软件度量学综述

邢大红　曹佳冬　汪和才　刘宗田
合肥工业大学微机所　(合肥 230009)　　中国建设银行安徽省分行　(合肥 230022)
E-mail:6368808@sina.com

摘　要　该文将介绍软件度量学的发展历史、软件质量度量体系和软件度量方法。着重分析面向对象软件开发技术的发展对软件度量方法的影响。最后给出了下一步研究方向。

关键词　软件度量　面向对象　C&K 度量指标

Introduction to Software Metrics

Xing Dahong Cao Jiadong Wang Hecai Liu Zongtian
(Insitute of Microcomputer Application, Hefei University of Technology, Hefei 230009)
(Anhui Branch of China Construction Bank, Hefei 230022)

Abstract: This paper is a summarization on software metrics. It presents software metric system and introduces software metric methods. It analyzes the influences of object-oriented development technology to software metrics. At last it raises further research issues.

Keywords: software metric, Object-Oriented, C&K metric suite

1　软件度量学概述

软件度量学(Software Metrics)最早在 1958 年由 Rubey 和 Hurtwick 提出。1970 年 Halstead 提出了软件科学(Software Scienes)的概念,他认为任何一门学科要成为科学,必须理论和实践结合,而软件度量学正是反映了这种结合的学科。Boehm 于 1976 年提出,对软件属性不能仅有定性的研究,还必须有定量的研究,软件度量学正是顺应这种趋势而产生的。其目的是用软件度量学的方法来科学地评价软件质量,更有力地对软件开发过程进行控制和管理,合理地组织和分配资源,制定切实可行的软件开发计划,以低成本获得高质量软件。

现在软件度量学已成为软件工程的一个研究方向。ANSI IEEE Std 1061-1992《Standard for a Software Quality Metric Method》中对软件度量学的术语作了明确的定义,对软件度量学的目的作了阐述。并且给出了一个指导性的软件质量度量框架。在此标准中,对于软件质量的需求分析的建立、鉴定、实施和度量结果的分析制定了一个方法标准。

软件质量度量(SQM)就是从整体上评价软件质量,用于软件开发过程中对软件质量进行质量控制,并对最终软件产品进行评价和验收。

软件质量度量模型有很多种,其基本点在于将软件质量的概念分解为若干层次,而对于最底层次的软件质量概念,再引入数量化的指标,从而得到软件质量的整体评价。

1976 年,Boehm 等人提出了定量地评价软件质量的概念,给出了 60 个质量度量公式,表明怎样用于评价软件质量,并且首次提出了软件质量度量的层次模型。Boehm 等人认为软件产品的质量基本上可从下列三个方面来考虑:(1) 软件的可使用性;(2)软件的可维护性;(3)软件的可移植性。Boehm 等人将软件质量的概念分解为若干层次,对于最底层的软件再引入数量化的指标,从而得到软件质量的整体评价。例如,从 Boehm 的模型可知,可维护性能从可测试性、可理解性及可修改性来度量,即高可维护性意味着高可测试性、高可理解性和高可修改性。

1978 年 Waters 和 McCall 提出了从软件质量要素(factor)、准则(criteria)到度量(metric)的三层次式的软件质量度量模型。他们将软件质量要素降为 11 个,且给出了各要素的关系。McCall 等人认为,要素是软件质量的反映,而软件属性可用作评价准则,定量化地度量软件属性,从而反映软件质量的优劣。McCall 定义了 11 个质量要素,分别为:正确性(correctness)、可靠性(reliability)、可使用性(usability)、可维护性(maintainability)、可测试性(testability)、灵活性(flexibility)、可移植性(portability)、重复使用性(reusability)、连接性(interoperability)。McCall 等人还定义了 23 个评价准则。

ISO 于 1985 年提出建议 (按照 ISO/TC97/SC7/WG3/1985-1-30/N382),软件质量度量模型由三层组成。

高层(top level):软件质量需求评价准则(SQRC)

中层(mid level):软件质量设计评价准则(SQDC)

低层(low level):软件质量度量评价准则(SQMC)

根据 ISO 近年来对软件质量的讨论趋势,反映软件质量要素的选用逐渐向面向用户的观点靠拢,使用户和软件人员方便地确定软件质量需求。

我国上海软件中心根据 ISO/TC97/SC7/的建议,同时参照 McCall 模型和 Boeing 模型, 并结合我国实际情况综合构成了 SSC(Shanghai Software Center)软件质量度量模型及其度量方法,从而形成 SSC 软件质量评价体系[1]。

* 该文受机械部发展基金赞助。

作者简介:邢大红,博士研究生,主要从事面向对象方法和软件度量学研究。曹佳冬,硕士研究生,主要从事 KDD 和软件度量学研究。汪和才,高级工程师,主要从事软件开发方法研究。刘宗田,博士生导师,研究员,主要从事软件工程、软件逆向工程和面向对象方法研究。

2 软件质量度量体系

目前软件质量度量体系一般分三个层次：软件质量要素(factor)、准则(criteria)、度量(metric)。质量度量体系(图1)反映了软件质量度量的层次关系。下面阐述三个层次的详细定义和内容。

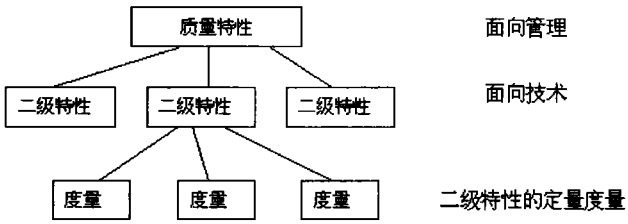

图1 软件质量度量体系

2.1 软件质量要素

参照 ANSI /IEEE Std 729-1983,软件质量的定义为:"与软件产品满足规定的和隐含的需求的能力有关的特征和特性的主体"。在 SSC 模型中软件质量由下列6个质量要素(即特性)来定义：功能性、可靠性、易用性、效率、可维护性、可移植性。

2.2 评价准则

上面6个质量特性是面向管理的观点,或者是从使用观点引入的。为了引进定量量纲,必须将这些面向管理的特性转化为与软件有关的观点,或者是面向技术的观点。这种转化是通过对每个特性定义一组二级特性来完成的。二级特性又称评价准则,它们进一步刻画了软件质量特性,并且有助于描述各个软件质量特性之间的关系,因为一个二级特性可以支持一或多个质量特性。这些二级特性是软件产品或软件生产过程的独立特性。利用这些特性,软件质量可被判定、解释和测量。

在 SSC 模型中,选用了22个评价准则,它们是:可追踪性、完备性、一致性、精确性、简单性、可操作性、培训性、通信有效性、处理有效性、设备有效性、模块性、软件系统无关性、硬件系统无关性、自描述性、结构性、清晰性、可扩充性、产品文件完备性、健壮性、通用性、可见性和安全性。软件质量特性与软件质量二级特性的关系如图2所示。这里仅以可维护性表示。

2.3 度量

SSC 模型的第三层是度量,每一度量由若干度量问题(又称度量元)组成,根据度量问题的回答与记分可反映度量的得分,从而反映评价准则与软件质量特性的得分。在 SSC 软件评价体系中提出了两套度量表,一套面向软件开发过程,对软件开发的每一个阶段采用相应的度量问题。在这一套表中,根据软件的需求分析、概要设计、详细设计、实现、组装测试、确认测试和维护与使用这7个阶段,运用了每一阶段的度量问题,从而实现了软件开发过程的质量控制,为获得高质量软件作了保证。另一套是面向产品的,即对已开发好的软件产品的质量进行度量。

3 软件质量度量方法

软件度量学发展已近30年,在面向对象技术出现以前,人们的大多数研究是基于面向过程的软件设计方法。在这一时期,对软件复杂性的度量的研究最为活跃,成果也较多,应用最为深入[10]。软件复杂性度量(Software Complexity Measure)已成为软件度量学的一个最重要的分支。自1970年 Halstead 提出文本复杂性度量方法以来,已经提出了许多软件复杂性度量方法,这些方法可以大致分为以下4类。结构复杂性度量:此类度量方法研究系统各部分的拓扑结构、控制路径以及控制路径对程序结构产生的影响。算法复杂性度量:此类度量方法考虑完成算法的相对困难性,即时间复杂性和空间复杂性。但此类复杂性不是软件的直接属性,而是算法的直接属性。数据结构复杂性度量:此类度量方法关注数据结构、数据流对软件的影响。文本复杂性度量：此类度量方法对源程序进行静态的分析,通过统计程序中的操作符和操作数,对软件中的潜在错误个数和开发工作量进行估算。

从70年代起，人们开始从统计学和心理学的角度研究软件复杂性。Maurice Halstead 进行了开创性的工作,他通过统计和分析程序中发现的操作符(operator)数和操作数(operand)来研究软件复杂性。从文本复杂性角度考虑,Halstead 认为程序是一个符号序列,设此程序序列包括 η_1 种操作符和 η_2 种操作数,则程序长度可估算为:$N = \eta_1 LOG_2\eta_1 + \eta_2 LOG_2\eta_2$。这比仅考虑语句行(LOC)度量方法进了一步。关于此估算式的正确性已经在理论上和实践中得到验证。

控制结构复杂性度量是对程序拓扑结构复杂性的度量。此类方法主要有 McCable 方法、Woodward 方法、Harrison 方法和 Edward Chen 方法。Thomas J.McCable 于1976年首次提出了基于程序拓扑结构的软件度量方法。McCable 认为程序控制图中线性无关的回路数越多,则软件复杂性越大,测试程序越困难。因此,可以用巡回圈数(CyClomatic number)来度量程序的复杂性和测试的难度。

1984年美国学者 Kuo-Chung Tai 在第七届国际软件工程会议上提出了一种在控制图中基于数据流的程序复杂性度量,其思想是对一个控制图根据它的"典型"数据流信息定义一个复杂性计算。

近年来面向对象技术的兴起,在面向对象分析、设计、面向对象程序设计语言(OOPL)和工具上发展很快,且已得到广泛应用。面向对象技术采用数据抽象、封装、继承、多态性、信息隐藏、重用机制等,为提高软件的可重用性,增强可维护性、可靠性,提高生产效率等方面提供了可能。这些机制在传统的软件开发中是不完善的或缺少的。面向对象系统强调的是对等实体之间的关系而不是传统控制流所决定的层次分解关系。传统的度量有一定的局限性，而且对某些面向对象特性根本无法度量,因此需要新的度量来反映。C&K、MOOD 等度量方法的提出标志着软件度量学进入了第二个阶段[2]。

目前，面向对象软件度量方法处于开始阶段。国际上,以

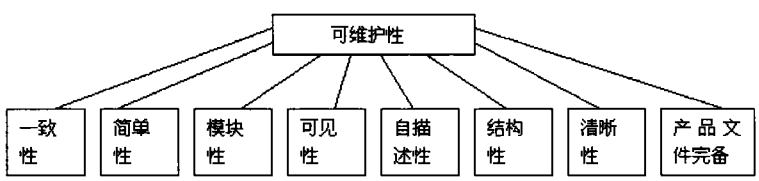

图2 软件质量特性与二级质量特性的关系

Chidamber 和 Kemerer 的研究较为著名。1994 年 Chidamber 和 Kemerer 在 IEEE Trans,softw 发表的《A Metric Suite for Object-Oriented Design》标志了面向对象设计中的软件度量学的初步形成[4]。此文阐述了面向对象设计中软件度量学理论基础，软件度量方法和意义。其他研究人员如 Neville I. Church、Martin J.Shepperd、Martin Hitz 和 Behzad Moutazeri 等针对 C&K 的研究方法做了部分试验和前瞻性的预见[6][7][8][9]。Chidamber & Kemerer 针对类提出了基于继承树的一套面向对象度量方法。在此方法中主要考虑类的继承、类的方法数、类之间耦合、类的内聚性等。C&K 度量方法是针对具体的类的度量，对于指导软件设计中对类的大小、层次、继承关系等设计有一定的指导意义。Chidamber and Kemerer 为面向对象度量定义了 6 个度量指标：类的加权方法数 WMC (Weighted Methods per Class)、继承树的深度 DIT(Depth of Inheritance Tree)、孩子数目 NOC (Number Of Children)、对象类之间耦合 CBO (Coupling Between Object classes)、类的响应集合 RFC(Response For a Class)、类内聚缺乏度 LCOM(Lack of Cohesion in Methods)。 WMC 揭示了开发和维护类的时间和精力。类的 WMC 越大对子类的可能影响越大，其通用性和可复用性就越差。类的 DIT 大，表示它的可能继承方法数目大，复用程度高，但预测它的行为将更困难，同时设计复杂性大。类的 NOC 越大，重用越好，表示该类在设计中有很大影响，此类应成为测试重点。对象类间过多的耦合对模块化设计和重用是有害的。一个类越独立，它就越易被另一个应用重用。类的 CBO 越大，设计中对另一部分的敏感越大，因此，维护越难。类的 RFC 越大，意味着该类测试和调试将更复杂，复杂度越大。设计中希望类的内聚度越大越好，因为这样可以提高封装度。类的 LCOM 越大，意味着类可以分裂成两个或更多的子类。类的 LCOM 大会增加类复杂度，在开发过程中出错的可能性加大[3][11][12]。

Brito 和 Abreu 提出的 MOOD 度量方法是从封装、继承、耦合、多态性 4 个方面提出了 6 个度量指标[5]。MOOD 对封装的度量提出了两个指标：方法隐藏因子 MHF (Method Hiding Factor)和属性隐藏因子 AHF(Attribute Hiding Factor)。数据封装(简单地说封装)经常被用于衡量一个语言隐藏具体实现的能力。通常可以用信息隐藏来定义。度量信息隐藏的属性是代码的可见性(Code Visibility)。AHF 和 MHF 使用类的属性和方法对其它类的代码的可见性来度量信息隐藏。MOOD 度量方法中对继承的度量使用了两个指标：方法继承因子(AIF)和属性继承因子(MIF)。MOOD 方法中使用耦合因子(CF)来度量类之间的耦合。CF 的计算考虑了所有可能的类对集合，判断每对类之间是否存在通过消息传递或通过语义联系(一个类的方法或属性被另一个类引用)产生的耦合。就耦合来说，这两种耦合关系是相等的。一个系统的 CF 的值，对于衡量系统的复杂性、封装性、可重用性、可理解性和可维护性具有重要意义。可以凭直觉判断出 CF 值越大，系统的封装性越差，重用的可能性越低，可理解程度越低，维护的困难越大。但很难说，耦合度越高，系统变得越复杂。例如，很容易构造一个耦合度很高的的简单系统。大量的经验数据在这方面可能会有助于分析 CF 值的范围对于这些外在属性的影响。MOOD 度量方法中用多态因子(PF)来度量系统中存在多态的可能性。PF 的计算是重新定义继承的方法数目除以可能出现多态情况(子孙类的新方法是重载的)的最大方法数目得出的。因而，PF 反映了一个系统的动态连接(dynamic binding)情况。

Brito 和 Abreu 从系统级提出了度量指标并给出了形式化定义。C&K 度量指标是从类一级出发考虑的，考虑的角度是有区别的。例如，耦合性度量(CBO 和 CF)非常相似。CBO 提供了从类一级考虑的耦合，但 CF 则是系统级的观点。根据两种度量指标的定义，笔者认为 C&K 对系统设计和开发大有帮助。而 MOOD 度量指标提供了对一个系统的判断，它将对工程管理者较为有用。

4 总结和下一步研究方向

软件度量方法学的研究还远远未达到人们对它的期望程度，这一方面由于面向对象技术新、发展迅猛，另一方面由于针对面向对象技术的度量的理论研究还不完善。针对面向对象设计的度量工具更是难以满足现实要求。无论是在理论、方法，还是在应用的研究上都存在许多的困难。理论上的不坚实、方法上的粗糙、结合实际应用成果不多，这些都是目前存在的主要问题。下面就以后需进一步研究的方面和内容作一阐述。

(1)应该为面向对象软件度量学建立更为坚实的理论。

(2)现有的软件度量方法较为粗糙，一般是粗线条的描述，大多是静态的，没有考虑面向对象方法设计的对象生成的动态的一面。

(3)软件度量值如何运用于软件开发，目前还只是定性的分析。 这方面需要运用统计学的方法， 为领域内的开发提供指导。

(4)目前的研究还是度量体系的第三层， 即度量元的研究，如何在此基础上建立软件度量体系，这方面的研究现在还没有开展。这方面的研究需要运用统计学知识与神经元网络研究成果。(收稿日期:2000 年 9 月)

参考文献

1.中国软件协会.软件质量及其评价技术.清华大学出版社,1990
2.弓惠生.面向对象设计中软件度量学的进展.计算机科学,1996;23 (3)
3.邢大红,刘宗田.面向对象软件设计中软件度量学的理论和方法.计算机科学,1998
4.R Chidamber,F Kemerer.A Metrics Suite for Object-Oriented Design.IEEE Trans.,Softw.,Eng.,1994;20(6)
5.Brito F,Abreu E.MOOD—Metric for Object-Oriented Design.OOPSLA'94 Workshop on Pragmatic and Theoretical Directions in Object_oriented Software Metric,Portland,OR,1994
6.J-Y Chen,J-F Liu.A new metric for object-oriented design.Information and Software Technology,1993;35(4)
7.E Weyuker.Evaluating software complexity measures.IEEE Transs,Softw.,Eng.,1988;14:1357-1365
8.Martin Hitz,Behzad Moutazeri.R.Chidamber,F.Kemerer's Metrics Suite:A Measurement Theory Perspective.IEEE Trans.,Softw.,Eng.,1996;22(4)
9.Neville I Churcher,Martin J Shepperd.Towards a Conceptual Framework for Object Oriented Software Metrics.ACM Softw.Eng.Notes.1995.4;20(2)
10.弓惠生.软件设计复杂性度量.计算机研究与发展,1992.3
11.邢大红.面向对象软件设计中的软件度量学的研究.硕士学位论文.合肥工业大学,1998.5
12.潘飚.C++语言面向对象的软件度量学研究.硕士学位论文.合肥工业大学,1999.4