第 34 卷 第 9 期 计 算 机 工 程 2008 年 5 月
Vol.34 No.9 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)09—0103—03 文献标识码：A 中图分类号：TP311

一种针对中小型软件的简化功能点分析方法

王晓程[1,2]，李 娟[2]，余 方[2,3]
(1. 中国科学技术大学计算机科学技术系，合肥 230027；2. 中国科学院软件研究所互联网软件技术实验室，北京 100080；
3. 中国科学院研究生院，北京 100080)

摘 要：现有的简化功能点分析方法对不同软件个体偏差较大，针对中小型软件，得到的结果普遍偏高。该文针对中小型软件的特点，基于 NESMA Indicative 方法的思想，提出一种简化方法，并在多个项目中进行了应用。实验证明，用于中小型软件时，该方法与同类方法相比，得到的结果更为准确。

关键词：功能点分析；IFPUG 方法；简化功能点

Simplified Function Point Analysis Method Aiming at Small-to-medium-sized Software

WANG Xiao-cheng[1,2], LI Juan[2], YU Fang[2,3]
(1. Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027;
2. Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100080;
3. Graduate University of Chinese Academy of Sciences, Beijing 100080)

【Abstract】 Existing simplified Function Point Analysis(FPA) methods have different effect among different software individuals, especially when applied to the small-to-medium-sized software, which may result in larger result. This paper proposes a simplified function point analysis method based on NESMA indicative method, aiming at the small-to-medium-sized software. Experimental results show that the proposed approach is more accurate than other simplified methods when applied to this kind of software.

【Key words】 Function Point Analysis(FPA); IFPUG method; simplified function point

1 概述

软件规模估算是否及时和准确，对项目的成败具有决定性的影响。功能点分析方法(Function Points Analysis, FPA)是一种基于软件系统的功能度量软件规模的方法，用“功能点数”表示软件的规模。与代码行这种度量单位相比，功能点数与编程语言无关，而且在项目早期确定了软件需求后即可计算。功能点分析方法比较复杂，对使用者要求较高，所需时间较长，还要求较高的需求完整性和准确性。由于这些因素的影响，在项目早期直接使用这种方法通常存在困难，因此需要一种简化、快速的估算方法。简化的功能点分析方法已成为国际上的研究热点之一[1-2]。在现有的简化方法中，比较流行的是 NESMA(Netherlands Software Metrics Users Associ- ation)组织提出的Estimated方法和Indicative方法。但对不同的软件个体，这两种方法的结果偏差较大，尤其是用于中小型软件时，结果普遍偏高。我国的很多软件公司以中小型项目为主，迫切需要针对此类软件的简化方法。

2 相关研究

2.1 IFPUG 方法

很多国际组织致力于功能点分析方法的研究，并形成了多种具体的方法，其中最主流的是IFPUG方法。该方法广泛应用于管理信息系统，已纳入ISO国际标准[3]。目前主要的简化方法均是以该方法为基础进行简化的。本文中的“功能点分析方法”指的就是这种方法。

按照 IFPUG 方法的思想，一个软件系统是由“逻辑数据文件”和“事务处理功能”构成的。前者分为 ILF(Internal Logical Files)和 EIF(External Interface Files)2 类基本功能组件，后者分为 EI(External Inputs), EO(External Outputs), EQ(External Inquiries)3 类。

用户首先根据规则[1]识别出这 5 类基本功能组件，然后确定它们的复杂度。复杂度分为“低”、“中”、“高”3 个等级。每个功能的类型和复杂度决定了它的功能点数。而整个软件的功能点数就是这些基本功能组件的功能点数之和。

2.2 NESMA 简化方法

NESMA 组织提出了 2 种简化的方法：Estimated 和 Indicative，它们也已纳入ISO国际标准[1]，并且完全兼容于IFPUG方法，在实际中经常使用。

(1)Estimated 方法

该方法将全部 ILF、EIF 的复杂度默认为“低”，将 EI、EO、EQ 的复杂度默认为“中”，其他步骤则与 IFPUG 方法完全一样。整个软件的功能点数仍是这 5 类基本功能组件的功能点数之和。

(2)Indicative 方法

该方法直接使用公式：

基金项目：国家“863”计划基金资助项目“支持 CMMI 高成熟度级别的综合量化评估体系及模型”(2006AA01Z182)
作者简介：王晓程(1981－)，男，硕士研究生，主研方向：软件规模估算；李 娟，博士；余 方，硕士研究生
收稿日期：2007-05-30 **E-mail：**wangxiaocheng@itechs.iscas.ac.cn

$$\text{功能点数} \approx (35 \times \text{ILF数量}) + (15 \times \text{EIF数量})$$

这一公式基于如下假设：平均情况下，每个 ILF 对应 3 个 EI、2 个 EO 和 1 个 EQ，每个 EIF 对应 1 个 EO 和 1 个 EQ；而 35 和 15 这 2 个权重，则是将上述 ILF、EIF 的复杂度默认为“低”，EI、EO、EQ 的复杂度默认为“中”，再考虑系统整体的功能性得出的。

这 2 种简化方法都是基于“默认值”来计算的。对大量样本项目，它们与 IFPUG 方法得到的结果的平均值相近；但对个体项目，它们的结果常有较大差异。具体问题如下：

(1)在于中小型软件中，每个基本功能组件通常比较简单，这 2 种方法设置的默认值会使结果偏高，尤其是Indicative方法。国外有学者进行了实验，也证实了这一点[2]。因此，根据笔者的经验，在用于规模不超过 500 功能点的中小型软件时，这 2 种方法的估算结果偏高。

(2)Indicative方法只考虑逻辑数据文件的个数，完全不考虑事务处理功能，准确性较差。NESMA的研究显示[4]，其估算结果与IFPUG方法相比，偏差可达 50%。

(3)Estimated 方法需要识别出每个 EI, EO, EQ，所需时间较长；而且在使用者经验不足或需求不完整时，识别它们是有困难的。

因此，这 2 种简化方法用于中小型软件时，并不能起到很好的效果。

2.3 其他简化方法

除了NESMA的简化方法外，学术界还提出过其他采用默认值的简化方法，都具有各自的局限性。而Early&Quick[5]、KISS等基于经验的简化方法都引入了新的概念，因此，不完全兼容于IFPUG方法，都没有被ISO接受，在实际中很少使用。

3 针对中小型软件的简化功能点分析方法

NESMA 的 2 种简化方法对中小型软件效果不好，原因在于它们完全使用默认值，而忽略了不同软件的具体情况。这些默认值是在大量实验基础上得出的平均值，没有特别针对中小型软件进行调整。

按照IFPUG的定义，事务处理功能，即EI, EO, EQ，必须和逻辑数据文件相关联。因此，Indicative方法所基于的思想，即根据逻辑数据文件判断EI、EO、EQ功能的数量，进而判断软件的功能点数，是有借鉴意义的。NESMA本身也认为，不同场合下，需要调整Indicative方法中的 35 和 15 这 2 个权重[4]。基于此，本文提出一种简化的功能点分析方法，这种方法针对中小型软件提供了默认值，并且提出一些附加的调整规则，支持不同的软件特点。

3.1 逻辑数据文件的计算

首先识别出软件系统中的逻辑数据文件，并确定它们的类型(ILF或EIF)。在IFPUG方法中，逻辑数据文件的复杂度是由其DET(Data Element Types)和RET(Record Element Types)数目决定的。基于经验，在中小型软件中，ILF中的DET数目通常不足 50，而EIF中的DET数目通常不足 20，在这种情况下，RET的数目就是决定复杂度的关键。同时，逻辑数据文件中的“循环数据字段”或“子类型数据文件”个数是其RET数的主要标志[3]。因此，提出以下的复杂度确定规则：

规则 1 对 ILF，如果其包含 4 个或 4 个以上的循环数据字段，或包含 4 个或 4 个以上的子类型数据文件，则认为其复杂度为“中”；否则认为其复杂度为“低”。

规则 2 对 EIF，认为其复杂度为“低”。

最后，根据逻辑数据文件的类型和复杂度，遵照 IFPUG 方法的规则，确定它们的功能点数。

3.2 事务处理功能计算

对逻辑数据文件，经常进行的操作是CRUD/L，即增删改查/列表操作。目前国外的相关研究[1,2,5]中，普遍对这些操作提供默认值。对中小型软件，增加、修改、删除、查看这 4 种操作通常要读、写逻辑数据文件中的多数字段；而查看记录列表的操作通常只须读取少量字段。因此，提出如下规则：

规则 3 对每个 ILF，认为其至少对应 3 个复杂度与其相同的 EI(对应增加、删除、修改操作)，1 个复杂度与其相同的 EQ(对应查看 ILF 中单个记录的操作)，以及 1 个复杂度为“低”的 EQ(对应查看 ILF 中记录列表的操作)。

规则 4 对每个 EIF，认为其对应 2 个复杂度为“低”的 EQ(对应查看 EIF 中单个记录的操作，及查看记录列表的操作)。

由于对某些 ILF，可能会有较多的操作，因此除了应用规则 3 和规则 4，还需要根据实际情况进行调整。于是，基于类似的经验，提出如下规则：

规则 5 对每个 ILF，除了增加、删除、修改这 3 种操作外，如果系统中还存在 3 种或 3 种以上会对其内容产生影响的操作，则认为其额外对应 3 个低复杂度的 EI。

规则 6 对每个ILF，除了查看ILF中单个记录及查看记录列表的操作外，系统中还存在 3 种或 3 种以上查看其中的数据的方式，则认为该ILF额外对应 3 个低复杂度的EQ。需要注意的是，按照IFPUG的定义[3,6]，多数“打开文件”操作及需要自己实现的“打印”、E-mail等功能都属于查看数据的方式。

规则 7 对每个 ILF，判断对其需要有几种统计或报表功能，将这个数目作为它对应的 EO 的数目。如果其中某个功能需要读、写 4 个或 4 个以上的逻辑数据文件，则认为该 EO 的复杂度为“高”，否则认为其复杂度为“中”。

规则 8 在上述分析过程中，如果同一功能针对多个 ILF 起作用，则该功能只计算 1 次，但复杂度设为“高”。

最后，根据事务处理功能的类型和复杂度，遵照 IFPUG 方法的规则，确定它们的功能点数。

3.3 整个软件功能点数的计算

整个软件的功能点数等于 5 种基本功能组件的功能点数之和，即

$$\begin{aligned}\text{功能点数} \approx {} & \sum(\text{ILF的功能点数}) + \sum(\text{EIF的功能点数}) + \\ & \sum(\text{EI的功能点数}) + \sum(\text{EO的功能点数}) + \\ & \sum(\text{EQ的功能点数})\end{aligned}$$

4 应用研究

笔者选择了 12 个软件项目对该方法进行验证。这些软件的规模都不超过 500 功能点，类型包括单机和 B/S 结构 2 种，来源包括作者所在单位的项目、开源软件和文献[3,6]中的示例。

为了避免 IFPUG 方法的结果对使用者产生主观影响，在实验中先使用本文提出的方法，然后再使用 IFPUG 方法、Estimated 方法、Indicative 方法来计算。对参考资料中的示例，IFPUG 方法的功能点数是资料中提供的。实验结果如表 1 所示，单位为功能点，百分比表示该方法和 IFPUG 方法的结果间的偏差。图 1 是实验结果的对比图。

表 1 几种方法估算结果的比较

编号	类型	来源	IFPUG 方法	NESMA Estimated 方法		NESMA Indicative 方法		本文提出的方法	
				结果	偏差/(%)	结果	偏差/(%)	结果	偏差/(%)
1	B/S	实际项目	227	246	+8	295	+30	238	+5
2	B/S	实际项目	288	329	+14	340	+18	282	-2
3	单机	实际项目	211	226	+7	245	+16	222	+5
4	B/S	实际项目	322	376	+17	525	+63	380	+18
5	B/S	实际项目	414	433	+5	440	+6	392	-5
6	B/S	开源软件	117	126	+8	210	+79	132	+13
7	B/S	开源软件	363	402	+11	455	+25	389	+7
8	单机	开源软件	109	117	+7	105	-4	109	0
9	单机	开源软件	372	384	+3	385	+3	363	-2
10	单机	开源软件	390	412	+6	420	+8	369	-5
11	单机	参考资料	65	80	+23	85	+31	64	-2
12	单机	参考资料	118	130	+10	200	+69	132	+12

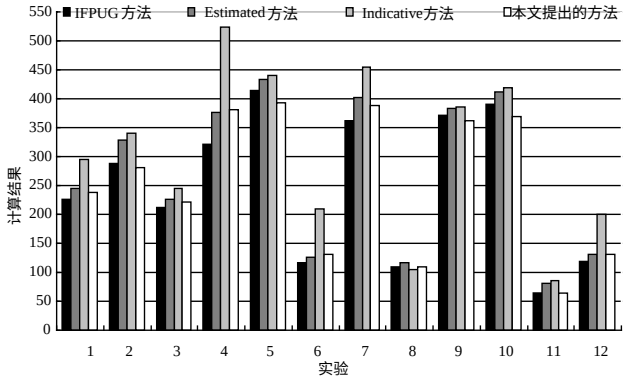

图 1　几种方法估算结果的比较

将几种方法得到的结果进行比较，可以看到：(1)本文的方法与 IFPUG 方法得到的结果十分接近，除项目 4 外，偏差都在 15%以内。(2)与 Indicative 方法相比，在所有项目中，本文的方法明显更接近 IFPUG 方法的结果。(3)与 Estimated 方法相比，在多数项目中，本文的方法更接近 IFPUG 方法的结果。而且不需要识别出所有的事务处理功能，估算所需时间比 Estimated 方法短。

唯一例外的是项目 4。这是一个大型软件中的数据管理子系统，对逻辑数据文件的许多操作位于其他子系统中。对于这种特殊的情况，采用本文的方法偏差稍大，但仍比 Indicative 方法准确。

5　结束语

本文提出的简化的功能点分析方法主要针对中小型软件，提供合适的默认值，并且根据软件的实际情况来进一步调整。对于中小型软件，本方法能得出比 NESMA 简化方法更准确的结果，而且所需时间较短，同时，完全兼容于 IFPUG 方法，没有引入任何新的概念，因此，随着项目的进展及需求的完善，估算者可以不断修正之前的结果，进行持续的软件规模度量。在实验中，本文的方法用于规模不超过 500 功能点的中小型软件时，能够得到很好的结果。对于更大的软件，由于逻辑数据文件的复杂度较高，所对应的功能更多、更复杂，会超出本方法所设定的默认值，因此可能会有较大偏差。这方面还需要进一步的研究。

参考文献

[1] ISO/IEC24570-2005 NESMA Functional Size Measurement Method Version 2.1——Definitions and Counting Guidelines for the Application of Function Point Analysis[S]. 2005.

[2] Candido E J D, Sanches R. Estimating the Size of Web Applications by Using a Simplified Function Point Method[C]//Proceedings of Conf. on WebMedia and LA-Web. [S. l.]: IEEE Press, 2004: 98-105.

[3] ISO/IEC20926-2003 IFPUG 4.1 Unadjusted Functional Size Measure- ment Method-counting Practices Manual[S]. 2003.

[4] NESMA. Early Function Point Counting[Z]. (2003-03-03). http://www.nesma.nl/english.

[5] Santillo L, Conte M, Meli R. Early & Quick Function Point, Sizing More with Less[C]//Proc. of the 11th IEEE International Symposium on Software Metrics. [S. l.]: IEEE Press, 2005: 41.

[6] Garmus D, Herron D. 功能点分析——成功软件项目的测量实践[M]. 钱　岭, 苏　薇, 盛轶阳, 译. 北京: 清华大学出版社, 2003.

（上接第 102 页）

表 4　www.microsoft.com 上的实验数据

Ld-c-Pricision/(%)	α	β	ε	γ	*Q*
90.4	10	0.5	500	2	25
91.3	10	0.7	500	2	24
95.8	10	0.8	500	2	27

表 5　www.ibm.com 上的实验数据

Ld-c-Pricision/(%)	α	β	ε	γ	*Q*
87.2	10	0.6	400	3	72
88.7	10	0.5	500	3	69
84.6	10	0.7	500	3	76

表 6　www.whitehouse.gov 上的实验数据

Ld-c-Pricision/(%)	α	β	ε	γ	*Q*
60.0	10	0.5	500	3	15
78.9	15	0.6	500	3	17
78.9	15	0.7	700	3	17

3.2.3　实验分析

由实验结果可知，在取回网页数达到 5 000 的规模时，算法对逻辑域的挖掘精度仍然较高，并且也有较低的时间复杂度，说明挖掘算法具有较高的效率和适应性。对于 www.whitehouse.gov，其逻辑域精度相对于其他 3 个站点较低，这是由于该站点各个逻辑域所含的网页数量悬殊较大，导致参数的调整对挖掘精度的改善不大。

在算法的改进以及数据积累过程中，需要根据不同的站点情况及其逻辑域规模制定相适应的α, β, ε值。

4　结束语

对大型 Web 站点的结构分析是当前 Web 挖掘领域的热点，本文通过挖掘 Web 站点逻辑域核来生成站点的逻辑结构，取得了较高的精度和效率，为 Web 站点的逻辑结构挖掘提供了一种新思路。

参考文献

[1] Crescenzi V, Merialdo P, Missier P. Discovering the Structure of Large Web Sites Valter Cresenzi[C]//Proceedings of the 27th International Conference on Very Large Data Bases. Washington D. C. , USA: [s. n.], 2001.

[2] Henzinger M R, Motwani R, Silverstein C. Challenges in Web Search Engines[J]. ACM SIGIR Forum, 2002, 36(2): 102-118.

[3] Candan K S, Li Wen-Syan. Reasoning for Web Document Associations and Its Applications in Site Map Construction[J]. Data & Knowledge Engineering, 2002, 43(2): 121-150.

[4] Li Wen-Syan, Kolak O, Vu Q, et al. Defining Logical Domains in a Web Site[C]//Proceedings of the 11th ACM Conference on Hypertext. San Antonio, TX, USA: [s. n.], 2000.