保证在处理由此产生的软件的发展、分发和复杂性时提供更高的效率。软件工程的这个风格看上去很有发展前景，尤其是对于以 Internet 的速度开发和维护基于 Web 的企业系统而言[GRIS0I，P.43]。

0.2.2 应用程序挖掘

为了响应对基于 Internet 的解决方案的更多需求，不同的系统必须互相连接。由于从一个技术或接口生成到下一个的变化发生得越来越快，所以这是一个很大的挑战。IT 经理必须问问他们自己，是否随着每项技术的改变都需要进行新的开发，或者基本目标是否应该通过新的技术使已有的过程可以使用。

与前一节描述的方法相比较，已有的软件系统通常没有为可能的重用进行设计。然而，很多遗留的系统包含了已确定的业务过程，这些业务过程也可以与新的技术一起使用。但是，为了能够重用这些系统的某些部分，已有的软件必须是可重用的[BALZ98-a,p.639]。如果不是这样的话，那么我们必须首先设法标识和增强软件的可重用组件。应用的方法是反向工程、二次工程和二次开发[BALZ98-a,p.640]。

对此环境下要执行的任务来说，一个合适的术语是应用程序挖掘(图 0-1)。它的目标是从已有的和成功的应用程序中提取业务逻辑，封装这个逻辑并使之在组件体系结构中可用(0.2.1 节)。实现应用程序挖掘的几个步骤是[MOOR00,p.1]:

(1) 用分析工具查找有价值的代码段，例如查找某个算术命令序列。目标是分析系统以揭示关键的业务过程和相互依赖性。

(2) 从已有系统中提取相关代码段，这是为了把它们封装在独立的模块中。目标是

得到关键任务的业务逻辑，以使它能够在新的开发中使用。

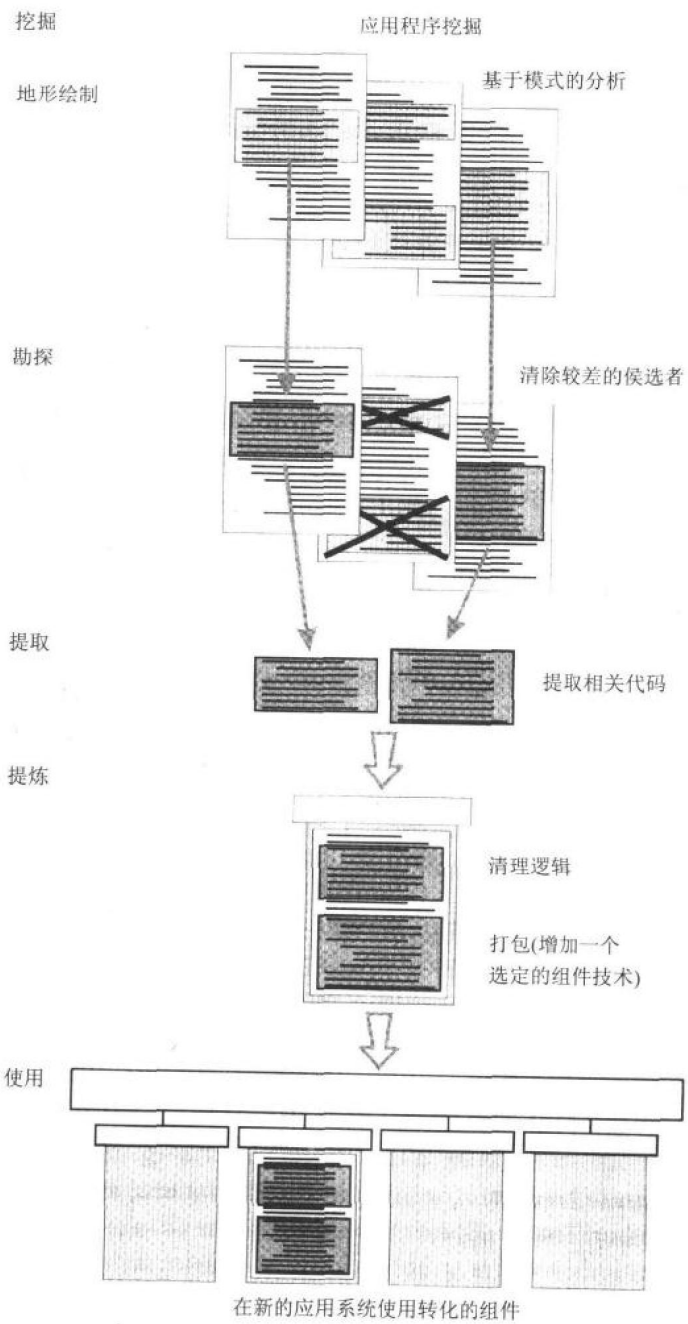

图 0-1　应用程序挖掘的过程

(3) 用选取的组件技术，通常是 CORBA、DCOM/Com+ 或 EJB，补足所生成的业务模块。

最后，转换后的组件(就像新开发的组件)将用于新的应用系统中，目前这个应用系

统通常基于面向 Internet 的体系结构。

如果应用正确的话，应用程序挖掘能够帮助解决缩短上市时间和必要的质量等级这两个目标之间的冲突。我们不应该低估为了从遗留系统获得可重用的业务组件而做的必要努力。但由此生成的组件的质量等级证明了这个努力是值得的。

0.2.3 开放式系统开发

开放源代码软件(例如操作系统 Linux 或 Apache Web 服务器)作为开发和发布软件的一个选择，在几年内已引起了大量的关注[LJUN00,p.50]。软件工程的基本风格看上去是用低成本生成高质量软件的成功方法，而且勿需质疑的是它对商界产生了很大的影响。一个重要的事实是，大型的 IT 厂商(像 IBM 和 HP)已经转向开放源代码软件。

开放式系统开发的根源部分在于 60 年代初期的黑客文化[LJUN00,p.501]，另一部分在于美国政府投资的研究活动，这导致了可为公众使用的结果，也就是开放的源代码，第三个方面在于 1984 年启动的 Richard Stallmann 的 GNU 项目。它可以被看作是免费软件开发的先驱[STAL01]，[LJUN00,p.502]。

Stallmann 提出了公共利益的概念来证明他的项目。确实，尤其是软件必须至少部分地与公共利益标准相一致[SAMU98]。这样，就不会有使用非法软件的广泛问题了：

(1) 消费中的非竞争：优秀软件的质量不会受日益增长的用户数量的影响(例外是 IT 基本结构中的瓶颈)——非常频繁的是大量用户将会提高总体的质量，例如交换或电子市场。

(2) 不能限制用户的消费(无排它性)：软件的分发(特别在 Internet 时代)实际上是容易的。限制人们使用非法软件的努力(例如，通过道尔芯片、版权保护或需要注册)经常是徒劳的。

在早期，Internet 已经在开放式系统开发中起了重要作用，并且因此可以被认为是这个方法的启动者[LJUN00,p.503]："在 Internet 普及之前，Internet 和电子邮件已经被免费软件社区使用了很多年。"它为致力于分布式项目的开发者提供通信和交换的平台。一些作者甚至声称，作为基于许多独立的、与位置无关的合作者的自愿加入的一个分散过程，在 Internet 时代开放式系统开发是优于其他任何组织结构的。

问题是这是否能作为系统开发的模式而得到推广。

以纯粹的形式，开放式软件开发的模型当然仅适于某些领域和某些项目。一些研究者举出对安全敏感的系统作为例子。我们也许会质疑积累名誉是否是有经验的开发者投身于开放源代码项目的充分动机。"在这个方面，典型的动机太弱了，而且及时交付与规范相一致的软件太困难了"[FISC01,p.64]。这种风格的软件工程最适合于改善现存的系统，但是软件开发的关键问题仍然存在。

但是，一些元素可以在系统开发模型中使用，就像下面极限编程(Extreme

Programming，简称 XP)的例子中说明的那样——至少当涉及到测试和改善软件的时候。

0.2.4　轻便模型，尤其是极限编程

在 Internet 时代，快速地交付软件产品变得日益重要。因此，加快软件开发的过程是必然的。系统开发的传统过程模型(像 V 模型[V-MOD01]或 Catalysis[ELT01])，都是非常详细和形式化的。它们以前被设计用于大型项目。在 Web 环境下的开发项目的特点是紧迫的上市时间进度表，所以它从这些模型中受益很少。

所谓的轻便模型正好弥补了这个缺点，像 Kent Beck 的极限编程(Extreme Programming，简称 XP)[BECK99]，这个模型针对的是特别短的开发周期。XP 没有一个详细的分析阶段，只是把焦点放在了软件开发的核心过程上，也就是编程上，因此，系统地计划和执行的测试起很重要的作用。

XP 项目的主要步骤是版本计划、代码迭代和每个独立版本的测试(图 0-2)。

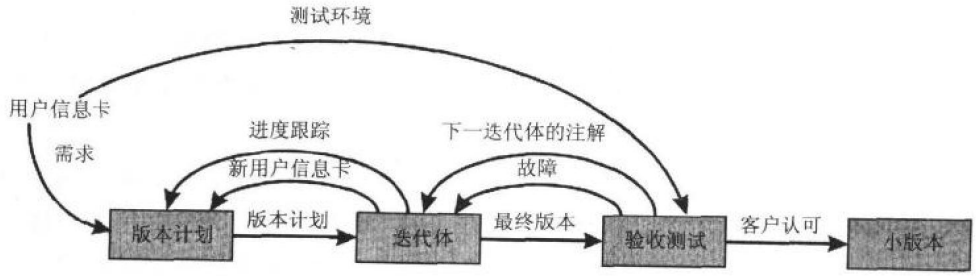

图 0-2　极限编程

尽管形式化水平较低，但为了确保正确的结果，我们引入了所谓的实践概念。最重要的实践是：

- 版本计划

首先，我们必须决定下一个程序版本的作用范围。与客户(软件的未来用户)一起讨论程序所期望的功能，包括决定实际能实现哪一部分的功能。客户选择在下一版本中能够完成的最重要的任务。每个开发版本的体积都应该保持相对的小。版本间隔一般为一或两个月(短版本周期)。由此，客户可以很早地获得至少提供了部分功能的产品，然后他就有机会发表意见并希望做一些改动。必须注意的是，在这个阶段，要忽略客户在以后的过程中可能希望的任何扩充。只考虑那些当前任务所需的扩充。

- 双人编程

实际的开发工作是由二人组完成的。其中一个成员负责实际的编程，另外一个为将来的实现开发策略、指定测试用例和详细阐述设计简本。由此产生的代码由所有编程人员共同拥有。因此，每个人都有权在代码中任意改动和运行测试用例。很明显，

我们强烈需要自动化的测试过程和通用编程规则。

- 连续的代码集成

在测试计算机环境中，各自的开发小组把软件的每个新部分集成在整个系统中，并用所有的测试用例检查这个部分。

- 测试

在 XP 中，质量保证基本上是由测试完成的。在开始写代码之前，编程人员必须确定以后自动运行的所有相关测试用例。在集成新的功能之后，所有现存的测试都要重新执行一遍，以确保在修改之后整个系统能正确运行。

当我们仔细观察这些实践的时候，我们就会清楚地知道它们类似于前面讨论的开放式系统开发。这两种方法在某些方面是一致的。两种方法一般都有短的版本周期，每个周期仅增加很少的特性，并很快地集成用户的反馈。两者都以用户驱动为焦点，都有连续的代码复查和紧密的反馈循环，并依靠一种文化来增加特性和修复故障。我们很可能会怀疑这些相似点是偶然的。我们倒是有这样的印象：专业系统开发系统地修改了开放源代码运动所证明的原则。

但是，新的轻便开发模型(像 XP)不应该免受批评。相反地，有很多方面是应该受到批评的。因此，这些模型(至少在当前的发展阶段)实际上是不是解决软件开发困境的方案，还值得怀疑。

由于它们的结构设计，所以轻便模型似乎主要适合于小型项目。不够详细可能在大型项目中是个问题。例如，项目管理任务很难用 XP 描述，而且，没有配置管理或版本的观念，而这却是分布于不同位置的开发活动的基础。参与的小组成员所需的资格水平没有确定。质量保证只限于应用自动化的测试，而到目前为止自动化测试并不能保证所有的质量标准。对非功能性需求的忽视(像健壮性或性能)，将会导致软件以后不能使用[STAH99,p.265]。

对批判 XP 来说，主要一点是 Kent Beck 假设修改的工作是在贯穿整个软件开发过程中不断进行的。这与公认的学说是相对的，公认的学说假定系统开发早期的错误可以在后期纠正，只不过需要额外的工作[ELTI01,p.189f.]。

0.3　科技驱动的开发

0.3.1　对等式计算

对等式是利用 Internet 周边的可用资源(存储器、周期、内容、人的存在)的一类应用[SHIR01]。它不是客户-服务器模型，这是对它的最简单定义。在对等式中，一个应

用程序被分为多个组件，这些组件有着相同的作用 [SING01,p.4]。从这个观点来看，对等式这个术语似乎不是新出现的。从字面意思来看，甚至服务器或大型机与其他服务器或大型机会话也有对等关系，并且 80 年代早期的 USENET 和 FidoNet 就被认为是第一个现代意义上的对等式应用。

现在似乎对等式计算得到了复兴。对等网被认为是 Internet 发展的下一个阶段。这种科技方法不受交换平台(像 Gnutella 或 Morpheus)的限制，而是业务解决方案的潜在基础。事实上，所有 IT 业的主要参与者，像 IBM、Intel、Microsoft(Farsite 项目)、Sun Microsystems(Juxtapose 项目)或 Lotus，都致力于商业部门的对等式应用。

对等式计算期望能够利用 Internet 巨大的性能潜力。直到现在，Internet 上的一般客户端计算机仅使用了处理器和存储空间中的一小部分，而大部分都在空闲。因此，非常有前途的是，开发出对等式应用程序可以利用和集中未使用的资源，并为例如研究和开发项目(Intel 已经提出在白血病研究中应用对等式计算)提供巨大的计算能力。

对等式体系结构可以应用于很多领域。因此，这个方法为系统开发带了来巨大的潜力。它和面向组件与代理的系统开发(见 0.2.1 节)、开放式系统开发(见 0.2.3 节)和 Web 服务的概念(见 0.3.2 节)有相互依赖性和可能的协同作用。另外，对等式计算不仅为开发中的应用系统提供了一个实现方法，而且它本身还可以用作支持系统开发的分布形式的开发平台的基础(像 USENET 支持 Linux 开发的例子)。

0.3.2　Web 服务是未来吗

就像我们在开始提到的，Internet 的内容变得越来越活跃。理想情况下，许多不同 Web 站点的功能都无缝地结合在一起。这样，Internet 将会从文档和一般事务的媒体空间变为服务的全面的基本结构，这个结构可能会非常复杂。这个想像的技术基础就是所谓的 Web 服务(图 0-3)。

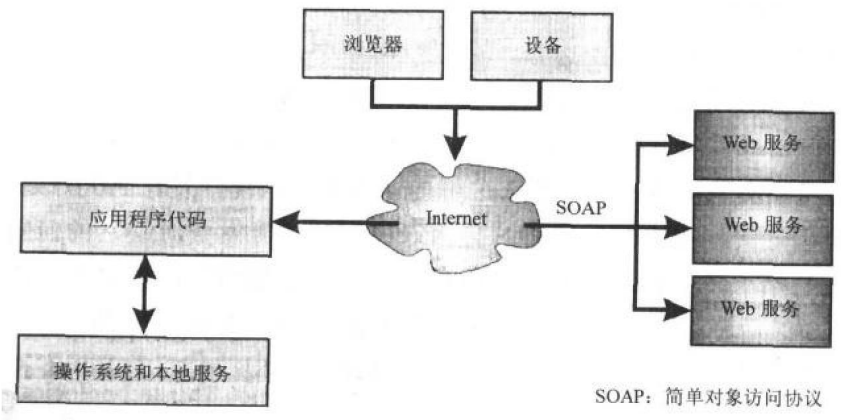

图 0-3　在基于 Internet 的应用程序中使用 Web 服务

Web 服务是程序模块或黑盒功能，它们在 Internet 上通过协议被提供并可以被编译成复杂的 Web 应用程序或门户[STAL01,p.72]。公司可以把财务、库存、后勤、营销和更多的 Web 服务嵌入到他们的 B2B Web 系统中以生成动态的、定制的商业 Web 或 B2B Web 市场。例如，公司可为跟踪货物使用一项特殊的 Web 服务，通过 Internet 可以一天 24 小时地向客户和代理商提供货物运输和跟踪的信息。Web 服务还可以作为提供应用服务(application service providing，简称 ASP)的理想基础，ASP 表示在 Internet 上提供动态的服务，以换取报酬。

Web 服务的实现需要某些基础结构的先决条件：

(1) 为了通过 Internet 提供功能和数据的通信协议；

(2) 目录服务，它可使 Web 应用程序动态地查找和检索所需的服务；

(3) 文档协议，用于标准化描述提供的所有服务。

而且，我们必须要解决其他一些基本问题，像身份验证、数据管理和系统管理等。全部的工作就是要提供一个完整的 Web 基础结构。一些大型的 IT 厂商，其中有 Microsoft 和 Sun Microsystems，它们都致力于用有竞争力的方法开发这类架构。

Microsoft 的方法称为“.NET”。它被誉为引入了一个全新的范式。.NET 是 Microsoft 用于开发 Internet 应用程序的未来平台，它基于像 HTTP 和 XML 一样的开放标准。.NET 平台包含 4 个组件，Framework 和 Tools、Building Block Services(相当于提前开发的连接到应用程序上的 Web 服务)、Enterprise Servers(提供基础结构)和 Mobile Devices(支持日益增长的移动计算部分)。

对开发人员来说，.NET 平台的 Framework 和 Tools 特别重要。.NET 平台的核心由所谓的公共语言运行库(Common Language Runtime，简称 CLR)组成。架构的编译器——现在的 VB.NET、Visual C++和 C#，都来自 Microsoft——不再生成本机代码，而是把源代码翻译成中间语言，也就是所谓的 Microsoft 中间语言(Microsoft Intermediate Language，简称 MSIL)，以后在运行库的管理下用实时编译器执行。所有提供 MSIL 的编译器都可以使这段代码在运行库的控制之下执行。这样，就实现了新一级的语言集成和开发语言的自由选择。在这个模型中，不同语言的集成在代码层，而不是在二进制层(就像在早期的 COM 方法中)。目的是能够在任何语言中编写对象类，并用其他语言使用或重用这些类，包括跨语言的继承(图 0-4)。

如果真的能够实现承诺的话，那么这可能意味着对系统开发有巨大好处。一方面，由于不再依靠特定的专家，开发语言的自由选择缓解了开发资源缺乏的问题。另一方面，软件重用开始了新的机遇。遗留的源代码，如果必要的话，可以被修改和重新编译到 MSIL 中。然后将它重用于开发项目(参见 0.2.1 和 0.2.2 节)。此外，有新的机会来进行独立于平台的开发，例如提高可移植性 [STAH99,p.88]。例如，在 Linux 和 Apple OS X 上实现.NET 的工作。

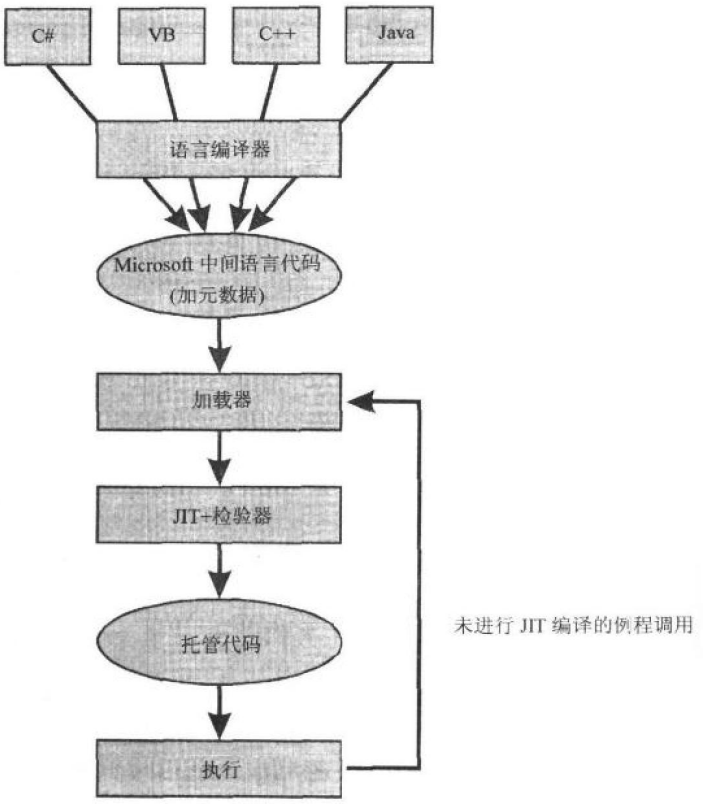

图 0-4　Microsoft .NET 程序的执行模式

Sun Microsystems 的 Web 服务平台称为开放网络环境(Open Net Environment，简称 ONE)。它基于 Java，以及许多其他来自 Web 和 XML 环境的标准。这样保证了平台的独立性，但是需要与其他公司和标准化团体紧密联系。Sun Microsystems 的方法当然要比 Microsoft 的.NET 更加开放，而且可以依靠市场力量和 Java 在开发行业中的成熟。ONE 的缺点在于当前这个体系结构的不完善状态和给人以仍处于概念阶段的印象。

在网络重要性日益提高的环境中，Web 服务的开发可以在技术上被理解为组件思想的进步。与已有的建立在特定对象协议上的组件技术(像 CORBA 或 DCOM/COM+)相比较，Web 服务是基于 Internet 标准的，因此不受特定平台的限制。

当然，在即将到来的时代，大量工作将会投入到 Web 服务。尤其是 IT 业大厂商的参与，使我们有理由期望它不是一个没有任何持续影响就会很快消失的宣传品。观察目前解决问题的方法是否与何时从开发阶段转到成熟阶段，将是令人兴奋的。如果它们准备进入市场，那么对 Internet 和系统开发的冲击将会是巨大的。Gartner 的一些市场研究人员认为 80%或更多的 Web 应用在未来将会基于 Java 或 Microsoft .NET 产品[CLAR01,p.17]。

0.4 对软件质量和测试的影响

对软件质量要求在很大程度上依赖于关联的 IT 系统的使用范围。一个公司或组织的 IT 系统的战略和商业相关性越大，系统故障所带来的危害就越大，所以有效的质量保证就变得越重要。如果系统开发必须在短暂的上市时间和有限的生产力下进行，那么日益增长的质量要求就成为一个问题。由于这些与 Internet 时代的软件开发相关联的限制，所以需要鉴定最高的质量风险，而且也要对质量保证和测试的不同任务的优先级进行计划。

在目前条件下，尤其是在这样巨大的时间压力下，彻底的测试似乎是不可能的。在很多情况下，有必要转变为抽样测试，因此限制测试正威胁着软件的质量。尤其是基于 Internet 的应用程序的质量要求的多样性(像可用性、可访问性、安全性、兼容性和性能)，在依靠抽样测试的情况下带来了风险。因为这个问题，所以测试应该基于一个风险分析。

另外一个解决途径是测试自动化和测试工具的使用[见本书中第 16 篇论文]，就像在极限编程的概念中实现的那样。XP 中的质量保证是由测试实现的。在 XP 开发中的进化特征(实现和集成新功能的短周期)使自动地运行所有测试用例变得很有必要。当在开发过程中应用 XP 或其他轻便模型时，不要忽略非功能性质量要求是很重要的，这些要求在 Internet 环境中是特别重要的(如性能)。

在系统开发中，我们所说过的面向对象和面向组件也会对软件质量和测试行为产生影响。测试过程被分为不同的部分：每个专家开发各自的对象或组件，并提前分别进行测试。在这个较低的级别上，应该可以保证软件质量相对较高的标准(还有在对象和组件的可能的重用方面)。在一个较高级别上，集成测试的重要性日益提高[见本书中第 8 篇论文]。由于基于对象和组件系统的整体质量高度地依赖于底层的 IT 体系结构，所以应该把大量的工作放在体系结构的设计上。

对提高软件质量和测试来说，开放式系统开发和外包测试仍然是进一步的选择。开放式系统开发似乎更适合于开发对安全敏感的和高质量的软件。这是集成在这个方法中的短暂的反馈循环、连续的代码复查和故障修复的特定习惯造成的结果。依次地，开放式系统本身是由 Internet 技术和像对等式计算这样的概念启动的。不利的一面是，只有在公司允许全世界共享它的源代码的领域，开放式系统开发才是有可能的，而当今竞争的市场中这种情况却相当少见。外包测试没有这样的限制，并且证明是非常成功的。