维普资讯 http://www.cqvip.com

层次型 Java 软件质量度量模型研究*)

黄 璜 周 欣 孙家骕
(北京大学软件工程研究所 北京100871)

Research on Layered Quality Metrics Model for Java Programme

HUANG Huang ZHOU Xin SUN Jia-Su
(Institute of Software Engineering, Beijing University, Beijing 100871)

Abstract Metrics model is in fact a cluster of criterions to assess software, which may show the characteristics of different software systems or modules and then serve different demands from users. The research on software metrics tries to give characteristic evaluations to software components in component extraction, and then supports users to select reusable components in high quality.

Java has been one of the main languages today. With consideration of characteristics of Java and research on some general metrics model, our model: Quality Metrics Model for Java is born.

Following the principle of "Factor-Criterion-Metrics", more detailed descriptions of factors, criterions and metrics of our model are given. In fact, the metrics model shows us some way for consideration. Through this model, we hope to normalize the point of the views of users.

In JavaSQMM, four activities organize software quality evaluating: understanding, function implementing, maintaining and reusing, and then four corresponding factors of quality come to birth, which are mixed by criteria and metrics.

When designing our Java metrics model, the original development of Object Oriented Metrics Model Tool for Java(OOMTJava)provides the support to process of metrics semi-automatically.

Keywords Java, Metrics, Metrics model, Metrics tool

软件度量研究方兴未艾,构建软件度量模型是其中一个重要的研究方向。

度量模型从本质上说,是一组对软件进行评价的标准。这个标准并不是仅仅为了用来证明软件的好坏优劣,也用来反映不同软件系统或者软件模块的特点,从而为不同的需求服务。计算机科学中没有绝对的概念,它从诞生起就是紧密围绕应用服务的,计算机科学的研究一定要和现实需求相关联,因此作为一种评价标准,度量模型同样是从不同方面来综合评测软件的属性,从而在不同需求面前提供可参考的选择。

在我们的工作中,对度量进行研究的直接目的是为了在构件提取中,对构件进行特性评估,也就是在性能和实现特性上对构件进行描述,从而为用户选取可复用的高质量构件提供支持。

目前,随着 Internet 的迅猛发展,Java 魅力四射。Java 是新一代的面向对象语言,非常适合于 Internet 应用程序的开发。与传统软件不同的是,Java 软件可以通过 Java 虚拟机,实现二进制代码上的兼容,从而运行在各种不同的运行环境中,减轻软件开发的工作量,减少工作时间。综合来看,Java 语言具有简单性、平台无关性、动态性、安全性、分布性、健壮性、可移植性和高效性等特点,它采用多继承机制,其虚拟机可以直接对字节码解释执行。

我们从 Java 的特性出发,结合 Factor-Criteria-Metrics 建模规则,REBOOT 模型[6],QMOOD 模型[1],JBRMM[13]模型等,对 Java 软件评测技术进行了研究,提出了层次型 Java 软件质量度量模型(JavaSQMM),并开发了相关的度量工具。

要素-准则-度量通用模型(Factor-Criteria-Metrics)(图1)是一个典型的层次度量模型。它的贡献在于给出了一套构造层次模型的方法,即软件产品的质量属性可以通过分解,用一组要素来表示,而同时每一个要素又可以分解成若干准则,每个准则继续分解成一组度量,度量的值则可以通过相应方法测量得到。

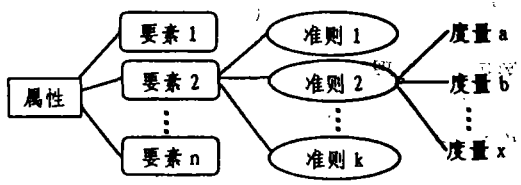


图1 要素-准则-度量模型

1. 层次型 Java 软件质量度量模型设计原则

JavaSQMM 构造遵循如下原则:

首先,"要素-准则-度量"建模原则;

其次,Java 程序共通的属性无需表现在模型中。比如可移植性。度量层也将考虑 Java 的语法及其面向对象的特性。

第三,模型具有可塑性。该模型实际上是一个思考方向,它力图能够对度量用户的思考角度规范化,明确化,使有章可循,有据可考。模型是一个整体的评测概念,从多方角度来考

*)本文研究得到国家"九五"科技攻关项目基金(No. 96-729)及日本理光公司资金资助。黄 璜 硕士研究生,主要研究领域为逆向工程和软件度量。周 欣 博士研究生,主要研究领域为程序理解、逆向工程和软件度量。孙家骕 教授,博士生导师,主要研究领域为语言与编译系统、程序理解、逆向工程。

维普资讯 http://www.cqvip.com

虑程序质量。但用户在实际评测中，往往只注重其中的一部分。而且评测的对象也往往会有粒度上的不同，系统，Java 包，或者一个类，一个方法等等。因此用户应该可以对模型进行裁减。同时，不同的用户对最终结果的偏重也会有不同的偏好，因此，权值应该由用户来设置。

第四，相同的下层属性可能会对不同的上层属性产生影响，其中，有些影响是积极的，有些却可能产生消极的效果。在模型中，一些相同的准则或者度量会关联到不同的要素上，这些要素会受到不同效果的影响，甚至是相反的影响。

第五，虽然 JavaSQMT 是针对源代码的半自动度量工具，但 JavaSQMM 模型并不仅仅限于对源代码信息的评估。我们考虑的是软件生产过程中各个要素和软件质量的关系。

第六，模型必须易于理解，易于支持评测过程。

2. Java 软件质量度量模型

在 JavaSQMM 模型中，软件质量评测被分成四个相关的活动：理解，功能实现，维护和复用。相应地，这四个活动分别代表四个质量要素：可理解性，功能性，可维护性，可复用性。

图2中给出了模型要素层的组成。

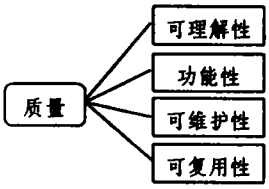


图2 要素层组成

计算公式：

$Q=w1\times U+w2\times F+w3\times M+w4\times R$。

其中 w1，w2，w3，w4是权值，Q 代表质量属性，U 代表可理解性，F 代表功能性，M 代表可维护性，R 代表可复用性。

2.1 可理解性

开发者、用户都会需要理解程序的设计，编码或者文档等。可理解性是一种描述用户识别程序逻辑概念及其适用性的程度的属性[5]。软件所涉及的领域，表现方式，设计文档等都会影响它的可理解性[1]。在 Java 程序中，多线程，多态，继承等都会对可理解性产生效应。同时，可理解性也影响了可复用性和可维护性——一个易于理解的软件设计或程序才易于维护和复用。

对可理解性，我们给出五个准则：文档水平，自描述性，继承，多线程，结构复杂度，如图3。

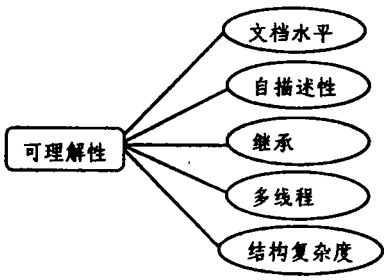


图3 可理解性准则

计算公式：

$U=w5\times D+w6\times S+w7\times I+w8\times T+w9\times Sc$。

其中 w5，w6，w7，w8，w9是权值，U 代表可理解性，D 代表文档水平，S 代表自描述性，I 代表继承，T 代表多线程，Sc 代表结构复杂度。

表1 可理解性度量表

准则	度量
文档水平	文档完整性
	易读性/形象化
	文档密度
继承	继承深度
	多态复杂度
	子类个数
自描述性	类平均注释
	方法平均注释
	包平均注释
多线程	线程个数
结构复杂度	成员函数扇入/扇出数
	类响应数(RFC)
	对象耦合度(CBO)
	方法内聚缺数(LCOM)
	Cyclomatic 环复杂度(CC)
	带权类平均方法数(WMC)
	递归调用语句数

文档是否容易理解(文档水平)以及是否与源代码一致，对于软件质量来说至关重要。开发人员，项目管理人员以及用户都会非常关心软件文档，因为文档可以帮助对软件体系结构，设计以及程序源代码的理解。文档之外，程序中的注释同样对可理解性产生重要的影响。它使用户或者新的开发者可以更直接地理解源代码。在文[6]中，自描述性被描述为对实现一个功能提供解释的软件属性。

继承是面向对象中的重要概念，是一种先进的软件开发机制。但很显然，继承机制增加了系统的复杂度，对软件的可理解性产生负面影响，因为用户需要花费更多的时间，来把相互分离的代码信息整合到一起[1]。

结构复杂度即理解各模块之间相互关系的难易程度[6]。对结构复杂度的度量如 RFC，CBO，LCOM，CC，WMC 等详细描述可以参见文[12]。

对于多线程，很显然，线程越多将意味着程序的可理解性越差。

2.2 功能性

每个系统都有其自身的实现目标，并通过各种功能和事件来实现这个目标。功能性描述了对于需求说明书中所指定的功能，有多少功能被实现[10]，也就是软件是如何来实现这个目标，而且这些实现功能是否能满足用户的需求。在 QMOOD 中，功能性指设计中的类的职责[1]。在模型中，功能性不仅是指在程序代码中的微观行为，也应该包括整个系统宏观上的实现和表达。

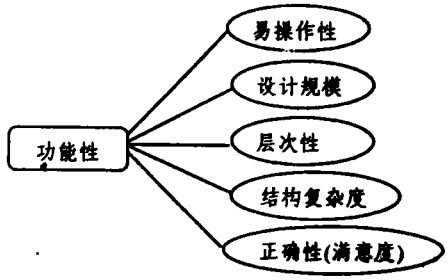


图4 功能性准则

维普资讯 http://www.cqvip.com

功能性包括五个准则：易操作性，设计规模，层次性，结构复杂度和正确性，如图4。

计算公式：

F=w10×O+w11×Sz+w12×H+w13×C+w14×Sc。

其中 w10，w11，w12，w13，w14是权值，F 代表功能性，O 代表易操作性，Sz 代表设计规模，H 代表层次性，C 代表正确性，Sc 代表结构复杂度。

表2 功能性度量表

准则	度量
设计规模	类/方法数
	系统/包/类代码行数
层次性	层次数
结构复杂度	成员函数扇入扇出数
	类响应数(RFC)
	方法内聚缺数(LCOM)
正确性(满意度)	用户满意度反馈
易操作性	界面友好
	帮助信息完整性
	人机交互的多种实现

一个小粒度的设计能够产生由很多小规模的类组成的系统，这比那些由少数大规模类组成的系统复用起来更加容易[1]。设计规模对功能性产生负面影响。

层次性描述了对象之间的概念逻辑关系。在一个设计中，类层次是对概念模型的抽象，并描述了设计中类的继承结构，即一个类共享由其他类定义的结构和行为[2]。而在一个设计中，不同概念的个数指出了这个设计所可以提供的不同的功能数[1]。

结构复杂度会影响功能的实现。在程序中，接口以及方法是一个类和其他类交流的出口。在不同的构件间如何更好地通讯对于功能性来说非常重要。内聚是面向对象方法的重要特性，在功能性定义中，内聚用来描述系统模块间的关系或一致性。如果能够对设计中构件的内聚度尽早地度量，可以找出并重新设计那些低内聚、低一致性的、复杂的构件，从而提高软件设计的质量[1]。

功能性也从宏观上来诠释系统功能实现。已实现的功能是否能满足用户的需求取决于用户的满意评价(满意度)。一个不能让人满意的系统不是一个功能完整的系统[1]。

系统功能在实现上可以一分为二。一部分是核心功能，可能是一些算法，一些业务流程等，用来完成系统需求；另一部分是人机交互，即帮助用户对系统的使用。易操作性即用户与系统交互的方便程度的描述。没有高效的操作方式和交互手段，即使是最好的设计，核心功能的开发都会降低价值。

2.3 可维护性

在文[4]中，可维护性描述软件在交付后，为了纠错，或是提高性能，或是为了适应新的应用环境而被修改的难易程度。为了适应需求的变化，我们经常会被要求对系统进行修改。如果缺乏高可维护性，在修改工作中很容易出现错误，并且不易改正。比如，在低可维护性下，如果我们要修改某个模块，但由于结构上的不合理，对这个模块的修改会影响到其他的许多模块，这样显然会加大工作量，而且无法保证在修改过程中不滞留错误。

对可维护性，我们给出四个准则：文档水平，结构复杂度，模块性，自描述性，如图5。

计算公式：

M=w15×D+w16×S+w17×Mo+w18×Sc。

其中，w15，w16，w17，w18是权值，M 代表可维护性，D 代表文档水平，S 代表自描述性，Mo 代表模块性，Sc 代表结构复杂度。

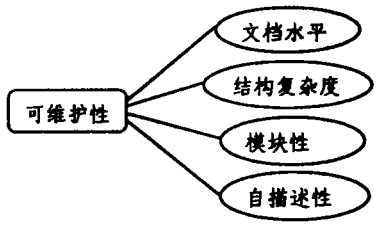


图5 可维护性准则

表3 可维护性度量表

准则	度量
文档水平	文档完整性
	易读性/形象化
	文档密度
结构复杂度	对象间耦合度(CBO)
	方法内聚缺数(LCOM)
	Cyclomatic 环复杂度(CC)
	带权类方法数(WMC)
	递归调用复杂数
模块性	消息发送数
	外部变量访问数
	方法/类内聚缺数
	友元方法数
自描述性	类/方法/包平均注释数

经常会因为需求的变化而需要对系统或系统的某个部分进行修改。如果单纯地依靠阅读源代码来改进，难度是很大的，尤其是在现在系统规模越来越大的情况下，就更不可能了。软件设计分析文档的水平(文档水平)对于理解并改进该程序非常重要。

与文档水平相配合的是自描述性，也就是程序代码中的注释行，同样起到了辅助理解程序的作用。

系统结构越复杂，越难做到对它的维护(结构复杂度)。耦合和内聚分别描述了程序类之间和类内部的结合程度。高内聚的程序，会产生许多小粒度的类，这有利于系统的模块性，并且易于修改。而对于高耦合的程序，某个类的修改会牵扯到许多其他类的修改，势必增加了维护的工作量和系统出错的概率。同样的，Cyclomatic 环复杂度描述了类之间的消息连接的复杂度，如果这个复杂度比较高，显然也不利于维护。

一个系统需要实现如下的目标：使因某个模块的修改而引起的对其他模块的影响达到最小[4]。模块性要求系统易于分割，每个部分可以被独立修改而无影响于其他。

2.4 可复用性

可复用性反映了一个设计或实现不需要很多工作量的修改就可以用来解决新问题的程度[1]。

可复用性包含五个准则：设计规模，继承，模块性，成熟度，容错能力。(图6)。

计算公式：

R=w19×Sz+w20×I+w21×Mo+w22×Ma+w23×Fa。

其中 w19，w20，w21，w22，w23是权值，R 代表可复用性，

维普资讯 http://www.cqvip.com

I代表继承，Sz代表设计规模，Mo代表模块性，Ma代表成熟度，Fa代表容错能力。

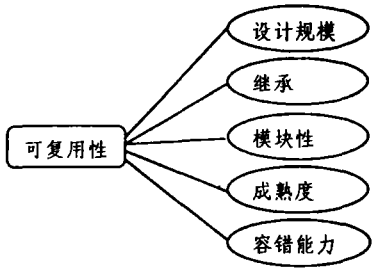


图6 可复用性准则

表4 可复用性度量表

准则	度量
设计规模	方法/类数
	系统/包/类代码行数
继承	继承深度
	多态复杂度
	子类个数
模块性	消息发送数
	外部变量访问数
	方法/类内聚缺数
	友元方法数
成熟度	整合及测试中MTTF
容错能力	成员函数错误容忍度

在对不同版本的设计进行比较时，设计规模是一个有效的属性，无论是对于相同需求的设计还是相关领域的设计。当提供相似能力并解决相似问题的若干设计在可复用过程中被考虑时，设计规模会对可复用性产生很大的影响[1]。

类继承是面向对象系统可复用技术中常用的一种形式[1]。继承使得一个类可以根据另一个类来得到定义，被复用的实现便易于被修改而适应新的系统需要[3]。继承提高了抽象结构的生成，即在一个继承层次中，使对象信息能够向其他多个对象传播，这显然有助于提升系统的可复用性[1]。

模块性有助于将系统切分成可复用的各个部分。高模块性意味着高内聚和低耦合。复用模块时，构件只需要提供少量接口与新系统进行交流。

成熟度描述了软件经受由于自身缺陷而引起的失误的承受程度[5]。成熟度和容错能力都是用来描述系统可信度的属性。我们将这两个属性纳入到可复用性中作为复用的选择度。对成熟度的详细介绍可以参见文[6]。

容错能力是对在有限的硬件和软件错误情况下，系统所能提供的连续的纠错固有能力的描述[4]。它并不意味着系统在错误发生时能够给出正确的结果，而是告诉用户系统不会崩溃，发出一些适当的错误消息，然后对其他不影响这些错误的输入继续工作[6]。

3. Java软件质量度量工具(JavaSQMT)

软件度量模型给出了对软件进行度量的一个标准，而软件度量工具则为软件度量提供了自动化或半自动化的支持。

目前，度量对象已经囊括整个软件开发过程，从需求分析到设计、编码，都有相应的度量方法提出，并配以度量工具来解决。如文[7]中描述了对软件体系结构的度量，文[9,11]中分析了面向对象设计的度量。

JavaSQMT以Java程序源代码为对象，对Java软件进行抽象、分析，用树形图来表示面向对象结构和信息，并提供度量模型创建、计算以及显示机制，辅助用户对Java程序的理解和度量。

图7描述了工具的工作流程。

从结构上，工具可看成由分析、构建、计算和显示四个层次组成。

分析前端对源代码进行词法和语法分析，从源代码抽取信息，并存入到信息库中。这些信息客观细致地描述了程序的结构和组成，如图7中1、2。

在构建层，用户可以自己创建层次型度量模型，也可以直接从度量模板模型库中调取原有的模型。模型创建后，被存储在度量模型库中(3、4)。Java度量模型是针对整个软件开发过程的，本工具主要基于对源代码的度量，因此需要对模型进行裁减，保留其中源代码级的度量属性存入模板库。

根据程序信息和构造模型，度量结果被计算出来，并回填到度量模型库中(5、6)。在计算时，首先计算底层节点。上层节点根据子节点的度量结果以及事先设定的权值，计算自己的度量值。为了方便观察和比较，计算结果都被规整为0－1之间的数字。如果规整值靠近0，说明被评价的属性会出现问题，反之，则说明该属性被控制在希望的范围内。这种计算方法针对度量对象不同侧面，参照这个属性测度范围中的绝对参照点，然后把度量值规整为反映好坏程度的评价度。但度量模型值通常是一个相对的数量。工具也提供相对规整计算。相对规整通常是以一组度量实体为对象。首先选定一个实体计算度量值，作为基准。其他实体的度量都和基准比较，得到相对度量值。两种计算方法的详细内容可参见文[14]。

JavaSQMT中用树形图来显示度量模型。用户可以通过工具提供的交互手段从不同角度来观察模型，或对不同模型进行比较(7、8)。

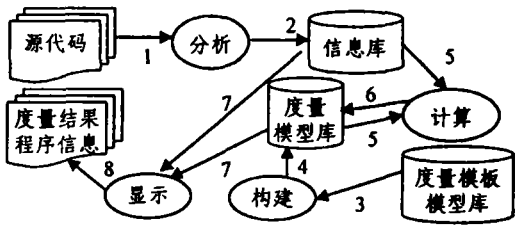


图7 工具工作流程

4. 分析案例

为了对工具进行验证，我们用工具对两个实现相同的Java程序，Vehicle_a和Vehicle_b进行度量，将度量模型中可自动得到的属性建模，度量结果见图8。

从图中可以看到这两个程序在设计上的差异。由于程序比较简单，总的质量属性的度量值相差不大，但逐层进行比较，可以发现它们各自侧重于不同的方面——总的度量值是各个方面综合的结果。在模型设计原则中曾提及，相同的度量可以对不同的上层属性产生不同影响，甚至截然相反。例如，继承属性对可理解性产生负影响，但对可复用性却产生正影响。从上面的论述可以看出，层次型模型的一个好处就是可以观察度量对象的不同侧面。这对于构件提取，程序选择，考核评比等都有很大的帮助。

Vehicle_a的可理解性度量值为0.813562，Vehicle_b的

维普资讯 http://www.cqvip.com

可理解性度量值为0.633626。前者明显高于后者。如果我们要选择一个可理解性高的程序，那么显然会选择 Vehicle_a。而 Vehicle_a 的功能性度量值为0.728198，Vehicle_b 的功能性度量值为0.881309，后者比前者高一成半，如果需要一个功能性强的程序，则会选择 Vehicle_b。从图8中我们可以看到这两个程序在要素层的详细比较。

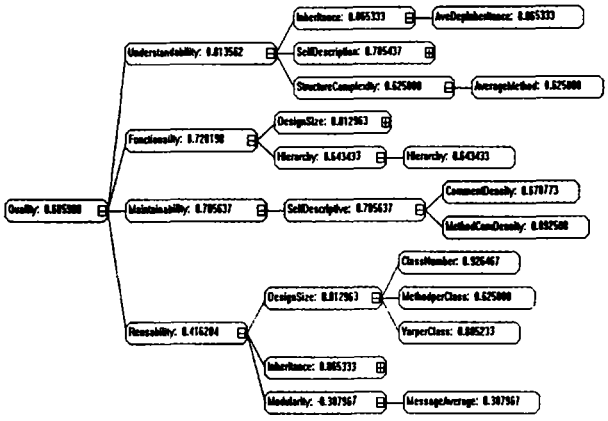
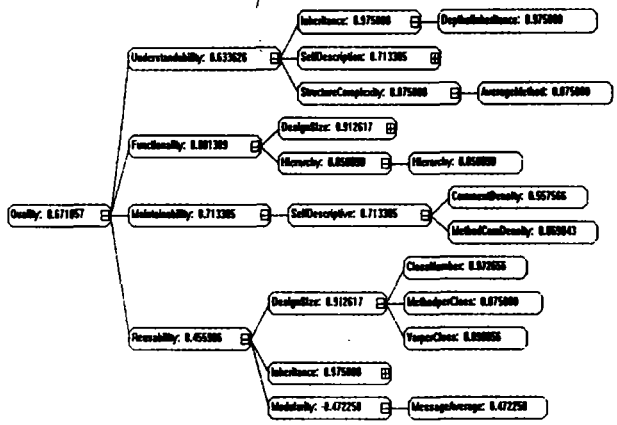


图8 a Vehicle_a 度量结果图



b Vehicle_b 度量结果

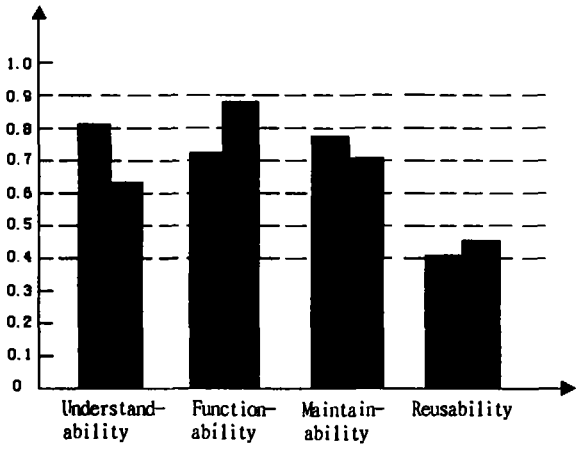


图9 度量值比较

总结 软件质量度量是对软件生产线起参考辅助作用的一个研究方向。度量模型是一种对软件进行评价的标准。通过将软件加载到模型上，我们可以看到该软件展现的各方面的性质。JavaSQMM 不仅仅是面向源代码的，在软件工程的不同阶段，开发者和用户都可以对它进行裁减以适应不同的需要。但也因此，我们并不能对该模型提供完全自动化的工具，其中的一些度量需要用户或开发者的主观判断。

度量模型是一个很具研究潜质的领域，在工程实践中尤为显得重要。现在的研究大多是建立在对实践数据的基础上，在各方面考虑下取得一定的平衡。在以后的工作中，我们将继续这方面的研究。

JavaSQMT 是在源代码级上的度量半自动化实现，在以后的工作中，我们会进一步把度量和软件开发的其他阶段，如设计阶段，进行结合。

参 考 文 献

1 Bansiya J. A Hierarchical Model For Quality Assessment Of Object-Oriented Designs:[Ph. D. Dissertation]. Huntsville: University of Alabama in Huntsville, 1997

2 Booch G. Object-Oriented Analasis and Design, 2nd Edition. The Benjamin/Cummings Publishing Company, Inc., 1994

3 Gammar E, helm R, Johnson R, vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995

4 IEEE Computer Society. IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 729-1983. The Institute of Electronical and Electronics Engineers, Inc, 1983

5 ISO 9126 Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for Their Use, International Organization for Standardization, Geneva, 1992

6 Software Reuse: A Holistic Approach-Measuring the Effect of Reuse Chapter, edited by Even-Andre Karlsson. Chichester ; New York : Wiley, c1995, published by Wiley&Sons, Ltd. Baffins Lane, Chichester Sussex PO191 UD, England

7 Mayrand J, Bruno Laguě. Object oriented architecture assessment using metrics. Bell Canada, 1996

8 McCall J A, Richards P G, Walters G F. Factors in Software Quality, Vols. I, II, III (NTIS AD/A-049 014/015/055), Springfield: NTIS, 1977

9 Harrison R, Counsell S, Nithi R. Coupling Metrics for Object Oriented Design. In: Proc. of the 5th Intl. Symposium on Software Metrics, 1998. 150～157. S. Benlarbi and W. Melo: "Polymorphism Measures for Early Risk Prediction". In: Proc. of the 21st Intl. Conf. on Software Engineering, 1999. 334～344

10 Hendriks R, van Vonderen R, van Veenendaal E. Measuring software product quality during testing. Published in: Conference proceedings European Software Quality Week, Oct. 2000

11 Chidamber S, Kemerer C. A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering, 1994, 20(6): 476～493

12 Systä T, Yu P, Müller H. Analyzing Java Software by Combining Metrics and Program Visualization. In: Proc. of the 4th European Conf. on Software Maintenance and Reengineering (CSMR 2000), Zurich, Switzerland, Feb. 29 - March 3, 2000

13 陈向葵. 面向对象系统中可复用构件的提取及工具支持. 2000. 6

14 谢涛. 青鸟面向对象软件度量框架及工具支持. 2000, 6