

# 利用 SCM 工具辅助功能点分析

李聪廷, 鱼 滨

(西北大学 计算机系, 陕西 西安 710069)

**摘 要:**在软件开发过程中,要管理、监控项目进展,很重要的一点就是要对软件的规模及生产率进行量化。介绍了常用的测量软件规模和生产率的功能点方法,及如何利用 SCM 工具中提取的元数据来辅助软件功能点分析,并介绍了如何利用程序实现该方法。该方法可以提高功能点分析法的数据来源可靠性、实时性,而且当软件功能增加或减少时能及时反应,使分析人员能做出相应调整。通过改进功能点分析出发,进而能改进基于功能点分析的软件测量。

**关键词:**功能点;软件配置管理;元数据;软件测量

**中图分类号:**TP311.5

**文献标识码:**A

**文章编号:**1673-629X(2006)05-0166-04

## SCM Tools - Aided Function Point Analysis Method

LI Cong-ting, YU Bin

(Computer Science Department, Northwest University, Xi'an 710069, China)

**Abstract:** In the process of software development, if want to manage and conduct your project, have to quantify the scale and productivity. First of all, this paper introduces the function point analysis (FPA) method, with which to measure the scale and productivity. Then it describes how to utilize the metadata which extracted from SCM tools to aid the FPA and its implementation. This method could improve the reliability and real-time of the data of FPA. Furthermore, it reflects quickly when the functions of software grow or shrink. Then analysts could recalculate the FP in time. Finally, it indirectly improves the software measurement based on FPA.

**Key words:** function point; SCM; metadata; software measurement

### 0 引言

随着软件规模和复杂度的增长,软件工程师们一直在致力于寻找一种能应用于大多数软件环境的测量方法。基于功能点分析(FPA, Function Point Analysis)的软件测量已经成为业内较普及的方法。功能点分析法,自 IBM 的 Albrech 在 1979 年发表,随后被 IFPUG (International Function Point User Group)继承,目前即将被 ISO 列为度量软件功能规模的标准<sup>[1]</sup>。

### 1 功能点分析法介绍

IFPUG 版本是在 Albrech 版本的基础上修改而成的。经过修改,IFPUG 加入了对软件复杂度的评估,并且使计数功能点的规则更严密。在 IFPUG 的版本中,计数功能点的过程由以下七步组成。第一步:确认要进行功能点计数的软件类型;第二步:确定系统边界;第三步:计算数据型功能点;第四步:计算事务型功能点;第五步:计算校准前的功能点;第六步:确定校准因子的值;第七步:计算最终校准后的功能点<sup>[2-4]</sup>。

为便于理解后续内容,这里对步骤 3、4 进行简要介绍。

#### 1.1 第三步:计算数据型功能点

数据型功能是指基于用户内部和外部数据需求的那些功能。它被分为以下两类:内部逻辑文件(ILF, Internal Logical File)和外部接口文件(EIF, External Interface File)。其定义如下:

ILF:1)这些“文件”是用户认可的;2)这些“文件”属于系统边界内维护的;3)这些“文件”没有被计入该项目的 EIF 中。

EIF:1)这些“文件”是用户认可的;2)这些“文件”属于系统外部维护的,也即它是别的系统的 ILF;3)这些“文件”没有被计入该项目的 ILF 中。

这里,“文件”指的是逻辑上的一组相关数据,不是指这些数据的物理实现。

在计算出有多少个数据型功能之后,还要分析每个功能的功能复杂度。功能复杂度的划分是基于该功能中数据元素类型(DET, Data Element Types)个数和记录元素类型(RET, Record Element Types)个数的基础上的。DET 是指一个隶属于同一 ILF 或 EIF、用户能够识别的、非重复或割裂的字段。RET 是指一个隶属于同一 ILF 或 EIF、用户能够识别的数据子集,可理解为一个关系数据表。表 1 为 ILF 和 EIF 的 RET/DET 复杂度矩阵。

收稿日期:2005-09-16

作者简介:李聪廷(1982-),男,浙江金华人,硕士研究生,研究方向为软件工程、软件度量;鱼 滨,博士,副教授,硕士生导师,研究方向为软件工程与理论、分布式应用、中间件技术。

表 1 RET/DET 复杂度矩阵

RET \ DET	1-19	20-50	51-
1	Low	Low	Average
2-5	Low	Average	High
6-	Average	High	High

1.2 第四步:计算事务型功能点

事务型功能是指那些供用户处理数据的功能。它可分为三种类型:外部输入(EI, External Input), 外部输出(EO, External Output)和外部查询(EQ, External Inquiry)。其定义如下:

EI:指那些处理来自于系统边界外部的数据和控制信息的功能。也即允许用户通过添加、修改或删除数据来维护 ILF。

EO:指那些将系统产生的数据或控制信息发送到系统边界之外的功能。其输出数据是从 ILF 和 EIF 中派生而来的。

EQ:指那些接收请求,输出相应的功能。其输出数据是直接来自 ILF 和 EIF 中得来的,在其处理过程中并不维护 ILF。

在计算出有多少个事务型功能之后,还要分析每个功能的功能复杂度。功能复杂度的划分是基于该功能中引用文件类型(FTR, File Type Referenced)个数和 DET 个数的基础上的。FTR 是指一个被事务功能所读取或维护的 ILF, 或一个被事务功能所读取的 EIF。表 2 为 EI 的 FTR/DET 复杂度矩阵。

表 2 EI 的 FTR/DET 复杂度矩阵

FTR \ DET	1-4	5-15	16-
0-1	Low	Low	Average
2	Low	Average	High
3-	Average	High	High

2 利用 SCM 工具辅助软件功能点分析

软件配置管理 (Software Configuration Management, SCM) 是软件工程中用来管理软件资产变更的一项规程, 包括相关工具和应用技术(流程和方法)。在 IEEE“配置管理技术标准”[IEEE 828 - 1998] 的引言中称之为 SCM<sup>[5]</sup>。如今几乎所有的软件企业在开发软件项目时, 都采用了配置管理工具对其进行存档和管理。SCM 工具的产品多达数十种, 国内用得较多的是 CVS、ClearCase。

既然软件开发过程中个阶段的相关数据或产品都存放在 SCM 工具中, 那么在对软件进行功能点分析的时候可以从 SCM 工具中获取丰富的参考信息。进而, 如果能把这些有参考价值的信息自动提取出来, 那将大大地提高分析的效率和准确度。

2.1 SCM 工具中的元数据

逻辑上的每个软件“功能”, 在物理实现上来说就是一

部分源代码, 并以文件的形式存在。在软件开发过程中, 软件开发人员会不定期地将自己近期所做的修改进行保存, SCM 工具能很好地对这些源代码的变更进行保存并记录变更的描述信息。其中每提交一次变更都会在 SCM 工具中生成一个新的版本。对每个版本都会有如下描述信息:

\* 创建时间:即开发人员检入 (check in, 即保存至 SCM 工具存储池中) 该版本的时间。

\* 创建者:即记录谁检入了该版本。

\* 状态:即记录该版本是正常存在还是已经删除。

\* 增加行数:即与上一版本相比, 该版本增加了多少行新代码。

\* 删除行数:即与上一版本相比, 该版本删除了多少行旧代码。

\* 代码行:即该版本的 LOC (Line Of Code)

\* 评注:上面所列的元数据都是由 SCM 工具直接或间接记录的, 而该项元数据是由开发人员在提交该版本的时候输入的。通常检入版本时, 开发人员同时应该输入此次变更的原因、内容、存在问题等描述信息。这是软件开发的一个良好习惯, 这使得开发人员可追踪软件的变更。

\* 标签:即开发人员可以通过对某些版本打上标签的方式将版本进行归类。

\* 基线:在项目的关键里程碑中, 全部工件 (artifact, 包括文档和源代码等) 需要一起组成基线。

当然 SCM 工具中还有其他对功能点分析有用的元数据, 这里不一一列出。

2.2 用这些元数据可进行的功能点分析

功能点分析的第二步是确定系统边界。在软件开发的初始(需求、设计)阶段只能从现有的需求、设计文档去确定系统边界。此后, 从编码阶段开始, 便可以将系统的范围与存储在 SCM 工具中的源代码(文件)对应起来。尤其是当项目进行到一定的阶段, 开发人员已经将属于系统范围内的源代码打上了基线, 因此可以通过找出打上了基线的源代码来辅助确定系统边界。

功能点分析的第三、第四步分别是计算数据型功能点和事务型功能点。计算它们时, 首先是要确定每个功能的边界和类型, 然后确定其复杂度。

1) 辅助确定功能的边界和类型:对初次进行功能点分析而言, 在每个文件的第一个版本中的元数据“评注”中, 通常会对该内容或用途有个概念性的描述。通过理解这些描述信息, 可以很快地将每个功能与其所属的源代码文件对应起来, 进而辅助确定这些功能的边界和类型。随着项目的进展, 需要更新每个功能的边界, 通过查看与该功能对应的那些源代码的“状态”(是否已删除)、其后续版本的“评注”和“标签”等信息, 可以很快地得到该功能的边界是否已扩大或缩小。

2) 辅助确定功能复杂度:确定某个功能点的复杂度是 Low, Average 还是 High, 关键是要计算出该功能含有

多少 DET, RET 或 FTR。初始时只能从需求、设计的文档中着手,得到它们的初值。随着项目的进行, DET、RET 和 FTR 都有可能发生变更,而且变更的概率很大。在所对应的源代码的各个版本中,有丰富的描述这些变更的描述信息,通常这类描述信息来自于“评注”和“状态”中,因此可以通过查看“评注”以更新 DET、RET 和 FTR 的值,进而更新该功能的复杂度。

例如,某个项目的其中一个 EI 为“输入用户个人信息(注册)”。在需求阶段对该 EI 进行分析,得到其 DET 为“用户名”、“密码”、“性别”、“出生日期”等共计 10 项, FTR 为“用户信息”共计 1 项。参照 EI 的 FTR/DET 复杂度矩阵(表 1)得到其复杂度为“Low”,再参照复杂度权值矩阵(表 2)得到该 EI 的功能点为 3。但到了代码阶段,通过查看该 EI 对应的源文件在 SCM 中的元数据,发现有的版本有如下描述信息:

“对输入用户个人信息功能,增加‘昵称’、‘头像’、‘等级’、‘个人主页’4 个可选项”

“对输入用户个人信息功能,增加‘选择组’1 个可选项”

从第一、二条信息可知该 EI 的 DET 增加了 5 个,也就是变成了 14。从第二条信息可知该 EI 的 FTR 增加了 1 个,因为“选择组”需要引用“组信息”这个 ILF,也就是 FTR 变成了 2。参照表 1 得到现在的复杂度为“Average”,再参照表 2 得到现在的功能点为 4。

从上述例子可以看到,运用该方法可以及时根据软件功能的变更,方便快捷地得到变更后的功能点。也就是说,不再是静态的分析软件功能点,而是动态的。这对运用功能点的方法度量项目工作量、生产率、成本等都是非常实用的。

### 2.3 用这些元数据进行基于功能点的软件测量

功能点方法从用户或客户的角度出发,独立于软件的具体实现,因此可用于对各种项目进行持久地度量。基于功能点方法的软件度量系统很多,较突出的有 QPMG (Quality/Productivity Management Group) 的 PQMPlus。PQMPlus 以健壮的功能点方法为基础,提供了基于历史数据的项目估算、项目计划和项目评估。该产品已获得 IFPUG 的一类和二类认证,是现今惟一获得 IFPUG 二类认证的度量工具。

QPMG 的软件度量流程(见图 1)<sup>[6]</sup>。

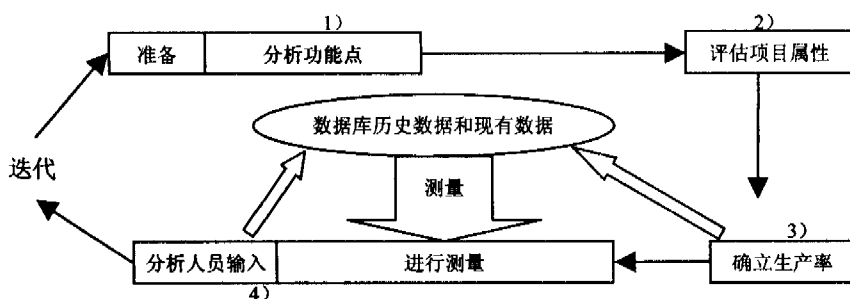


图 1 QPMG 的软件度量流程

#### 1) 辅助确立生产率。

一般来说,生产率可以由多少“代码行/人·月”或“功能点/人·月”来表示,两者各有各的长处。“功能点/人·月”是公认的能比较准确反应生产率的指数,因为功能点方法充分考虑到了复用、复杂度等跟项目本身相关的因素。但是如果及时跟踪项目进度,查看项目组在数天或一周内的工作量或生产率,功能点可能就不太合适了,这时候“代码行/人·月”能比较好地反映短期内的生产率和工作量。因此,测量软件生产率的时候,如果两者都能作为参考指标,就能更好地反映生产率。

由于是基于功能点的软件测量,因此利用 SCM 工具中的“创建时间”和“创建者”等元数据,就可以很快地得到整个项目组或某个项目成员的生产率(“功能点/人·月”)。另外,在上文中还可以看到 SCM 工具中有“代码行”这个元数据,同理,也能得到“代码行/人·月”这样的生产率参考指标。

#### 2) 辅助评估项目属性。

评估项目属性其实就相当于上文所说的第六步:确定校准因子。每个项目都有特殊的属性,一般能影响功能点分析的因子在 IFPUG 里列出的有 14 项。它们是:在线数据输入、在线更新、数据通信、分布式数据处理、复杂的处理逻辑、灵活定制的要求、事务的时效性限制、系统配置的约束、多站点(远程)支持、性能要求、易操作性、易维护性、易安装性、可复用性。SCM 工具中的元数据如“评注”、“标签”、“基线”等,很多都对分析这些因子有帮助(供分析人员参考),这样做最大的好处是既能很好地与实际情况结合,又避免分析人员接触源代码。例如,在某一版本的评注中有“日期校验 Javascript 脚本来自于网上的一个 Open Source 的 Javascript 网站”这样的描述,那么就可以对“复用”因子进行校准。又如,在软件维护过程中,仅凭代码行的增加并不能如实地反映工作量及生产率,找出不合理代码并删除也应计入工作量中,进而影响生产率。因此,“增加行数”和“删除行数”能进一步精确测量软件的规模、工作量和生产率。

总之,SCM 工具中的元数据在软件测量中主要起参考作用,辅助并尽量使测量更精确。

### 2.4 如何用程序提取 SCM 工具中的元数据

以 Java 和 IBM Rational 公司的 ClearCase 为例,从其存储池(Repository)中提取某一文件“/vobs/project1/a.java”的创建日期(CreatedOn)、创建者(Author)、文件类型(Type)、评注(Comment)这四项元数据。

/\* 代码示例 \*/

```
String[] cmd = {"sh", "-c",
```

```
"cleartool describe -fmt '%Sd \n%Fu \n%[type]p \n%c \n' /vobs/project1/a.java"};
```

```
Process ps = Runtime.getRuntime().exec
```

```

(cmd);
BufferedReader in = new BufferedReader(new InputStreamReader
(ps.getInputStream()));
String createdon = in.readLine();
String author = in.readLine();
String type = in.readLine();
String comment = "";
String tmp;
while(tmp = in.readLine() != null)
    comment += tmp;
ps.destroy();
/* 代码示例 */

```

其中“cleartool”是 ClearCase 的命令行工具(区别于一些图形化的工具),基本命令都以其为开始。“describe”用于输出指定对象的描述信息,其可选项“-fmt”可用于指定输出格式。该选项的参数中“%Sd”为输出创建日期,“%Fu”为输出作者,“%[type]p”为输出文件类型,“%c”为输出该文件的描述信息,“\n”为换行符。也就是说执行该命令后的输出可能如下:

2004-05-21

Fred

Text

“a.java is created to test if ‘javac’ could compile .java files”

“Runtime.getRuntime().exec(cmd)”方法是用来输出命令到操作系统级执行,因为平时执行 ClearCase 命令就是在命令提示符中输入并执行的。并且该方法还会返回执行命令的进程的句柄,即 Process 对象。通过 Process 类

的 getInputStream() 方法,可以得到执行该命令后的输出(对我们的程序来说是输入流)。分析输入流(在这里每一行分别代表一个元数据,当然最后的 comment 元数据可能由多行组成),就可以得到想要的上述四个元数据了。

### 3 结 论

在功能点分析时,人为的数据收集较多。利用该方法可以减少人为的行为,让部分数据的收集自动化。更可以使数据来源可靠、及时,这对基于功能点方法对软件进行测量来说是非常重要和有用的。

### 参考文献:

- [1] Kusumoto S. Function point measurement from Java programs [A]. International Conference on Software Engineering [C]. [s.l.]:[s.n.], 2002. 576-582.
- [2] Al-Hajri M A. Modification of standard function point complexity weights system[J]. Journal of Systems and Software, 2005, 74(2): 195-206.
- [3] IFPUG. Function Point Analysis Nears Approval As First International Standard For Software Functional Size [EB/OL]. <http://www.ifpug.org/about/ISOPress.htm>, 2002.
- [4] Longstreet D. Function Point counting practices. Manual release 4.1[Z]. IFPUG. 2001.
- [5] IBM Rational ClearCase. User Manual[Z]. 1999.
- [6] Q/P Management Group, INC. Solutions for improving quality and productivity [EB/OL]. <http://www.qpmg.com/index2.htm>, 2003.

(上接第 165 页)

UDP 承载的 SIP 消息和媒体流穿越大多数的防火墙和 NAT。这种方案的优点就是不需要对现有的防火墙做任何处理,利用 http 协议本身的机制去穿越。并且结合 TURN 的控制方法,使这种方式更加合理。而且用 http 隧道穿越方法比较简便,因为需要服务器去做信令、媒体流的中转,当与公网交互的终端数或出入私网的媒体流数据量较大时,可能成为瓶颈。如果将这种方案应用在 P2P 网络,可以很好地解决这一问题。

### 参考文献:

- [1] 李鸿彬,杨雪华,雷为民. TURN 服务器原型系统的设计与实现[J]. 计算机应用, 2005, 25(7): 1688-1691.
- [2] Rosenberg J. SIP: Session Initiation Protocol [S]. RFC3261, IETF, 2002-06.

- [3] Rosenberg J, Weinberger J, Huiterna C, et al. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs) [S]. RFC 3489, 2003-03.
- [4] Rosenberg J. Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP) [S]. draft-ietf-mmusic-ice-05, 2005-07-17.
- [5] UC Irvine J, Gettys J. Mogul HTTP - Hypertext Transfer Protocol [S]. RFC 2068, 1997-01.
- [6] Rosenberg J. Traversal Using Relay NAT (TURN) [S]. draft-rosenberg-midcom-turn-07 (work in progress), 2005-02-21.
- [7] 黄伟峰. http tunnel 技术在 VOIP 系统中的实现[J]. 微型电脑应用, 2004, 20(2): 44-47.

## 刊 名 变 更 启 示

经国家新闻出版总署[2005]1066号文件批准,本刊自 2006 年开始,更名为《计算机技术与发展》。新编国内统一连续出版物号为:CN61-1450/TP;新编国际标准连续出版物号为:ISSN 1673-629X。邮发代号仍为 52-127。