UML 分析模型中功能点计算的探讨*)

Function Point Analysis in the UML Analysis Model

林扬帆　李师贤

(中山大学计算机科学系　广州510275)

Abstract Function point analysis is a method used to measure an application's functional size, it is independent to the implementation programming language, it's measuring result can be compared between different development processes. UML is a standard modeling language used to OO amalysis and OO design, this paper describes a method to use the UML anslysis model to analysis the application's function points So the project manager can use it to estimate the project's size and cost in the early development

Keywords Function point analysis, UML, Software estimate, Requirement analysis model

功能点分析是从用户的角度去度量一个应用程序大小的一种方法，与代码行(LOC 或 KLOC)度量不同，它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，度量出来的结果可以在不同的开发方法之间进行比较，而且由于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，因而功能点分析广受欢迎，并且得到广泛的应用。

UML 是一种定义良好、易于表达、功能强大且普遍适用的标准建模语言，容易被大众接受和采用。目前，UML 在国外和国内都已经得到比较广泛的应用，并在未来较长的一段时期内将是软件开发建模时采用的一种标准。

在计算一个系统所要实现的功能点时，按传统的方法一般需要系统的数据流程图，在利用 UML 进行建模和表示的软件开发中，是否可以利用 UML 的分析模型来直接计算出系统要实现的功能点呢?Thomas 在文[1]中介绍了将 Jacobson 的 OOSE 方法映射到功能点分析的规则，该规则使我们可以直接利用 OOSE 方法中的分析模型进行功能点计算，然而这些规则具有某些局限性：(1)这些规则并没有给出很具体的操作方法，不利于实现功能点的自动化计算；(2)UML 是目前面向对象设计广泛采用的建模语言，要将这些规则用于 UML 分析模型，必须进行修改和扩充；(3)这些规则没有考虑到控制和算法较复杂的系统的情形。本文在此基础上提出了一种利用 UML 分析模型进行功能点分析的方法。该方法可以适用于控制复杂的系统，并有利于实现功能点计算的自动化。

1　功能点分析简介

1.1　功能点分析的五个功能域

功能点分析使用软件所提供的功能的测量作为规范值，它是基于软件信息领域的可计算的测量及软件复杂性的评估而导出的。功能点计算涉及到五个功能域：内部逻辑文件(ILF)、外部接口文件(EIF)、外部输入(EI)、外部输出(EO)和外部查询(EQ)。国际功能点用户组织(IFPUG)在 IFPUG 操作手册4.0版本中关于这五个功能域的定义见文[2]。

这五个功能域可以分成两类：事务(transaction)和文件(file)，事务包括外部输入、外部输出和外部查询；文件包括内部逻辑文件和外部接口文件。

1.2　功能域的复杂度值和复杂度加权因子

每个功能域的复杂度值分为低、平均和高。文件的复杂度值由 DET(数据元素类型)和 RET(记录元素类型)决定，而事务的复杂度值由 DET 和 FTR(引用文件类型)决定(参见文[3])。DET、RET 和 FTR 的定义如下：

DET：是数据的一次唯一出现，也可以认为是一个数据元素、一个变量或一个字段。

RET：是一个 ILF 或 ETF 内的一个唯一的记录格式，一个 RET 可以认为是一个文件内一组相类似的记录。

FTR：指被一个事务读取、建立或更新的 ILF 和(或)EIF 的个数。一个 FTR 可以被认为是对一组相似数据的一次引用。

*)本文得到 NSFC 与 RGC 联合科研资助基金资助(批准号：79910161989)和广东省现代控制技术重点实验室资助。

每个功能域对应不同复杂度值具有不同的复杂度加权因子，见文[3]。

1.3 计算功能点的步骤

计算功能点的步骤如下：

(1)定义要度量的系统的边界；

(2)计算 ILF、EIF、EI、EO 和 EQ 出现的次数；

(3)计算出每个功能域每次出现的复杂度值；

(4)由复杂度值确定每个功能域出现的权重；

(5)计算出未校正的功能点计数(UFP)；

$$UFP=\sum_{i}^{5}\sum_{j}^{3}N_{ij}W_{ij}$$

其中 N_{ij} 表示第 i 个功能域在复杂度值 j 的计数，W_{ij} 表示第 i 个功能域在复杂度值 j 时的加权因子。

(6)确立14个校正因子(F_i，i=1，…，14)的值；

(7)计算出校正后的功能点计数(FP)。

$$FP=UFP\times(0.65+0.01\times\sum F_i)(i=1,\cdots,14)$$

2 UML 需求分析模型中功能点的计算

2.1 UML 需求分析模型

利用 UML 进行建模的软件开发过程一般包括四个阶段[4]：开始(Inception)、细化(Elaboration)、构建(Construction)和移交(Transition)。开始阶段主要是获得应用的业务用例和系统的基本架构；在细化阶段，产品中的大部分用例被详细地定义下来，系统基本架构也被设计出来，项目管理员一般要在细化阶段的末期计划活动和估算完成该项目所需要的资源；构建阶段主要是建造产品；移交阶段主要是发布产品的测试版。

利用 UML 建模，在用户需求阶段获得的是用例模型[5]，包括用例图、顺序图(Sequence Diagram)、状态图和活动图。然后对用例模型进行细化和分析，可以进一步得到系统的分析模型，包括类图、顺序图、合作图、状态图和活动图。

2.2 功能点计算

2.2.1 *系统边界的确立* 用例图描述了系统外部的执行者与系统提供的用例之间的某种联系。用例是指对系统提供的功能的一种描述，执行者是那些可能使用这些用例的人或者外部系统。因此，我们通过用例图来确定系统的边界，每个用例的执行者都在系统边界外，而每个用例都在系统的边界内。

2.2.2 *内部逻辑文件(ILF)和外部接口文件(EIF)数的计算* 内部逻辑文件和外部接口文件的计数主要由 UML 模型中的类图来计算。类图描述了系统中的类及其相互之间的各种关系，这些类在分析模型(交互图)中按 UML 标准划分为三种版型(Stereotypes)[6]：边界类(Boundary Class)，控制类(Control Class)和实体类(Entity Class)，它们的表示符号见图1。

图1 UML 三种标准的类版型

边界类用来建立系统和执行者之间相互交互的关系，位于系统与外界的交界处，包括所有窗口、报表、与打印机等硬件的接口和与外部系统的接口等，每个边界类至少与一个执行者相关。实体类保存持久储存的信息，通常表示一个逻辑的数据结构，有利于理解系统所需要的信息。控制类负责协调其它类的工作，每个用例通常有一个控制类，控制用例中事件的顺序。使用控制对象的好处是能够将业务逻辑和程序逻辑分开，如果程序需要改变，只影响控制对象。

文件(ILF 和 EIF)的计数主要由实体类来计算，但并非每一个实体类都计算为一个文件。类之间可能存在聚集关系和继承关系。对于聚集关系，聚集类实际上相当于被聚集类的一个逻辑结构，因此应将聚集类计算为被聚集类的一个 RET。对于继承关系，没有实例化的抽象类对于用户是不可见的，不应将它计算为一个文件，但它定义了继承它的具体类(有实例化的类)的一个逻辑结构，因此应计算为具体类的一个 RET。

类图中的每一个实体类将计算为一个文件，除了以下的两种情形：(1)该类是另外一个类的一部分(即该类聚集成另一个类)，这种情形将该类作为它聚集最高层的类的一个 RET；(2)该类是抽象类，则将抽象类作为继承它的每一个具有实例化子类的一个 RET。

文件的类型根据该文件具体的描述和该文件的用途，以及内部逻辑文件和外部接口文件的具体定义来确定。

控制类一般不用来计算内部逻辑文件和外部接口文件，但在比较复杂的系统(例如实时控制系统和嵌入式系统等)中可以作为计算其它特征的参考。边界类主要用来帮助挑选出事务和确定事务的复杂度。

2.2.3 *外部输入、外部输出和外部查询数的计算* 用例描述了系统提供的功能，因此事务(外部输入、外部输出和外部查询)的个数可以由用例图来确定。一个用例对应一至若干个事务，具体的个数必须通过该用例的交互图(Interaction Diagram)来计算。交互图常用来描述一个用例的行为，显示该用例中所涉及的对象和这些对象之间的消息传递情况，它有两种形式：顺序图和合作图，顺序图和合作图之间可以相互转换[7]，因为它们记录同样的信息，顺序图着重体现对象

间消息传递的时间顺序，而合作图侧重于说明哪些对象之间有消息传递。

事务的个数主要由合作图（或顺序图）中由执行者发送给边界类的消息和由边界类发送给执行者的消息来计算，但并非每个消息都计算为一个事务，具体的计算规则为：(1)执行者发送给边界类的消息可能是一个外部输入；(2)边界类发送给执行者的消息可能是一个外部输出；(3)而执行者发送给边界类并且要求有反馈的消息可能是一个外部查询；(4)其它情形的消息不当作事务来计算；(5)由(1)-(3)得到的消息是否计算为一个事务还必须根据该消息的具体描述和相关事务（外部输入、外部输出和外部查询）的定义来确定。

所有用例的外部输入、外部输出和外部查询个数的总和便是整个系统的外部输入、外部输出和外部查询总个数。

2.2.4 *复杂度的计算* 一个文件的DET相当于该文件对应类的属性，一个文件的DET计数等于该类所有属性的计数，如果该类包含聚集类，则还要加上每一聚集类的属性计数。一个文件的RET计数至少是1，如果该文件对应的类包含聚集类，或者该类是某个抽象类的后代，则还要加上按2.2.2中(1)和(2)的情形计算出的RET计数。

在每个用例对应的交互图中，一个事务的FTR计数相当于与该事务直接相关联的对象的个数，一个事务的DET计数相当于该事务所要输入或(和)输出的数据项的计数及控制信息的计数。计算可以根据有关用例和交互图的详细描述进行计算。

根据每个功能域的RET(或FTR)和DET个数可以确定其相应的复杂度，将所有功能域的复杂度加权因子求和便可以得到这个系统的未经校正的功能点计数。

3 复杂系统功能点计算的考虑

功能点度量主要用于商业信息系统应用软件中，但不适合于算法复杂性较高的应用程序。为解决这一问题，不少功能点研究者提出了不同扩展的功能点解决方法，这些方法包括Jones的特征点、Symons的Mark-II、Reifer的Asset-R和Whitmire的3D功能点方法等。在这里，仅讨论3D功能点方法。

3D功能点方法将软件的数据维、控制维和功能维集成起来考虑，以提供一个面向功能的测量[6]。数据维仍按功能点分析的方法进行计算，控制维的测量是计算状态之间的变迁数，功能维的测量是考虑把输入变换成输出数据所需要的内部操作数。一个变换被视为一系列由一组语义表示的约束的加工步骤。一般情况下，一个变换是由一个算法来完成，在处理输入数据并将其变换成输出数据的过程中，它导致输入数据的根本改变。每个变换的复杂度值是加工步骤和控制加工步骤的语义语句的一个函数。不同复杂度值的变换具有不同的加权因子。

在UML分析模型中，一个用例的变迁数可以通过用例对应的交互图来计算，交互图中对象间发送消息可以看作是状态间的转移，交互图中所有对象间发送的消息总数便是该用例的变迁数。

变换主要对应于类的方法，但并不是类的每个方法都计算为一个变换。如果类的一个方法将输入数据（相当于调用方法的参数）变换成输出数据，并且已经导致输入数据的根本改变，那么可以将该方法计算为一个变换。变换的加工步骤和语义语句可以根据其对应方法的算法描述或程序流程图进行计算。

计算3D功能点时，除了要计算1中数据维的功能点计数外，还要加上功能维和控制维的功能点计数，计算公式如下：

$$UFP_{3D}=UFP+T+R$$

其中UFP是数据维的功能点计数值，T是所有用例的变迁数，R是变换的复杂度加权值。

4 功能点自动计算工具的实现

建立UML分析模型是系统开发人员的工作，而项目的成本估算则是项目管理员的工作。利用UML分析模型按上面的方法还不能自动估算出项目的功能点数。为了实现自动化估算工具，项目管理员必须对开发人员建立的UML分析模型进行适当的处理。这些处理包括：

(1)在确立系统的边界时，将用例图中的执行者设置为两种版型：用户和外部系统；

(2)如果类图中的类还没有设置版型，则将类的版型根据实际情况设置为边界类、控制类或实体类中的一种；

(3)根据实体类的使用性质，在类描述模板中说明该类是ILF或EIF；

(4)对交互图中的消息设置成以下版型之一：EI、EO、EQ或其它，并在消息的描述模板中增加DET和RET(或FTR)的内容。

项目管理者不一定需要完成以上全部的工作，一些开发工具(如Rational公司的ROSE)已经提供以上工作的部分支持，开发人员在建立模型时可能就已经设置好有关类、信息、用例等的版型及相关的模板描述。将经过以上处理的UML分析模型作为估算工具的输入便可以实现功能点的自动计算。下面给出功能点计算的算法(不包括3D功能点中控制维和功能维的计算)：

算法　利用 UML 分析模型计算功能点

1.[确立系统的边界]
循环　对每一个用例图
(1)将用例添加到系统用例集;
(2)将执行者添加到执行者集;
2.[计算文件的个数]
循环　对类图中的每个类
(1)若该类是实体类,并且不是其它类的聚集类,也不是抽象类,则若该类的模板说明是 ILF,则将该类添加到 ILF 集;否则将该类添加到 EIF 集;
3.[计算事务的个数]
循环　对步骤1确定用例集中的每个用例
1)找出该用例对应的交互图
2)循环　对交互图中的每个消息
①如果该消息添加到 EI,则将该消息添加到 EI 集;
②如果该消息的版型是 EO,则将该消息添加到 EO 集;
③如果该消息的版型是 EQ,则将该消息添加到 EQ 集;
4.[计算文件的复杂度]
1)循环　对步骤2确定 ILF 集中的每个类
①将该类的每个属性计算为该类的一个 DET;
②将聚集成该类的每个子类(包括子类的聚集后代类)计算为该类的一个 RET;
③由该类的 RET 和 DET 的个数确定该类对应 ILF 的复杂度值和加权因子;
2)循环　对步骤2确定 EIF 集中的每个类
①将该类的每个属性计算为该类的一个 DET;
②将该类的每个祖先类计算为该类的一个 RET;
③由该类的 RET 和 DET 的个数确定该类对应 EIF 的复杂度值和加权因子;
5.[计算事务的复杂度]
循环　对步骤3确定的 EI、EO 和 EQ 集中的每个消息,由该消息模板说明中的 FTR 和 DET 的个数确定该消息对应事务的复杂度值和加权因子;
6.[计算出系统的功能点计数]
对步骤2和步骤3确定的每个功能域,将其复杂度加权因子进行求和,得到的值便是未校正的功能点计数。

5　应用

将该方法用到早期的分析模型上,我们可以估算出项目的大小,将估算值作为 COCOMO2.0模型的输入值可以估算出项目开发的成本和持续时间等。而将该方法应用到后期的设计模型上,我们可以收集本次功能点的度量数据,以找出本次估算偏差的原因,并为以后的项目估算提供历史经验数据。

功能点估算的准确性依赖系统需求的准确性及完整性,因此本方法计算结果的准确性也依赖于 UML 分析模型的准确性和完整性。

参 考 文 献

1　Fetcke T. Mapping the OO-Jacobson Approach into Function Point Analysis, 1998. http://user.cs.tu-erlin.de/~fetcke/papers
2　David, Improved Function Point Definitions. http://www.ifpug.com/Articles/index.htm
3　Ashley N. Measurement as a powerful software management tool. McGraw-Hill International(UK)Ltd. 1995
4　Cantor M. Object-Oriented Project Management with UML. Wiley, 1998
5　Jacobson I. Applying UML in The Unified Process. http://www.umlchina.com
6　Jacobson I, Booch G, Rumbaugh J. The Unified Software Development Process. Addison Wesley, 1998
7　邱仲潘等译. UML with Rational Rose 从入门到精通. 电子工业出版社, 2000
8　黄柏素, 梅宏译. 软件工程——实践者的研究方法. 机械工业出版社, 1999
9　刘超, 张莉. 可视化面向对象建模技术——标准建模语言 UML 教程. 北京航空航天大学出版社, 1999

(上接第95页)

通过浏览器查看各种统计报表。

结论　Java 依靠其虚拟机及预编译字节码等技术,能创建跨平台的应用。XML 走的是另一条路,它发展了一种对象传输协议,将基于网络的信息标准化,使得开发者和电脑易于辨认信息。XML 能创建不依赖于平台、语言或限制性格式化协定的开放数据,XML 得到了微软公司的大力支持。

XML 使文档数据库化,为文本文件、便于程序自动生成、提取数据、网络传输。

XML 不能代替数据库,当数据量大时,XML 文档将变得很大或文件将会很多,这时管理效率将成问题。当数据量少时,可代替部分数据库。其实 XML 和数据库侧重点不同,数据库侧重数据管理,而 XML 则侧重于提供不依赖于平台、不依赖于语言的开放数据。

一般认为 XML 和 JAVA 也不冲突,在很多方面增强了 Java,Java 能创建不依赖于平台的应用(是语言级的),而 XML 能提供(创建)不依赖于平台的数据(是数据级的)。

由于 XML 的优越性能,它将成为异构环境下的通用语言(数据或协议转换);在 Web 领域,它将弥补 HTML 的不足(在短时间内不能代替 HTML,由于 HTML 简单易用,非常适于信息发布,而 XML 更注重数据内容(含义)表示),所以 XML 在 Web 领域将会有更多的应用。由于 XML 的加入,Web 将变得功能更强大、更完善,这时 Web 数据将变得容易管理。

XML 技术虽然发展迅猛,但其应用目前仍处于初级阶段。相信在近几年内将会出现大量的不同领域的各种应用。

参 考 文 献

1　Goldfarb C F,张利,等译. XML 实用技术. 北京:清华大学出版社,1999
2　冯延晖,等. XML 完全手册. 北京:中国电力出版社,2000
3　王海燕,孟小峰,王珊. 基于 XML 的 Web 信息查询系统 XWIS:结构与实现. 计算机科学,2000,27(10.增刊)