基于UML模型的全功能点自动化度量*)

程莉莉　刘宗田

(上海大学计算机工程与科学学院　上海200072)

摘　要　软件规模在项目估算和决策中起着举足轻重的作用。基于软件需求从用户角度和功能角度度量规模的全功能点(FFP)广受欢迎,但为手工操作,弊端显而易见,所以亟需自动化度量。本文提出并实现了基于UML模型全功能点自动化度量系统,不但能得到精确的量化结果,而且能减轻工作量、降低花费,以及很大程度上避免抽取数据及度量过程中差错所造成的风险,更重要的是具有可重复性,从而易于功能规模度量方法的推广。

关键词　规模度量,全功能点,统一建模语言,自动化

The Automatic Measurement of Full Function Points Based on UML

CHENG Li-Li　LIU Zong-Tian

(Department of Computer Engineering and Science, Shanghai University, Shanghai　200072)

Abstract　Software size plays a significant role in project evaluation and decision-making. Full Function Points(FFP) which give a measure of the size of the system by measuring the functionalities from an end-use perspective and a functional perspective ,are gaining an increasingly widespread application. Unfortunately FFP calculation is mostly a manual process ,so many abuses obviously occur and automatic measurement is demanded very much. This paper presents and implements an automated system for measuring the FFP software metric based on UML. Automation not only gains the more accurate of its size measurement but also lessens the work effort and reduces measurement costs of FFP counts. Secondly automation provides a good opportunity to reduces the risk of errors being introduced into the extracted data ,and the most important thing is it makes repeatability of the results. Thus is important for the acceptance of functional size measurement.

Keywords　Size measurement, FFP, UML, Automation

1　引言

随着计算机技术的日新月异,软件规模的扩张也达到了叹为观止的速度。规模是软件项目可量化的结果,是软件的一个重要属性。在软件日益庞大的今天,对规模的估算误差和决策差错都将会造成盈利及亏损间的天壤之别,因此作为项目估算重要参数和决策重要依据的软件规模至关重要。

对于软件开发和项目管理而言,在开发的早期进行规模估算的要求非常迫切,但这时有关软件的信息还很少,没有编码可供度量。为了解决开发初期进行规模度量的迫切要求,出现了试图基于软件需求从用户角度和功能角度度量规模的功能点,来估计未来系统的大小。功能点最大程度地突破了传统评估方法的局限,能够不依赖于外部条件,客观、公正、独立地评估系统的规模,因而广受欢迎并且得到广泛的应用。

遗憾的是,功能规模度量为手工操作,不仅抽取数据及度量过程非常繁琐、容易发生差错,而且度量人员不同,工作方法亦不同,带有很强的主观性,结论很难具有可重复性,所以亟需自动化地度量。UML是一种总结了以往建模技术的经验并吸收当今优秀成果的标准建模方法,它的提出和普遍应用,为客观、可比较、自动化地度量功能点提供了可能性,而且用例已经成为捕获用户需求的一个通用方法,它们能比较容易地转换成功能点。

2　FFP功能点概述

功能点是一种间接的软件规模的测量,基于应用软件的外部、内部特性以及软件性能,使用软件所提供的功能的测量作为规范化值。IFPUG功能点[1]在MIS中应用得最为广泛,有很多实际经验可借鉴和许多历史数据库可利用,但是,此功能点固有的缺陷与不足限制了它的使用。之后不少研究者提出了很多不同扩展的功能点解决方法。其中的全功能点[2](Full Function Point)版本2最具影响力,最初由St. Pierre等人1997年提出,之后2001年COSMIC发布了版本2。FFP吸取以前各种方法的精华,旨在覆盖所有领域,包括MIS、实时、技术和系统软件,是一种非常有前景和潜力的方法。它基于一种全新理念,假设:数据传送代表系统的规模,一个系统的规模就是所有数据传送的总合,从而通过分析用户需求决定软件规模。

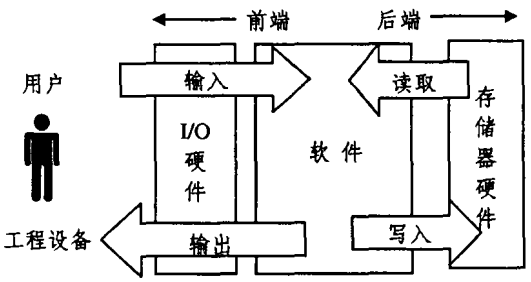

图1　数据项的功能流向

如图1所示,数据传送有四种类型:输入(entry)、输出(exit)、读取(read)和写入(write)。其中输入和输出表示在前

*)基金项目:上海市科技发展基金项目(编号025115035)资助和上海市高等学校科学技术发展基金重点项目(编号02AZ86)资助。程莉莉　硕士研究生,研究方向:软件工程;刘宗田　博士生导师,研究方向:软件工程、人工智能、多媒体技术等。

端通过像鼠标、键盘、打印机、显示器等I/O硬件与用户交换数据;读取和写入表示在后端与像硬盘、RAM和ROM等存储器交换数据。FFP还添加了一个重要的新概念:层,层是软件环境按照功能划分的结果,这样软件就能够被分成几个层次,分别代表软件的不同视图,这样就有可能度量分布式和复杂系统。一个原子数据传送定义为1Cfsu(Cosmic Functional Size Unit)规模,那么将属于同一抽象层的四种表征数据项的功能流向的数据传送各自的规模进行聚合就可以得到这个软件系统的规模。

3 UML模型概述

统一建模语言[3](Unified Modeling Language)是一个用于对软件进行描述、可视化处理构造和建立软件系统制品的建模语言,适用于各种软件开发方法、软件生命周期的各个阶段、各种应用领域以及各种开发工具,因此得到了工业界的广泛支持,1997年由OMG组织(Object Management Group)采纳作为行业标准。正是UML的普遍应用,为客观、可比较、自动化地度量功能点提供了可能性。这种建模语言将系统描述为一些离散的相互作用的对象并最终为外部用户提供一定功能的模型结构,包括系统的静态结构和动态行为。UML提供各种图来描述系统的结构和行为。用例图(Use case diagram)描述系统总的功能。类图(Class diagram)最适合描述系统的静态结构:类、对象以及它们之间的关系。而序列图(Sequence diagram)、协作图(State diagram)、状态图和活动图则适合于描述系统的动态行为,即描述系统中的对象在执行期间不同的时间点是如何动态交互的。

4 FFP概念与UML模型的映射

自动化度量的关键问题是把FFP的概念和定义应用到软件的UML模型的表示上。通过分析研究,FFP功能点度量主要应用UML模型中的用例图、类图、交互图(顺序图或者协作图)中的用例、参与者、消息等元素。

表1 FFP概念与UML模型映射关系

编号	FFP概念	UML模型
1	边界	参与者集合
2	层	同抽象层上用例集合
3	功能过程	用例
4	数据组	类
5	数据项	类的属性
6	子过程	消息
7	输入	参与者发送的消息
8	输出	参与者接收的消息
9	读取	Get版型消息
10	写入	Set版型消息
11	读/写	其它版型消息

总结FFP概念与UML模型之间的对应关系如表1,具体解释如下:

(1)边界==参与者集合

[FFP]一块软件的边界(Boundary)是以用户的角度从外面观察到的,这块软件概念上与它操作的环境之间的边境。

[UML]参与者是直接与系统相互作用的系统、子系统或类的外部实体的抽象概念。

既然参与者是UML模型用例图中标识出的系统之外的所有元素,确定参与者集合也就无歧义地区分待度量的软件内部和外部的操作环境各包含什么,即确定了在FFP概念中待度量软件的边界。

(2)层==同抽象层上用例集合

(3)功能过程==用例

[FFP]层(layer)是软件环境的功能分解结果,使包含的所有功能过程执行在相同级别的抽象层上。一个功能过程(Functional process)是实现一个内聚的、逻辑上不能分割的用户功能需求集合。

[UML]用例视图是被称为参与者的外部用户所能观察到的系统功能的模型图。其中的用例就是外部可见的系统中的一个功能单元,这些功能由系统单元所提供。

UML模型的用例视图中的每个用例就描述了FFP概念中系统的一个功能过程,因此相同抽象层上用例的集合也就表示了一个FFP概念中的层。

(4)数据组==类

(5)数据项==类的属性

[FFP]一个数据组(data group)是一个独特的、非空的、无序的、非冗余的数据项集合,其中所包含的每一个数据项(data attribute)描述相同对象的一个侧面,是数据组内含有意义的最小信息单元。

[UML]静态视图对应用领域中的概念以及与系统实现有关的内部概念建模,主要由类及类间相互关系构成,由类图来实现。类图中的对象类是描述具有相似特性(属性)的一组对象。属性是一个类中对象所具有的逻辑数据值。

显而易见UML模型的类图中的类和类的属性分别等价于FFP概念中的数据组和数据项。

(6)子过程==消息

(7)输入==参与者发送的消息

(8)输出==参与者接收的消息

(9)读取==Get版型消息

(10)写入==Set版型消息

(11)读/写==其它版型消息

[FFP]子过程(sub process)是一个功能过程执行当中发生的一个数据传送,等价于ISO基本功能部件类型。子过程只表达用户功能需求,不包含质量和技术需要,所以只包含特定相联系的数据操作,有四种类型:输入(entry)、输出(exit)、读取(read)和写入(write)。一个数据组中的数据项从软件边界的用户边到软件内部的一个传送定义为输入,其中不更新传送的数据。相反,一个数据组中的数据项从软件边界的内部到软件用户边的一个传送定义为输出,其中不读取传送的数据。而读取是引用一个数据组中含有的数据项。相反,写入就是提交一个数据组中含有的数据项。

[UML]相当于功能过程的用例是使用系统与一个或多个参与者之间的一系列消息来描述系统中的交互作用。交互视图就描述了执行系统功能的各个角色之间相互传递消息的顺序关系,其中的消息是从一个发送者到另一个或几个其他接收者的发送信号,或由一个发送者或调用者调用另一个接收者的操作。

UML模型的交互视图中由参与者发送的消息表示在功能上从软件边界的用户边将数据传送到它所属的功能过程内部,即为FFP概念中的输入而由参与者接收的消息表示在功能上将数据从它所属的功能过程内部发送到边界的用户边,即为输出;类似地,Get版型消息表示在功能上将数据从存储器带到它所属的功能过程内部,即为读取;Set版型消息表示在功能上将数据从它所属的功能过程内部传送到存储器,即为写入;其余版型的消息则有可能是传入存储器或传出或者既传入又传出数据,所以标识为读/写。

5 自动度量系统的实现

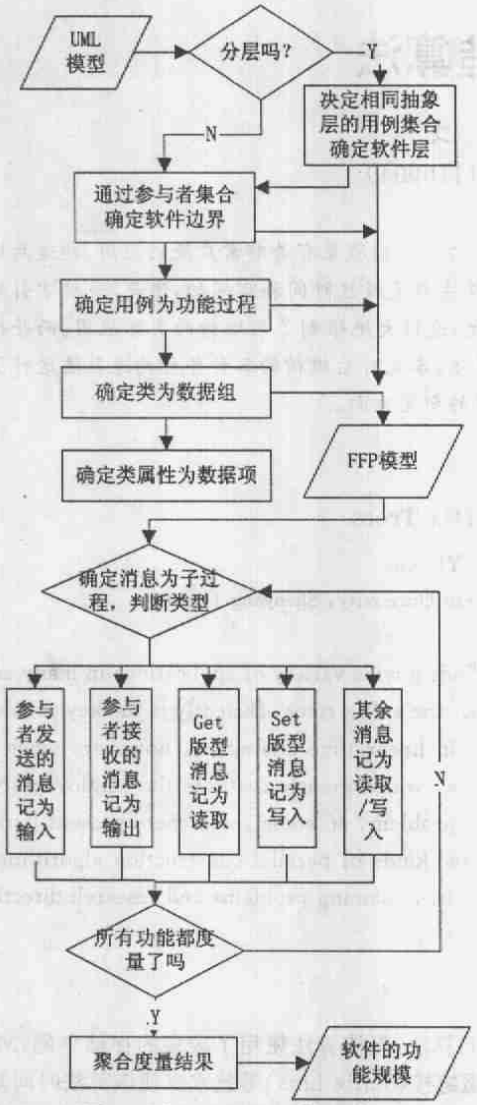

图2 COSMIC FFP 简要流程

图3 FFP 度量结果

整个度量系统的简要流程如图2。首先读取 UML 模型中的信息，按照上面所述将 UML 模型中的参与者、用例、类、消息等映射为 FFP 模型中的层、功能过程、数据组、数据传送子过程等。然后将子过程(输入、输出、读取或者写入)的每一个实例记为1 Cfsu 规模。最后在确定的抽象层内，对于每一个功能过程中确定的每一个子过程的功能规模值按照下列公式 $Size(layer,i)FFP. = \Sigma size(entries) + \Sigma size(exits) + \Sigma size(reads) + \Sigma size(writes)$，进行算数相加获得一个单一的功能规模值，即为所要求的软件系统抽象层的规模。

图3是该系统对一个 UML 模型中的 sale 用例的 FFP 度量结果示例，因篇幅的限制，UML 图就不显示了。

6 规模度量的应用

(1)在开发前，将规模作为估算模型的输入值可以得到项目开发的成本、工作量和持续时间等，从而更有效地控制软件开发过程，防止"救火"情况的发生。COCOMO II 是最著名的估算模型，例如它的工作量的导出公式为：$PM = A * (SIZE)^B$，其中 SIZE 是将功能点转化为代码行数。

(2)测试方法、测试用例、测试和维护人员个数等都与软件规模有着密切的联系，可以根据软件规模，对测试和维护工作做出决策方案。例如规模为100至1000个功能点的新软件一般只需采用子程序测试、单元测试和新功能测试这三种测试方案。

(3)收集规模值，建立宝贵的历史经验数据库，因为如果没有历史数据存在，成本等各种估算也就像海市蜃楼，有了具完整性和准确度的历史数据库就可以评估一些在应用领域、环境和复杂度上与历史项目相似的项目，通过新项目与历史项目的比较得到高精度可信赖的的规模估计。

(4)规模度量的准确性依赖系统描述的准确性及完整性，开发前较之开发后所需的信息量非常少，对比开发前的规模估算值与开发后的规模测度值，找出偏差的原因，减少以后的估算误差。

(5)分析规模数据和由此导出的生产率等，并与过去的平均值进行比较，评估和确定软件质量是否已有提高，项目和组织的生产率是否已有明显改善。

结论 FFP 自动化度量系统具有以下优势。首先 FFP 方法本身与程序设计语言无关，既可以用于传统的语言，也可用于非过程的语言；采用了层概念，可以对软件的各部分分别进行度量；度量出来的结果可以在不同的开发方法之间进行比较；在项目的开发初期就可以利用需求分析模型进行功能点估算，对项目开发工作具有指导作用，也可以在整个软件生命周期中使用。其次自动化地度量更是锦上添花：使得数据收集不但能得到比较精确的量化结果，而且减轻了工作量，降低花费，以及很大程度上避免了抽取数据过程中差错造成的风险，更重要的是具有可重复性，从而易于功能规模度量方法的推广；但是因为计算是基于 UML 模型，它描述的粗细粒度对度量的结果有很大的影响。

参 考 文 献

1 IFPUG. Function Point Counting Practices Manual, Release 4.1. International Function PointUser Group, 1999

2 Abran A, Desharnais J M. SergeOligny, Charles Symons "COSMIC FFPMeasurement Manual Version 2.1", Common Software Measurement International Consortium, 2001

3 Rumbaugh J, et al. The Unified Modeling Language Reference Manual. Addison Wesley LongMan, 1999

4 Albrecht A J. Measuring Application Development Productivity. In: Proc. IBM Application Development Symposium, Monterey, CA, Oct. 1979. 83～92

5 Caldiera G, et al. Estimating Size and Effort for Object Oriented Systems. In: Proc. 4th Australian Conf. on Software Metrics, 1997