第31卷 第3期 计 算 机 工 程 2005年2月
Vol.31 №3 Computer Engineering February 2005
·发展趋势/热点技术· 文章编号：1000—3428(2005)03—0007—03 文献标识码：A 中图分类号：TP 311.54

# 面向性能的软件再工程研究

沈 铖，彭 鑫，夏宽理，赵文耘
(复旦大学计算机科学与工程系软件工程实验室，上海 200433)

**摘 要**：将提高遗产系统性能作为软件再工程的一大目标加以考虑，提出了一种基于反模式的再工程方法。这种方法通过对一些反模式的特征进行识别，在遗产系统中发现对于系统性能有不良影响的设计并通过一定的方法加以消除，从而达到改善遗产系统性能的目的。

**关键词**：再工程；性能；模式；反模式

# Research on Reengineering Based on Performance

SHEN Cheng, PENG Xin, XIA Kuanli, ZHAO Wenyun
(Lab of Software Engineering, Department of Computer Science and Engineering, Fudan University, Shanghai 200433)

**【Abstract】** Reengineering is a feasible way to improve the quality and maintainability of legacy systems. Current researches focus on reengineering of structures, but do not take performance as an independent aspect into account. This paper mainly concentrates on reengineering of performance, and presents a reengineering approach based on anti-pattern. By describing characters of some anti-patterns, the approach can identify and eliminate those bad designs to develop the performance of legacy systems.

**【Key words】** Reengineering ; Performance; Pattern; Anti-pattern

## 1 面向性能的再工程

性能是一种很重要的软件特性，性能的好坏在很大程度上决定了软件质量。性能上的失败即使并未导致整个项目功能上的失败，其负面影响还是可能波及到其它方面，比如破坏客户感受、减少适用范围、导致成本上扬甚至错过商机等等。及时性是软件性能的一个主要指标，它包括响应性和可伸缩性，这是本文考虑的主要方面。响应性是系统实现其响应时间或吞吐量目标的能力。可伸缩性是系统在对其软件功能的要求增加的情况下，继续实现其响应时间或吞吐量目标的能力。

软件再工程是提高遗产系统质量和可维护性的现实途径。国内外许多研究机构都开展了许多再工程方面的研究，并提出了一系列的支持系统。比如青鸟程序理解系统JBPAS、SourceForge项目 ESS_MODEL、瑞士伯尔尼大学的FAMOOS和美国McCabe & Association公司的McCabe IQ。以上这些工具都提供了一系列自动化/半自动化的支持功能，大大减轻了人工负担，提高了再工程的效率。但是，这些系统都没有将性能作为一个独立的方面考虑，尤其是对常见的性能反模式的标识和消除。

在系统结构设计这个层次上，性能不良软件往往存在着各种各样的性能反模式，这些反模式在很多方面都影响着系统的性能，而且通常比较隐蔽，人工发现比较困难。自动定位这些反模式，提醒开发人员对其进行修正将有助于提升系统性能。我们将性能作为再工程的一个独立方面加以考虑，并将性能工程的相关方法与再工程支持工具结合起来，从而达到提高遗产系统性能的目的。在这里，我们将使用FDReengineer辅助进行相关的分析工作，通过在再工程过程中对反模式进行匹配，发现性能反模式，进而给出相应的修改方案，从而提高系统的性能。

## 2 几种典型的反模式

### 2.1 上帝模式

上帝模式就是在系统设计中，某个类承担了系统的大部分的计算逻辑(图1)；或者相反——包含了系统中大部分的数据。这两种情况，从本质上来说都是计算逻辑或数据分布过于集中。从面向对象设计的角度看，这种设计必然造成类间耦合度和内聚度上的不合理。从性能工程的角度而言，操作和数据的异地化必然导致冗余的消息开销，这将大大影响应用系统特别是分布式系统的响应性。

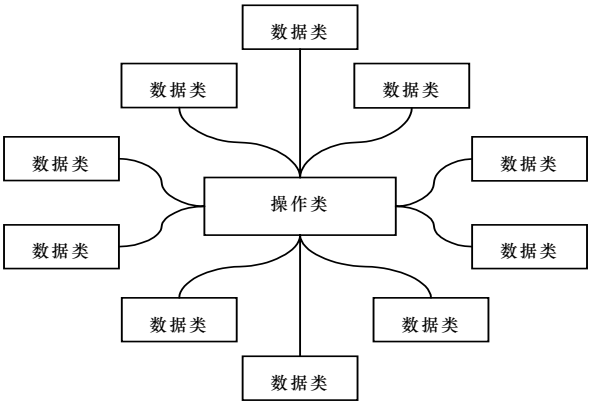


图1 上帝模式

### 2.2 过量动态分配模式

动态分配是一种使用广泛的技术。系统可以在需要的时候创建对象实例，而在使用完成后将实例销毁。这种方法可以根据实际需要动态调整对象空间，极大地增加了系统的灵活性。但是，过量的动态分配会导致出现大量的不必要的对象实例创建和销毁过程。举例来说，用户访问一个JSP页面

**基金项目**：“863”计划基金资助项目“基于Internet以构件库为核心的软件平台”（2001AA113070）；“863”计划基金资助项目“上海构件库及其应用”（2001AA114010）

**作者简介**：沈 铖(1979－)，男，硕士生，主研方向：软件工程；彭 鑫，博士生；夏宽理，教授；赵文耘，教授、博导

**定稿日期**：2004-05-19 **E-mail**：scieny@yahoo.com.cn

时，服务器端需要创建一些类的实例来响应用户的访问。或许单个用户的访问并不会占用多少资源，也根本不会产生任何问题。但是，当这个网页的访问量大到一定程度时，对象实例的创建和销毁将成为系统性能的一个瓶颈，甚至影响到整个网站的响应性和可伸缩性。

**2.3 迂回寻宝模式**

相信很多人都有这样的经历：某件东西被转借多次，往往最后会兜了一大圈才终于找到。这样的情况就是迂回寻宝模式。由于程序设计的问题，数据往往被存放在难以直接取得的地方，其结果就是程序在运行过程中，常常不得不经过很多不必要的路径，花费颇多周折，才终于找到最终需要的数据。比如程序从A处获得目标数据库的URL，从B处获得数据库的用户名，从C处获得相应的密码，又在访问数据库之后，才终于取得了最终所需文件的URL，然后开始下载。

**2.4 单行道桥梁模式**

单行道桥梁模式的典型特征是流量的单向单行性。由于流量每次只能单向通过，而且每次只能通过一个流量，因此当流量达到一定程度时，系统就会不堪重负，造成大量的流量积压。

单行道桥梁模式与寻宝模式最重要的区别在于：前者没有使用并行处理，改进方法主要是缩短处理时间和采用并行处理；后者采用了迂回的路径，改进的方法是通过改变数据结构等直接取得最终数据，详见图2。

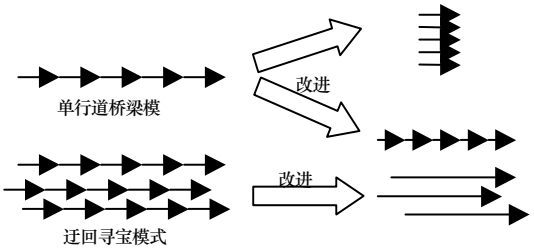


**图2 迂回寻宝模式和单行道桥梁模式的区别**

这种反模式情况最容易发生的场景就是访问数据库。比如在读数据时对所有数据进行锁定，大量的用户请求达到，必将引起系统的拥塞。事实上，很多单行道桥梁模式都是可以进行改进的。J2EE中的值对象模式就是采用了这样的思想：为了减少远程调用的数目，一次调用可以返回多个值。在使用打印机等独占设备时也会发生类似的情况。

## 3 反模式的识别方法

由于性能反模式的抽象层次比较高，因此对其识别也比较复杂，需要通过各个方面的观察：包括程序的静态结构、动态结构，还有系统资源的使用情况。静态结构主要通过对程序代码的静态分析获得；动态结构需要使用AOP技术插入日志代码，然后运行改造后的程序，进而分析所得的日志，从而获得程序的动态信息；资源使用状况的监控既可以通过AOP内嵌代码，也可以借助于第三方的软件。在获得以上的所有信息之后，就可以通过一定的分析和匹配，找到可能存在的反模式。

**3.1 代码的静态分析**

代码的静态分析在不运行程序的情况下获取程序的信息。它把源代码解析成抽象语法树(Abstract Syntax Tree, AST)，通过对这些语法树进行分析，得出相应的结果。静态分析技术已经被广泛应用，它可以执行如下功能：

· 生成类图。类图展现了系统中类的静态结构，即类与类之间的相互联系。面向对象系统中，类是经过良好封装的逻辑实体，也是最基本的抽象单位。因此类图通常是系统最低抽象层次上的表示。全局类图体现整个系统中各类之间的相互关系。通过这种类似于树形的组织结构可以较好地获得各类在系统中所处地位的感性认识。局部类图以一个类为观察对象，具体体现该类内部结构以及与其他类的关系。

· 度量分析。度量是一种以量化的方式获取软件属性的方法，它能从大量的软件代码中提取出某些具有丰富涵义的评价信息，可以帮助再工程实施者获得一部分系统理解。同时可根据度量结果对相应指标做出评价，帮助实施者对再工程实施细节进行决策。比如代码长度、继承深度、类包内聚度等，还可以识别出一些类的特殊属性。

· 系统划分。通过对各类间引用关系的分析，以及技术人员对系统的理解，可以进行一些系统划分。系统划分可以帮助实施者更好地理解程序，同时也可以帮助提取备用构件。

**3.2 代码的动态分析**

代码的动态特征只有在系统运行的时候才会表现出来。和静态结构不同，它包括系统的一些实际的运行状况。对它的分析主要是利用AOP技术，通过在原有代码中插入日志代码来实现。通过分析，可以获得：

· 对象图。对象图是类图的一个实例，它描述类图中的类的特定实例以及某一时刻这些实例之间的特定链接。对象图使用了与类图相同的符号，只是在对象名下附加下划线，对象名后可接以冒号和类名。

· 时序图。时序图用来描述对象间的交互行为，它关注于消息的顺序，即对象间消息的发送和接收的顺序。时序图还揭示了一个特定场景的交互，即系统执行期间发生在某时间点的对象之间的特定交互。它适合于描述实时系统中的时间特性和时间约束。

· 协作图。协作图着重于协作对象之间的交互和链接。它可用于图示系统中的操作执行、用例执行或一个简单的交互场景。协作图展示了对象及其间的链接，还展示了链接的对象之间如何发送消息。协作图和时序图都可以用来展示系统的交互，时序图强调时间和顺序，协作图则强调对象之间的关系。

· 系统内部响应时间。即从系统中的某一点运行到另一点，所需要的时间。系统内部响应时间，可以作为系统运行状况的一个指标。同时可以和外部监测相对照。

**3.3 监测系统资源使用**

系统资源包括CPU、磁盘、内存、网络和数据库等。系统资源使用的监控方法有两种：(1) 内部监测，也就是在程序内部插入操作系统的API函数，直接将系统资源使用状况和当时的利用率写入日志；(2) 外部监测，即利用第三方的软件监测系统资源利用率，不过需要和系统日志相对照。通过监控,可以发现系统资源的异常频繁使用,也可从中积累关于系统资源使用的经验，为系统性能建模提供必要的参考。

## 4 反模式的具体识别方法

**4.1 度量指标定义**

通过上面的静态分析、动态分析和系统资源监测，可以获得一系列的数字、图表和文字描述。为了发现上述反模式，首先定义一些度量指标：

(1) 数据类度量$C_D$：设一个类中存取函数数目为$M$，它除去构造析构函数后的函数数目为$N$，那么这个类的数据类度量$C_D=M/N$。

(2) 操作类度量$C_O$：操作类通常具有极少的成员属性，并且大量调用数据类的存取函数。假设某个类存取自身成员属性的次数为$P$，调用其它类存取函数的次数为$Q$，那么它是操作类的度量就是$C_O=Q/(P+Q)$。

(3) 操作上帝类度量$G_O$：假设$\overline{C_D}$表示目标类所引用的类平均

操作类度量，$C_O$表示目标类自身的操作类度量，那么$G_O=\overline{C_D}+C_O$ 。

(4) 数据上帝类度量$G_D$：假设 $\overline{C_O}$ 表示目标类所引用的类平均数据类度量，$C_D$表示目标类自身的数据类度量，那么$G_D=C_O+C_D$ 。

(5) 动态内存分配度量*DA*：假设*A*是申请和销毁此对象需要的时间，*U*是此对象的实际使用时间，*C*是在时间段*T*内对象创建和销毁的次数，那么 $DA=\frac{A}{U}\times\frac{C}{T}$ 。

(6) 源使用频率*RUF*：假设在时间段*T*内，*C*是目标资源的使用次数，*t*是目标资源的平均使用时间，那么 $RUF=\frac{C\times t}{T}$ 。要注意的是，因为存在并发现象，所以*RUF*可能会大于1。

### 4.2 上帝模式的识别

我们以前一种情况为例，说明上帝模式的自动化识别的方法。首先标记存取函数。存取函数的识别比较简单，这些函数通常只有一句存取语句，也就是对于类中的属性进行赋值或者返回类中属性的值。其次通过数据类度量$C_D$和操作类度量$C_O$识别数据类和操作类。最后从拓扑结构上看，操作类处于许多数据类的中心，因而可以根据操作上帝类度量$G_O$进行判断。

### 4.3 过量动态分配模式的识别

对于过量动态分配模式的识别过程如下：利用AOP技术插入目标对象的创建和销毁的监控语句，比如在对象的构造和析构函数中插入某段特定的代码。运行经过上述改造的程序，产生对象生命过程的日志文件。日志可以存入数据库，或者记录为XML文件。通过上述的日志文件，可以得到整个系统中对象的创建和销毁情况，从而确定对象的生存周期。那些被频繁创建和销毁的对象应该受到特别的关注，因为其中可能就存在着过量动态分配的情况。我们使用度量*DA*判断是否存在过量动态分配模式。这其中，要特别注意过量动态分配的情况并不一定是发生在系统整个运行周期中的，也就是说，有可能只是在系统运行的某一段时间出现爆发式的过量动态分配。所以时间段*T*的选取可以手工，也可以根据动态分配频率的提高由嵌入程序自动决定。

### 4.4 迂回寻宝模式的识别

迂回寻宝模式比较抽象，自动识别有一定的困难。在这里，对它进行一定的改进，定义泛寻宝模式：对各种紧张的资源进行不必要的频繁调用。泛寻宝模式造成稀缺资源的大量浪费。而且这些调用完全可以使用更经济的方式，比如将多次调用合成为一次。

泛寻宝模式的识别可以按照这样的过程。首先确定稀缺资源，比如数据库读写、远程调用、网络访问。然后利用AOP技术，在调用稀缺资源的语句附近插入日志代码。运行经过改造的程序，产生系统对这些资源的使用日志，从而可以获得稀缺资源的使用频率度量*RUF*。结合再工程工具对软件的结构等进行深入了解，重点关注那些被频繁调用的资源，找出相应的改进方式。

### 4.5 单行道桥梁模式的识别

在单行道桥梁模式的识别中，首先确定目标资源，比如数据库读写、远程调用、网络访问。然后利用AOP技术，在调用目标资源的语句附近插入日志代码。接着运行经过改造的程序，使用资源检测工具对目标资源的利用率进行监控，得到资源利用率图。与此同时，嵌入代码产生系统对这些资源的调用日志，从日志可以生成目标资源调用图。将资源利用率图和调用日志结合起来，如果发现目标资源的利用率始终很低，而同时对资源进行调用的记录数目又始终等于1，那么就很有可能是单行道桥梁模式。

## 5 反模式侦测举例

某公司每天产生36张报表，程序ReportBackuper负责远程下载近10天的所有报表，并且将记录存入数据库中。由于报表格式的改变，同时公司对其性能感到不满，现在需要对其进行再工程。在这个过程中，我们特别进行了性能再工程的试验。

我们对其进行单行道桥梁模式的侦测：

(1) 确定目标资源，这里是网络带宽占用率。

(2) 通过FDReengineer的帮助，发现使用网络的函数只有ReportFecher.java的connect函数，利用AOP技术，在这个函数的头尾分别插入日志代码。

(3) 运行经过改造的程序，使用资源检测工具对网络带宽的利用率进行监控，获得网络资源利用图，如图3。

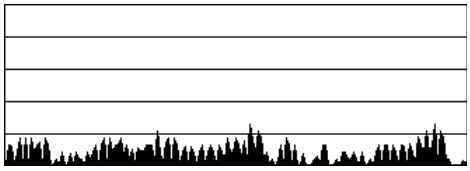

**图 3 网络资源利用图**

(4) 与此同时，(2)中插入的代码产生系统日志。

(5) 分析获得的数据，发现网络利用率始终很低，而且connect()函数从未发生并发现象。经过对程序的进一步理解后确认，这其中的确存在单行道桥梁模式：系统始终只使用单线程依次下载所有360份报表资料，这其中网络延迟占用了大部分的时间，完全可以对此进行改进。

经过思考，我们将这个系统改造为多线程的程序：总共设置10个线程，每个线程分别负责特定某一天的所有报表的下载；而且添加了互助机制，如果一个线程已经完成了它本身的所有工作，它将会设法分担其它线程的工作。通过这样的改造，程序的平均运行时间从62min缩短到了9min。那么在这样的基础上，还能不能有所改进呢？

我们又尝试发现过量动态分配模式，这次特别关注数据库的访问。

(1) 利用AOP技术插入语句，对java.sql.Connection对象实例的创建和销毁的进行监控。

(2) 运行经过上述改造的程序，产生Connection生命过程的日志文件。

(3) 通过上述的日志文件，可以得到整个系统中Connection对象实例的创建和销毁情况。

(4) 经过分析，发现Connection对象实例的生命周期非常短,同时使用又非常频繁,结合程序理解,判断这是一种过量动态分配模式。

由于程序中需要对数据库进行频繁的读写，因而整个过程中系统反复生成和销毁Connection对象达370次，而这个对象的生成和销毁是需要占用较多的系统资源的，在测试机器上，创建这个对象实例平均需要50ms，而平均使用时间只有10ms，因而 $DA=\frac{370}{540}\times\frac{50}{10}=3.43$ ，是这个程序中*DA*度量最高的系统调用。我们继续对这个程序进行改造，试着使用连接池，又将运行时间缩短到了8min。如果对于更大型的数据库访问系统，以上改造的效果应该更加明显。

## 6 总结

再工程可以对遗产系统进行很好的复用。面向性能的再

（下转第47页）

表1 恒星轨道计算服务的部分计算结果

| HD | MaxDG | MinDG | MaxR | MinR | MaxZ | MinZ |
|---|---|---|---|---|---|---|
| 400 | 9.417 724 979 | 7.359 853 302 | 9.417 712 117 | 7.359 839 485 | 0.023 157 041 31 | -0.023 158 649 69 |
| 3 454 | 12.907 995 19 | 8.180 644 823 | 12.905 731 5 | 8.175 776 174 | 0.494 612 757 0 | -0.494 596 831 0 |
| 6 834 | 10.631 369 87 | 8.425 455 728 | 10.629 768 65 | 8.422 575 452 | 0.328 493 368 2 | -0.328 585 023 0 |
| 6 840 | 8.882 434 716 | 5.668 196 142 | 8.881 220 688 | 5.666 863 401 | 0.236 799 381 3 | -0.236 793 952 2 |
| 10 307 | 8.758 476 619 | 6.520 187 255 | 8.758 358 689 | 6.520 068 895 | 0.067 378 156 29 | -0.067 377 170 94 |
| 11 007 | 11.454 213 00 | 8.141 444 23 | 11.447 157 44 | 8.123 503 068 | 0.857 172 143 9 | -0.857 115 559 5 |
| 11 592 | 10.106 302 23 | 6.366 070 751 | 10.104 294 91 | 6.363 433 894 | 0.341 395 952 | -0.341 575 422 1 |
| 19 373A | 10.254 617 37 | 6.657 543 681 | 10.251 764 39 | 6.653 210 030 | 0.436 444 201 2 | -0.436 544 672 9 |

## 5 可视化服务

天文学家获得恒星样本铁元素丰度数据和恒星样本运行的动力学参数后，调用可视化服务。可视化服务将恒星样本铁元素丰度数据与恒星样本运行的动力学参数进行匹配和统计分析，以可视化的形式将统计分析结果返回给天文学家。可视化的试验结果显示恒星样本在银河系中铁元素丰度的分布。为了进行丰度梯度分析，对试验的样本进行线性拟合，采用*Linear Regression* 算法拟合样本数据。

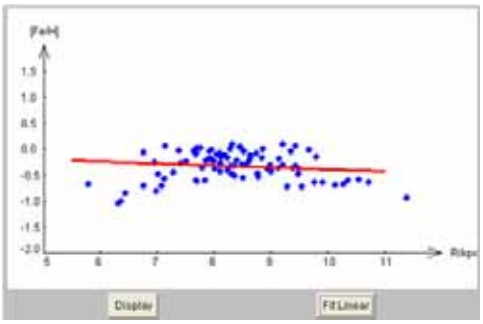

图2 银盘径向恒星样本铁元素丰度分布及线性拟合

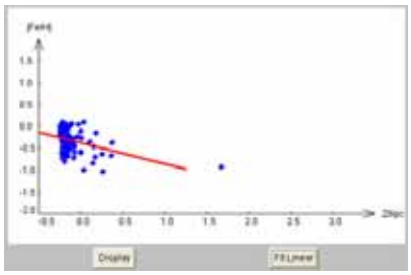

图3 银盘法向恒星样本铁元素丰度分布及线性拟合

以下公式描述 *Linear Regression*模型：

$$Y_i = A + BX_i \qquad A=EY-B\times EX$$

$$B=\frac{\sum_{i}^{N}(X_i-EX)(Y_i-EY)}{\sum_{i}^{N}(X_i-EX)^2}$$

公式中$X_i$表示恒星样本$i$的银盘径向距$R$或银盘法向距$Z$，$Y_i$表示恒星样本$i$的铁元素丰度值。$EX$表示所有恒星样本银盘径向距$R$或银盘法向距$Z$的算术平均值。$EY$表示所有恒星样本铁元素丰度的算术平均值。$B$值即为恒星样本在银河系中铁元素丰度梯度值。沿银盘径向和银盘法向恒星样本铁元素丰度分布及线性拟合的结果如图2和图3所示。

通过试验结果，初步获得以下结论：根据给定的恒星样本计算，银盘径向铁元素丰度梯度大概为$(-0.083)dex/kpc$，银盘法向铁元素丰度梯度大概为$(-0.403)dex/kpc$；沿银盘法向梯度明显大于沿银盘径向梯度。初步证明银河系铁元素丰度梯度确实存在。

## 6 结论

目前初步完成网格环境下银河系铁元素丰度梯度统计分析示例。该示例展示在分布、异构和动态的网格环境下，通过网格服务的方式实现有效的资源共享和科学数据的高效利用。网格数据服务实现对分布、异构物理数据资源的统一访问；恒星轨道计算服务使得天文学家能够利用高性能的计算资源；可视化服务将统计分析的结果以可视化的形式返回给天文学家。铁元素丰度梯度统计分析是网格环境下银河系化学演化研究的一个重要内容，作为虚拟天文台应用系统中在大量数据资源的基础上开展深层次的分析和处理的一个示例。在将来的研究过程中，一方面需要扩展网格环境下银河系化学演化研究的内容，计划研究氧、氮、钠、硫等化学元素的丰度分布；另一方面，进一步完善已有的网格服务，并将更多有效的网格技术应用到虚拟天文台应用系统中来。

## 参考文献


1 张彦霞, 赵永恒. 虚拟天文台. 天文学进展，2002

2 南　凯, 阎保平. 科学数据库系统平台建设设想. 科学数据库与信息技术论文集(第六集)，北京: 科学出版社，2002

3 南　凯, 阎保平. 中科院“十五”信息化建设项目——科学数据库及其应用系统项目系统平台设计方案. 2002

4 崔辰州, 赵　刚, 赵永恒. 银河系Fe丰度梯度分析. 中国科学, 2000, 30:(10)

5 Press W H, Teukolsky S A, Vetterling W T. Numerical Recipes in C. Cambridge：Cambridge University Press，1992


（上接第9页）

工程将性能作为再工程的一个重要方面加以研究，可以进一步提高再工程的质量。特别是通过使用一些自动化的方法和工具，可以大大减轻人工劳动，减少在提高性能的过程中所消耗的人力物力。

本文的方法已经能够在很大程度上发现以上4种反模式，下一步的工作是对更多的反模式的特征进行抽象总结，从而发现和解决更多的问题。另一方面是对方法进行进一步改进，希望能够取得更高的自动化程度。

## 参考文献


1 Smith C U, Williams L G. Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Reading: Addison-Wesley Pub. Co. , 2001

2 孙家骕，袁　勇. 面向Java语言的逆向工程工具JBRET-JAV. 岳阳师范学院学报(自然科学版)，2002,(1)

3 B"ar H, Bauer M, Ciupke O, et al.The FAMOOS Object-oriented Reengineering Handbook. http://dis.sema.es/projects/FAMOOS/

4 彭　鑫，赵文耘，夏宽理等. 基于Java的软件再工程支持工具研究. 计算机工程与应用，2003, (18):63