文章编号:1001 - 9081(2006)02 - 0439 - 03

构建面向对象系统规模评价模型

张桂珠,须文波
(江南大学 信息学院,江苏 无锡 214122)
(zhangguizhuhz@ yahoo. com. cn)

摘 要:分析了现有的软件规模度量特点和应用于 OO(Objected-Oriented)系统时存在的问题,提出了类点分析法的规模评价模型,用以评估 OO 软件产品特性,能精确地捕捉 OO 的关键概念和结构,并应用在 OO 的设计阶段,独立于软件开发的技术和设计工具。类点分析法中定义了两个相关联的度量,其目标旨在软件的整个开发过程中,当有效的信息被得到时,能不断地精化评估规模模型,使之精确地预测 OO 系统的开发代价和项目管理。给出了实施类点分析法的系统规模评估过程。

关键词:面向对象系统;类点分析法;软件规模度量;开发代价模型
中图分类号: TP311 **文献标识码**:A

Modeling size estimation of object-oriented system

ZHANG Gui-zhu, XU Wen-bo
(*School of Information Technology, Jiangnan University, Wuxi Jiangsu* 214122, *China*)

Abstract: Taking into account existing software size measures and their defects, Class Point Analysis for estimating the size of object-oriented systems was presented. Two measures were correlatively defined to refine preliminary size estiamtion when more information is available in the development process, as predictor of development effort and management projects. The size estimation process that led to the definition of proposed measures was gived. It turned out to be suitable to capture specific object - oriented feasures, based on design documentation and independent technology.

Key words: object-oriented systems; class point analysis; software size measurement; modeling development effort

0 引言

软件系统规模评估是估算软件项目工作量、编制成本预算、策划合理项目进度的基础。系统规模是软件产品最感兴趣的一个特性,被使用在项目代价/费用的评价模型中,以便预算软件设计和实现所需的开销和费用。因此,一个好的系统规模评价模型是规划软件项目开发的首要任务之一,也是项目成功的核心要素,它将有助于软件开发团队准确把握开发时间、费用分布以及缺陷密度等。

由于面向对象(Objected-Oriented,OO)方法在软件开发中的广泛使用,已促使人们研究 OO 产品的度量,以便在 OO 环境下能够监控、测试和支持软件的开发与维护。至今,已有一些度量被定义用来评估软件的规模。例如,源代码行方法(LOC)和功能点方法(FPS)已经获得广泛的接受,以间接地预测软件开发的代价、费用和周期。但是,现有的软件规模度量是为面向过程的范畴而设计的,已不再适用于捕捉 OO 系统的特性,如类、封装、继承、多态和消息传递等。因而,需要新的适合 OO 系统规模的的评价方法。

为了得到有效的 OO 系统规模的评价方法,文中给出了实施类点法的软件规模评估过程模型,它被结构化五个主要步骤:第一步:分析系统设计规格说明书,识别出系统范围内的所有类的集合,并按类范畴划分成四种系统组件类型:问题域类型、人机交互类型、数据管理类型和任务管理类型;第二步:计算被识别出来的每一个类的复杂度。一个类的复杂度是由类中定义的方法数目、定义的数据属性数目和此类向其他类请求的服务数目决定;第三步:计算未调整的类点数。根据计算得到的每个类的复杂度和系统组件类型,进行复杂度加权因子求和,得到未调整的类点数 CP;第四步:针对 OO 应用环境。考虑 18 个系统特性对系统的影响程度,计算技术复杂度调整因子。第五步:用调整因子对 CP 进行调整,得到调整后的类点数。本文提出的类点分析法的技术,通过经验证明:能够评估软件规模属性,为及早地估计 OO 系统的成本、工作量成为可能,为 OO 开发过程的管理活动提供了基础。

1 现有的软件规模度量方法

目前在软件度量领域已经存在几种不同的方法,以度量软件产品、软件开发过程和相关资源的特征。例如,一个衡量软件规模最常用的概念是 LOC(Line Of Code),LOC 指程序中所有可执行的源代码行数,它计算简洁、通用性强,LOC 的价值和人均月代码行数可以体现一个软件生产组织的生产能力。但 LOC 计算仅能在软件产品得到之后,它不能用来预测软件的规模。软件规模的评估,需要在软件生命周期的早期进行,这推动了软件规模技术的更进一步地研究,功能点分析法 FPA (Function Point Analysis)就是这样的一种研究技术,它是在需求分析阶段基于系统功能的一种软件规模评估方法,用来评估 MIS(Management Information Systems)的规模。FPA 从系统的复杂性和系统的特性这两个角度来度量系统的规模,其特征是:“在外部式样确定的情况下可以度量系统的规模”,“可以从用户角度把握的系统规模进行度量”。经由 ISO 组织已经有多种功能点评估方法成为国际标准,如:

收稿日期:2005 - 09 - 03;**修订日期**:2005 - 10 - 25

作者简介:张桂珠(1962 -) 女,江苏无锡人,副教授,硕士,主要研究方向:软件工程、数据库技术; 须文波(1947 -),男,江苏无锡人,教授,硕士,主要研究方向:计算机应用.

IFPUG 功能点法、Mark II FPA 功能点法、NESMA 功能点法、COSMIC-FFP 方法等,这些方法都属于 Albrech 功能点方法的发展和细化。

FPA 能应用在软件开发的整个过程,即可应用于“需求文档”、“设计文档”、“源代码”、“测试用例”的度量。FPA 用下列步骤完成软件规模度量:

1) 首先识别出系统的所有功能,并确定每个功能的类型。功能元素的类型有五种:外部输入 EI(External Input)、外部输出 EO(External Output)、外部查询 EQ(External Quiries)、内部逻辑文件 ILF(Internal Logical File)和外部接口文件 EIF(External Interface File),前三种功能类型属于事物交易功能,后两种功能类型属于数据的读取和写入功能;

2) 依据预定的标准给每个功能加权值,并求系统五种功能类型的元素加权总和 FP。每个功能的权值由它的复杂度和功能类型决定。例如,一个外部查询如果有多个(≥16)数据类型和至少 2 个以上的文件类型,则被分配一个“高”的复杂度;

3) 考虑 14 个系统环境因子的影响程度,得到一个调整因子乘以 FP,从而得到调整后的 FP,调整的幅度可使调整前的 FP 从 -35% 变化到 35% 的范围。

近几年,FPA 评估已经被越来越多的开发者所认可。基于功能点的规模评估的成功,是由于它能应用在软件开发过程的早期阶段,并且独立于程序设计语言和整个软件生命周期使用的技术或工具。但 FPA 是面向过程时代的特有产物,它将数据和操作相互分隔开,而 OO 的范畴中是将数据和操作密切结合并捆绑在类的概念中。由于 FP 方法不足以适用于面向对象软件的生产力跟踪和工作量预测,于是人们开始研究面向对象软件设计的其他的一些度量方法。一些人对 FP 方法进行了改进,希望它适用于 OO 系统规模度量。例如,Whitmire 提出了 3D 功能点(3D Function Points)中,是将类看作为内部文件,将系统组件之间传递的消息看作为事务,但 3D 功能点需要大量的详细信息以评估软件规模,这使得在软件开发周期的早期预测规模变得非常困难。Sneed 提出了对象点 OP(Object Points)方法并使用在改进的 COCOMO 模型评估技术中,对象点的计数类似于 FPA,但此方法是基于计数处理而引入的一组对象,而这组对象与 OO 范畴中类的对象并不直接关联。Minkiewicz 提出了预言性的对象点 POP(Predictive Object Points)方法,组合了一些良好的 OO 度量,采用了 FPA 的计数模式分析法。POP 的计数是以 WMC(Weighted Methods per Class)为基础应用在每个顶层类上,并结合继承树的平均深度 DIT(Depth of Inheritance Tree)和子类数目 NOC(Number Of Children)平均值。为了计算 WMC 的值,每个方法被加权,并使用方法的类型(构造方法、析构方法、修改方法、读取方法、叠加方法)、方法影响的属性个数和对系统提供的服务数目。POP 法中决定方法复杂度的信息,同样在软件开发过程的早期不可能全部有效地得到。以上这些度量方法都是基于 FPA 标准的扩展,将 FPA 中的概念映射为 OO 中的概念,系统元素类型的分类是基于方法进行的,而在 OO 范畴中,类作为实体,最好是按类范畴进行分类和计数才比较合理。另外,计数所用的信息通常应该在设计阶段能有效地得到,而不是用估算值代替,这需要收集额外的数据和修正原始结果。

2 类点法评价软件规模

类点分析法 CP 对 OO 产品规模提供了系统级的评估。它在 OO 范畴内,对 FPA 进行了改造,并组合了一组良好的 OO 度量套件。类点法中自定义了两个度量 CP1 和 CP2。CP1 度量提供了软件开发过程的初始规模评估,它将类的外部方法和类的请求服务数目包含在内;随着软件开发过程的进行,当计算的数据信息被有效地得到时,CP2 度量对 CP1 度量进一步精化,它使用了类中的数据属性。在类点法中,强调了类作为计数的实体,以类的复杂度和类的类型而被加权。决定每个类的复杂度的信息来自于:类中的方法、此类与系统其他类的交互,并且在类的数据属性能有效地得到时,也将数据属性包含在内。与 FP 方法不同的是:FP 方法中,数据功能类型和事务功能类型的分类是基于事务应用模型,而类点分析法独立于任何应用模型,并且将类按范畴进行了分类,而不是按方法分类;在 CP 法中考虑了数据属性,并且进行计数所需的信息在设计阶段能够被得到。

图 1 给出了用类点法实施软件规模评估的模型,整个过程被结构化为五个步骤。

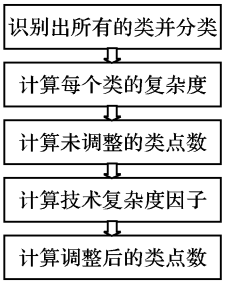

图 1 类点法评价软件规模的模型

2.1 识别出所有的类并进行分类

类点法计数的第一步是分析系统设计规格说明书,识别出系统范围内的所有类的集合,并且将这些类分成四种系统组件类型:

1) 问题域类型 PDT(Problem Domain Type):PDT 包含的类集代表着系统问题域中的现实世界的实体集。

2) 人机交互类型 HIT(Human Interaction Type):HIT 类型的类被用来满足信息的可视化和人机交互的目的。

3) 数据管理类型 DMT(Data Management Type):DMT 类型的类集提供了数据存储和提取的功能。

4) 任务管理类型 TMT(Task Management Type):TMT 类型的类集被设计用于管理任务,它们负责定义和控制任务,负责在不同的子系统中进行通信,以及和外部子系统的通信。

例如:一个典型的面向对象的成绩管理系统中,Student 类、Teacher 类和 Course 类属于 PDT 组件,各种 GUI 类、报表类属于 HIT 组件;发出 SQL 命令访问数据库的类属于 DMT 组件,消息和连接属于典型的 TMT 组件。在任何一个 OO 系统中,都存在这四种类型的组件,组件的分类独立于系统的应用域和使用的设计方法。

2.2 计算每个类的复杂度

在类点法中,通过分析每个类的行为,计算每个类的复杂度。将系统中类的复杂度级别区分为三种:低级、中级和高级。在 CP 计数中,包含了一组良好的 OO 度量,它们是:

1) 类对外提供的方法数目 NEM(Number of External Methods)。NEM 度量了一个类对外接口的大小,它定义为类中 public 型的方法数目;

2) 类请求服务数目 NSR(Number of different Services Requested to other classes)。NSR 度量了系统部件之间的内部连接,一个类的 NSR 定义为此类向其他类请求的不同服务数

目;

3) 类的属性数目 NOA(Number of Attributes)。一个类的 NOA 定义为此类中的数据属性的数目。

CP1 和 CP2 两个度量的区别在于决定类复杂度的方式是有区别的:对于 CP1,一个类的复杂度的计算是基于 NEM 和 NSR 的范围值,定义 CP1 中类的复杂度计算规则如表 1 所示。对于 CP2,NSR 的范围也将考虑在内,定义 CP2 中类的复杂度计算规则如表 2 所示。

表 1　CP1 度量中类的复杂度计算规则

NSR	NEM		
	0 ~ 4	5 ~ 8	≥9
0 ~ 1	低	低	中
2 ~ 3	低	中	高
≥4	中	高	高

表 2　CP2 度量中类的复杂度计算规则

0-2 NSR				3~4 NSR				NSR≥5			
NEM	NOA			NEM	NOA			NEM	NOA		
	0~5	6~9	≥10		0~4	5~8	≥9		0~3	4~7	≥8
0~4	低	低	低	0~3	低	低	中	0~3	低	低	中
5~8	低	中	低	4~7	低	中	高	3~6	低	中	高
≥9	中	高	中	≥8	中	高	中	≥7	中	高	高

2.3　计算未调整的类点数

根据计算得到的每个类的复杂度和每个类所属的系统组件类型,给每个类进行复杂度加权,并计算所有类的复杂度加权因子总和,得到的值称为未调整的类点数 *CP*。为了计算类的复杂度加权因子之和,系统用一个二维矩阵定义四种系统组件按不同复杂度级别(低、中、高) 的加权规则,如表 3 所示。则类点的复杂度加权因子之和 *CP* 的计算公式:

$$CP = \sum_{i=1}^{4}\sum_{j=1}^{3} c_{ij} * w_{ij}$$

其中:i = 组件类型(PDT, HIT, DMT, TMT),j = 类的复杂度级别(低,中,高)

C_{ij} 表示系统组件类型为 i 复杂度级别为 j 的类的集合中元素的计数;

w_{ij} 表示为系统组件类型为 i 复杂度级别为 j 的类所加的权值。

表 3　类点的复杂度加权规则

系统组件类型	复杂度		
	低	中	高
问题域类型 PDT	3	6	10
人机交互类型 HIT	4	7	12
数据管理类型 DMT	5	8	13
任务管理类型 TMT	4	6	9

2.4　计算技术复杂度因子

类点法中,技术复杂度因子是由一组系统特性(或称环境因子)对应用域的影响程度决定,并从设计者角度出发。针对 OO 应用环境,通过考察系统的技术和运行特点,得到技术复杂度因子由 18 个方面的系统特性决定。它们是:

1) 数据通信、分布式功能、性能、硬件负荷、事务频度、联机数据输入、界面复杂度、联机更新、内部处理复杂度、代码复用性、易安装性、易操作性、多站点、支持可变更性;

2) 用户可适应性、快速系统原型、多用户的交互性、与用户相关的多个接口。

前面的 14 个系统特性与 FP 方法中定义的一组系统特性相同,后面的 4 个系统特性是针对 OO 交互环境特点而增加的,因为它们显著地影响了 OO 交互系统的开发代价。

设第 i 个环境因子对系统的影响程度定义为 Di,Di 范围值从 0 到 5,其值含义如下:0(不影响)、1(轻微影响)、2(温和影响)、3(中等程度影响)、4(较显著影响)、5(巨大影响)。则 18 个环境因子对系统总的影响程度为 TDI(Total Degree of Influence),TDI 的计算公式如下:

$$TDI = \sum_{i=1}^{18} Di$$

由 *TDI* 的值可得到计算技术复杂度因子 TCF(Techinical Complexity Factor) 的公式如下:

$$TCF = 0.55 + (0.01 * TDI)$$

TCF 也称为调整因子,用于对前面计算得到的未调整的类点数 *CP* 进行调整。

2.5　计算调整后的类点数

类点法计数的最后一步是用技术复杂度调整因子 TCF 对未调整的类点数 CP 进行调整,得到调整后的类点数 ACP (Adjusted Class Points)。ACP 可由下列计算公式得到:

$$ACP = TCF * CP$$

从公式中可知,ACP 可使调整前的 CP 的变化幅度从 -45%(对应 18 个系统特性的影响成度取值为 0)变化到 45%(对应 18 个环境因子影响程度取值为 5)。

调整后的类点数 ACP, 作为在预定期限内交付 OO 产品的规模度量值,它是软件项目估算的一项重要内容。OO 软件项目估算的另一项重要内容是获得交付 00 产品的生产率(或称做开发每个类点所需要的时间),可以根据历史数据计算或使用行业标准数据,通常表示为类点/小时(CPs/Hr)。我们可以利用它来计算同类项目的开发工作量,即:

项目总工时数 = 类点规模度量值 ACP * (CPs/Hr)。

3　结语

类点分析法从用户的角度按类实体来表达开发的 OO 产品,并独立于所采用的技术或工具。它能准确度量 OO 系统规模,是项目估计、变更管理、生产率度量和沟通系统功能的有效技术。它已成功地用于多个 OO 软件项目的评估,例如网上学籍管理系统和服装设计系统。经验证明,在类点分析法中,在软件的设计阶段应用 CP1 和 CP2 两个度量,CP1 是用于软件开发过程的初期规模度量,当计算所用的数据信息在开发阶段的后序阶段得到时,用 CP2 度量精化 CP1 度量信息,使 OO 软件的规模评估不断细化和完善。随着计算机环境复杂性的增加,类点分析法将成为准确评价和反映我们所开发、维护的 OO 软件系统的优良工具。

参考文献:

[1] Chidamber, A Metric Suite for Object Orinted Design [J]. IEEE Trans. Software Eng., 1994.

[2] Bansiya, A Hierarchical Model For Quality Assessment of Object-oriented Design[D], PHD Dissertation, Univ. of Alabama in Huntsville, 1997.

[3] KOKOL P. A Software Metric Tool Generator[C]. ACM SIGPLAN Notices, 1995.

[4] BASILI VR, BRIAND L. A Validation of Object - orinted Design Metrics as Quality Indicators[J]. IEEE Trans. Software Eng., 1996.

[5] 张桂珠. 面向对象软件度量技术[J]. 计算机工程, 2003, 12(4): 40 - 43.