扩展功能点*

吴 际[1], 汤铭端[2]

[1](北京航空航天大学 计算机科学与工程系,北京 100083);

[2](北京计算机应用和仿真技术研究所,北京 100854)

E-mail: wuji@safepro.buaa.edu.cn; bicast@public3.bta.net.cn

摘要: 针对功能点方法的缺点和不足提出了扩展功能点 EFP(extended function points). EFP 符合最新的功能规模度量(FSM)国际标准 ISO/IEC 14143,因此适用于所有的软件应用,并且具有很强的实际可操作性.

关键词: 软件工程;软件度量;软件规模;功能度量;功能点

中图法分类号: TP311 文献标识码: A

规模是软件的一个重要属性,是成本估计和生产率分析的重要参数[1]. 对于软件开发和软件项目管理而言,在开发的早期进行规模估计的要求都很迫切. 但这时有关软件的信息还很少,没有编码可供度量,因此出现了试图从功能角度度量规模的功能点 FP(function points). 但是,功能点固有的缺陷与不足限制了它的使用,本文提出的扩展功能点 EFP(extended FP)从根本上解决了功能点方法的缺陷.

本文将系统地讨论扩展功能点模型. 第 1 节概要介绍功能点,第 2 节指出功能点存在的缺陷,最后给出扩展功能点.

1 功能点

Albrecht 于 1979 年提出了功能点,以求在开发的早期度量软件的规模[2],而后又于 1983 年改进了功能点,使得功能点由 5 个功能分量和一批调整因子组成. 这 5 个功能分量分别是外部输入 EI、外部输出 EO、外部接口文件 EIF、内部逻辑文件 ILF 和查询 EQ. 5 个功能分量的加权累加就是未调整的功能点,再应用 14 个调整因子可得到调整的功能点[3]. 本文所讨论的功能点就是 1983 年 Albrecht 改进了的功能点.

1986 年,SPR 改进功能点后得到特征点(feature points),使其适用于非信息处理领域[4,5]. 1994 年,波音公司提出了 3 维功能点——3DFP,即所谓的三维是指数据、功能处理和控制[6].

2 功能点存在的问题

虽然功能点已取得了广泛的应用,但细心研究会发现它存在如下一些问题:

(1) 功能点的应用领域具有一定的局限性.

功能点主要是针对信息处理系统而设计的,因此其应用领域有很大的局限性.

(2) 功能点度量元素不完整,或者说度量元素的概念间存在重叠.

功能点定义系统的用户输入为维护 ILF 的 EI 和不维护 ILF 的输入(简记为 UI)两类. 功能点

* 收稿日期: 1998-11-12; 修改日期: 1999-11-15

作者简介: 吴际(1973-),男,安徽肥西人,博士生,主要研究领域为软件工程,软件度量,软件测试,编译理论,软件开发环境与技术;汤铭端(1964-),男,江西南昌人,博士,研究员,博士生导师,主要研究领域为软件工程,数字仿真.

明确指出要计算 EI,但并没有明确指出要计算 UI,只是在计算 EQ 时提到[3].这说明可能根本不计算 UI,使得功能点度量元素在概念上不完整;也可能是在计算 EQ 的同时计算了 UI,这说明 UI 是 EQ 的一个组成部分,因此,功能点元素概念间有重叠.

(3) 功能点中 ILF 和 EIF 的定义不清晰,不便于实际操作.

什么是文件?什么是数据的逻辑组?这使得 ILF 和 EIF 的定义模糊.

(4) 实际上往往无法得知某一个外部输入是否出自于另一个系统的内部逻辑文件.

功能点定义 EIF 必须同时又是另一个应用系统的 ILF,而实际上这往往无法验证.

(5) 查询被定义为一个处理,却被描述为一个事务.

功能点把外部查询定义为一个映射,(1,0)对,也就是一个功能处理,可实际查询时却被描述为事务.

3 扩展功能点 EFP

扩展功能点 EFP 从功能角度度量软件的规模,独立于开发所使用的语言.一旦获得了用户功能需求 FUR[7](functional user requirements),就可以用它来度量规模.下面先讨论功能规模度量的特征,然后分析 EFP 的问题域模型,并从问题域模型中获得 EFP 的基本功能块 BFC(base functional component*)类型,同时给出各个 BFC 类型的定义和度量方法,最后介绍 EFP 计算模型.

3.1 功能规模度量

ISO/IEC 14143 是有关功能规模度量 FSM(functional size measurement)的标准[8],它用于评价某种规模度量方法是否为 FSM 方法,该标准要求一个 FSM 方法满足:

(1) 基于用户功能需求 FUR 度量软件的规模;

(2) 一旦得到了 FUR,就可以应用它来度量规模;

(3) 通过对 BFC 的评价可以得到软件的功能规模.

BFC 是 FSM 方法基于用户功能需求 FUR 定义的基本单元,它具有以下特征:

- 只描述 FUR;
- 不描述任何有关技术需求的信息;
- 不描述任何有关质量需求的信息;
- 任何一个 BFC 都可以被识别为一个 BFC 类型,并且只能识别为一个类型.

3.2 EFP 问题域模型

FSM 方法从功能角度来度量软件的规模,因此,EFP 的问题域就是软件的用户功能.软件对于用户就是一个黑盒,而用户关心的是软件需要用户提供哪些输入,同时能为用户返回哪些输出.软件就是一个把用户输入转换成用户输出的机构,其中用户输入和用户输出都是由用户定义的,因此转换功能也是由用户定义的.有些用户输入对于用户具有特殊的意义,用户希望软件能够将这些数据保存起来,以便更好地实现用户提出的功能.这些数据,我们可称为用户数据实体,虽然保存在软件的内部,但却是由用户提供的,因而由用户维护和使用.由于用户数据实体直接与用户输入、用户输出和用户功能相关,因此它对于软件规模的影响将不能被忽略.这样,我们最终得到 EFP 问题域的 4 个元素——用户输入、用户输出、用户功能和用户数据实体.

---

* 此处的"Component"与软件开发中的构件(component)技术无关.

为了有效地识别用户和软件之间的关系,我们为软件定义一个概念边界.由此,用户输入就是穿过边界去往软件的输入,用户输出就是穿过边界去往用户的软件输出.同理,根据计算机科学理论,用户功能就是用户输入到用户输出的映射.为便于后面的讨论,我们将用户输入和用户输出更名为外部输入和外部输出,所谓"外部"是相对于这个概念边界而言的.这样,我们就得到了如图1和式(1)所示的EFP问题域模型.

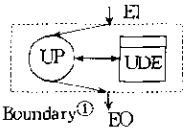

Fig. 1 EFP domain model in graph
图1 EFP问题域图示模型

$$\begin{aligned}&\text{Domain}=(\text{EI},\text{EO},\text{UP},\text{UDE},\text{Boundary})\\&\text{EI}:\text{User}\mapsto\text{Boundary}\\&\text{EO}:\text{Boundary}\mapsto\text{User}\\&\text{UP}:\text{EI}\mapsto\text{EO}\\&\text{UDE}:\exists\ p((\text{UDE}\to p)\vee(p\to\text{UDE}))p\in\text{UP}\end{aligned}\tag{1}$$

其中Domain表示EFP的问题域,User代表软件的用户,EI(external input)是外部输入,EO(external output)是外部输出,Boundary是软件的概念边界.符号"→"表示映射运算符,UP(user processing)是用户处理,用以处理外部输入.UDE(user data entity)是用户数据实体,谓词$A\to B$表示有信息从A流到B,A和B可以是任何形式的实体存在.

3.3 EFP的BFC类型

由上节的EFP问题域模型,我们可以直接得到EFP的4个BFC类型:外部输入EI、外部输出EO、用户功能UP和用户数据实体UDE.从EFP问题域模型可以看出,EFP的4个BFC类型不仅完整地描述了功能规模,而且彼此之间绝无任何概念上的重叠.下面给出各个BFC类型的定义和所包含的属性.为了保证严谨性,首先定义几个术语.

(1) 实体.由多个元素有机地结合在一起,执行特定的功能以达到特定目标的集合体[9].

(2) 数据项.程序可以处理的最小数据命名单位[9].

(3) 数据实体.数据项的集合体,可作为一个整体完整地描述一个实体.

(4) 用户.任何描述用户功能需求的人和任何时候与系统通信或交互的人或事物.其中的事物可以是软件应用、动物、传感器或其他硬件[8].

3.3.1 外部输入EI

EI是一个数据实体,如果一个用户输入有不同于其他EI实例的格式,或者要求系统对它有不同的处理逻辑,则该输入应该被视为一个EI实例[3].EI的度量目标是软件对于外部输入的接收所对应的规模,它有3个属性:

- 数据元素类型DET(data element type)[10]数目;
- 用户输入获得方式IEA(approach of user-input entering into software);
- 用户输入传输方式ITA(approach of user-input transmitted toward software).

DET是在结构上不能再分解的元素类型.IEA通常包括通过非人机界面NHI(non-human interface)获得、通过控制的人机界面CHI(controlling HI)获得、通过文本的人机界面THI(text HI)获得和通过图形的人机界面GHI(graphic HI)获得这4种方式.ITA通常分为远程传输方式Distant和本地传输方式Local.Distant方式是指软件在远程获得用户输入后,通过网络将其传输到本地来进行处理.在实际中会出现捆绑输入(如通过NHI和CHI获得输入的软件),由EI的度量目标可知,应该将所有捆绑在一起的用户输入作为一个EI实例来度量.所谓捆绑输入,是指将若干用

户输入杂凑成一个输入变量来完成这些用户输入的接收.

3.3.2 外部输出 EO

EO 是一个数据实体,如果一个用户输出有不同于其他 EO 实例的数据格式,或是由系统不同的处理逻辑产生,那么这个输出应该视为一个 EO 实例[3]. EO 的度量目标是软件关于将用户输出发送到用户所对应的规模. 外部输出有 3 个属性:

- 数据元素类型 DET[10]数目;
- 用户输出发送方式 OLA(approach of user-output leaving away software);
- 用户输出传输方式 OTA(approach of user-output transmitted toward user).

第 2 个属性 OLA 同 EI 的 IEA 属性,第 3 个属性 OTA 同 EI 的 ITA 属性. 同样,有时会出现将多个不同的用户输出捆绑发送给用户的情况,由 EO 的度量目标可知,应该将所有捆绑在一起发送的用户输出作为一个 EO 实例. 所谓捆绑输出,是指将若干用户输出装配成一个输出变量来完成这些用户输出的发送.

3.3.3 用户功能 UP

用户功能处理 UP 接受用户输入,对其加以处理产生用户预期的输出. 它的度量目标是由用户输入产生用户输出的过程所对应的规模. UP 有 4 个属性:

- 所有相关的用户输入 DET[10]数目;
- 所有相关的用户输出 DET[10]数目;
- 所属功能范畴 FD(function domain);
- UDE 使用计数.

FD 包括科学计算范畴 SC(science computing)、数据管理范畴 DM(data management)、系统外部状态监控范畴 SM(state monitoring)和系统功能范畴 SF(system function). SM 通常是指软件中的那些控制功能,如监视系统外部的状态或者某个信号,并且在必要时能够发出指令进行控制等. 系统功能范畴是指系统软件中的功能.

3.3.4 用户数据实体 UDE

UDE 是一个数据实体,它的度量目标是使用它所对应的规模. UDE 有如下属性:

- 结构元素类型 SET(structural element type)[10]数目;
- 数据元素类型 DET[10]数目;
- 数据管理方式 IMA(information management approach).

结构元素可分解为若干子结构元素或数据元素. IMA 有数据库管理 DBM(data base management)、数据文件管理 DFM(data file management)和内存变量管理 MVM(memory variables management)3 种管理方式.

3.4 BFC 实例度量

BFC 实例度量的过程是依据 BFC 类型,定义从系统的用户功能需求中抽取出所有相关的 BFC 实例,然后将它们类型化. 对于 EI,EO 和 UDE,可直接获得度量结果. 为度量 UP 实例,我们把系统 S 定义为一个有向图[11]:

$$S=\{V,E,D,F,L\}, \tag{2}$$

其中 V 是顶点集;E 是边集,$E:V\to V$;D 是数据集,通过 F 依附于 E,$F:E\to D$;L 表示依附于边的数据之间的限定关系,共有 3 种:$*$,$+$,$\oplus$[12,13]. 设 $d_1,d_2\in D$,

若 $L(d_1,d_2)=*$,则表明 d_1 和 d_2 同时产生、同时消亡;

若 $L(d_1,d_2)=+$,则表明 d_1 和 d_2 之间没有确定的产生和消亡的约束;

若 $L(d_1,d_2)=\oplus$,则表明 d_1 和 d_2 一定不能同时产生、同时消亡,二者具有互斥性.

同时,我们定义两种特殊的顶点 V_E 和 V_I,它们分别代表外部节点和内部节点.所谓外部节点就是用户,内部节点就是 UDE.如果数据流图有环,首先应消除它.令接受回流数据流的节点为 v,那么它至少有两个引入数据流,d_1 和 d_2,其中 d_2 是回流数据流.如果 d_1 和 d_2 的类型一致,则忽略 d_2;如果 d_1 和 d_2 的类型不一致,则将 v 分解为两个节点 v_1 和 v_2,其中 v_1 只接受 d_1 并输出 d_2 到 v_2,回流的 d_2 也由 v_2 来接受,这时又是回流数据流不变的情况.

假设存在一个节点 V,它有若干输出边,依据所有这些边之间的约束关系可以将其分组,组内的边之间的约束关系为"$*$",而组间的约束关系为"$+$"或"$\oplus$".那么我们可以将节点分裂为两个(满足该条件的组的个数)节点 V_1 和 V_2.同时增加一个节点 V_0,接收节点 V 的所有输入边,它有两条输出边(由节点 V 分裂出来的节点个数),分别连接到节点 V_1 和 V_2,这两条边之间的约束关系就是节点 V 的两个组之间的约束关系.对所有这样的节点都作相同的处理.在进入算法描述之前我们先定义几个术语:

节点输入集 Input:$V\to D$.当搜索到节点 V 时,搜索过程中遇到的用户输入的集合;

节点输出集 Output:$V\to D$.当搜索到可达节点 V 时,搜索过程中遇到的用户输出的集合;

边的头尾节点 $H:E\to V$ 和 $T:E\to V$.对于边 $e=(u,v)$,$H(e)=u$,$T(e)=v$.

将所有节点 V_E 的输出分成 n 类(可能存在某些输出无法分类,它们的处理后面会提到).设存在类 C,对于任意 $e\in C$,有且仅有一个 $v\in V$,使得 $e=(V_E,v)$;同时,对于任意 $e\in E$,若 $T(e)=v$,则 $H(e)=V_E$,且 $e\in C$.初始化 Input$(v)=\varnothing$,对于任意 $e\in C$,执行 Input$(v)=$Input$(v)\cup\{F(e)\}$.对任意 $v\in V$,初始化 Output$(v)=\varnothing$.从其中的任一类对应的节点开始进行遍历,设当前节点为 N,通过有向边 e 可到达非 V_I 类节点 Q:

① Input(Q):=Input(N),Output(Q):=Output(N)(:=代表赋值操作,下同);

② 若存在一条边 $e'=(V_E,Q)(e'\neq e)$,执行 Input(Q):=Input$(Q)\cup\{F(e')\}$;

③ 若 Q 还存在另外的输入边 $e_1,e_2,\ldots,e_m$,若其中 $e_i,\ldots,e_j(1\leqslant i\leqslant j\leqslant m)$,对于任意 $i\leqslant t\leqslant j$,都有 $H(e_t)\neq V_I$,$L(F(e),F(e_t))=*$(事实上,如果约束关系为"$+$"或"$\oplus$",则每个 e_t 都要被单独处理),则把所有这些边的头节点的输入集合并到节点 Q 的输入集里,即执行 Input(Q):=Input$(Q)\cup$Input$(H(e_t))(i\leqslant t\leqslant j)$;

④ 若存在一条边 $e'=(Q,V_E)$,则这条边是穿越系统边界去往用户的,执行 Output(Q):=Output$(Q)\cup\{F(e')\}$.如果有多个这样的输出,处理方法相同.

⑤ 若 Q 还有另外的输出边 $e_1,e_2,\ldots,e_m$,任意 $1\leqslant t\leqslant m$,$T(e_t)\neq V_I$,不论这些边上的数据有何约束关系,分为 m 个分支往下搜索,转算法第 1 步.否则,此时我们已找到一条处理链,将其抽象为一个处理 P,其输入 $P.I=$Input(Q),输出 $P.O=$Output(Q).

⑥ 重新挑选另一个分类来搜索,转算法第①步,直到处理完所有的分类为止.

⑦ 对于某个内部节点 V_I,设存在一条边 e 与之相连,并且包含 e 的另一端节点的用户功能处理分别为 $P_1,P_2,\ldots,P_m$,那么对任意 $P_i(1\leqslant i\leqslant m)$,其 UDE 使用计数加 1,如果 V_I 有不止一条这样的边相连,作相同的处理,直到处理完所有的 V_I 为止.

上述算法是一个遍历算法,它能够有效地度量出用户功能处理 UP 实例和 UP 的 UDE 使用计数.

3.5 EFP 度量模型

在得到 BFC 实例之后,首先需要确定各个 BFC 实例的复杂度来索引不同的权值.BFC 实例的复杂度判定可以根据其属性来完成,EI 和 EO 通过它们所包含的 DET 数目(见表 1),UDE 通过它所包含的 SET 和 DET 数目(见表 2),UP 则通过 DET 数目和 UDE 使用计数共同决定(见表 3).UP 的 DET 数目是指其对应的输入和输出所包含的 DET 数目最多的那个.

Table 1 EI/EO complexity matrix

表 1 外部输入/外部输出复杂度矩阵

1~4 DETs	5~15 DETs	16+ DETs
Simple①	Average②	Complex③

①简单,②平均,③复杂.

Table 2 UDE complexity matrix

表 2 用户数据实体复杂度矩阵[10]

SETs	1~19 DETs	20~50 DETs	51+ DETs
1	Simple①	Simple	Average②
2~5	Simple	Average	Complex③
≥6	Average	Complex	Complex

①简单,②平均,③复杂.

Table 3 UP complexity matrix

表 3 用户功能复杂度矩阵

UDE	1~4 DETs	5~15 DETs	16+ DETs
0~1	Simple①	Simple	Average②
2	Simple	Average	Complex③
≥3	Average	Complex	Complex

①简单,②平均,③复杂.

如何将 BFC 实例转换为扩展功能点数目呢?我们采用式(3)给出的模型,针对不同的 BFC 类型和实例复杂度,用加权的方法实现转换,累加结果就是扩展功能点数目.

$$\begin{aligned} BFC_i &= W_L * N_L + W_A * N_A + W_H * N_H, \\ EFP &= \Sigma BFC_i \quad (1 \leqslant i \leqslant 4). \end{aligned} \tag{3}$$

其中(W_L,W_A,W_H)为 3 种复杂度情况下的权值,分别为(低复杂度权值、平均复杂度权值、高复杂度权值);(N_L,N_A,N_H)为对应 3 种复杂度的实例数目.BFC 类型的权值由它的属性确定,EI 权值由 IEA 和 ITA 确定,EO 权值由 OLA 和 OTA 确定,UP 权值由 DF 确定,而 UDE 的权值则通过 IMA 来确定.表 4 是权值表,IEA/OLA("/"代表"或",下同),ITA/OTA,DF 和 IMA 的取值方法见表 5.

Table 4 References of BFC weight

表 4 BFC 类型权值表

	Simple①	Average②	Complex③
W_{EI}	UEA+ITA−0.5	IEA+ITA	IEA+ITA+1
W_{EO}	OLA+OTA−0.5	OLA+OTA	OLA+OTA+1
W_{UP}	FD−1	FD	FD+2
W_{UDE}	IMA_L	IMA_A	IMA_H

①低复杂度权值,②平均复杂度权值,③高复杂度权值.

Table 5 References of IEA/OLA, ITA/OTA, DF and IMA

表 5 IEA/OLA,ITA/OTA,DF 和 IMA 的取值

	NHI	CHI	THI	GHI
IEA/OLA	1/1	1/1	2/3	4/5
	Distant		Local	
ITA/OTA	1/1		0/0	
	SC	DM	SM	SF
FD	3	4	5	5
	DBM		DFM	MVM
(IMA_L,IMA_A,IMA_H)	(5,7,10)		(3,4,6)	(1,1,2)

扩展功能点模型输出的是扩展功能点个数,可以将其转化为LOC,见表6.

Table 6 Ratio of LOC to EPP[13]

表6 扩展功能点对LOC的转化[13]

Language①	LOC/EFP	Language	LOC/EFP
Ada	71	Fortran 77	105
AI Shell	49	Forth	64
APL	32	Jovial	105
Assembly (Macro)	320	Lisp	64
ANSI/Quick/Turbo Basic	213	Modula 2	80
Basic-Complied	64	Pascal	91
Basic-Interpreted	91	Prolog	64
C	128	Report Generator	80
C++	29	Spreadsheet	6
ANSI Cobol 85	91		

①语言.

4 结束语

EFP模型是本文第一作者硕士学位论文的一部分,由于时间的限制和其他种种原因,该模型仅在作者所在单位里进行过一次实际的度量,因而可用的数据积累很少,不足以用来调整模型参数,所以EFP模型参数的初值基本上是参考功能点和一些经验数据得到的.但是,通过严谨的分析和比较,我们坚信EFP模型无论在模型本身,还是在其可操作性方面都要好于功能点模型.本文致力于把EFP模型推向工业界,通过广泛的应用和数据的积累,就一定会得到一个有效的功能规模度量模型(几乎所有的度量模型都要经过这样的试用阶段以获取重要的数据积累,从而得到广泛的应用).

References:

[1] Fenton, N. E., Pfleeger, S. L. Software Metrics, A Rigorous and Practical Approach. 2nd ed., Boston: PWS Publishing Company, 1997.

[2] Albrecht, A. J. Measuring application development productivity. In: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium. Monterey, CA: 1979. 83~92.

[3] Albrecht A. J., Gaffney, J. Software function, source lines of code and development effort prediction. IEEE Transactions on Software Engineering, 1983,SE-9(6):639~648.

[4] Capers, J. Applied Software Measurement, Assuring Productivity and Quality. 2nd ed., New York: McGraw-Hill, Inc., 1996.

[5] Capers, J. Programmer Productivity. New York: McGraw-Hill, Inc., 1986.

[6] Whitemire, S. A. An introduction to 3D function points. Software Development, 1995,3(4):43~53.

[7] Mazza, C., Fairclough, J. Software Engineering Guides. London: Prentice Hall, Inc., 1996.

[8] ISO/IEC 14143. Information Technology-Software Measurement-Functional Size Measurement. ISO/IEC/JTC1/SC7 Standard, 1998.

[9] Zhang, Xiao-xiang, Niu Tian jia, Jiang Xue-guo, *et al.* New English Chinese Computer Dictionary. Beijing: People Post and Telecommunications Press, 1988 (in Chinese).

[10] Abrain, A., Robillard, P. N. Function points analysis: an empirical study of its measurement processes. IEEE Transactions on Software Engineering, 1996,22(12):895~910.

[11] Wang, Shu-he. Graph Theory and Algorithm. Hefei: University of Science and Technology of China Press, 1990 (in Chinese).

[12] Zhang, Hai-fan. Introduction to Software Engineering. 3rd ed., Beijing: Tsinghua University Press, 1998 (in Chinese).

[13] Pressman, R. S. Software Engineering, a Practitioner's Approach. 4th ed., New York: McGraw-Hill, Inc., 1997.

附中文参考文献:

[9] 张效祥,牛田佳,江学国,等.新英汉计算机大辞典.北京:人民邮电出版社,1998.

[11] 王树禾.图论及其算法.合肥:中国科学技术大学出版社,1990.

[12] 张海藩.软件工程导论.第3版,北京:清华大学出版社,1998.

Extended Function Points*

WU Ji[1], TANG Ming-duan[2]

[1](*Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, Beijing* 100083, *China*);

[2](*Beijing Institute of Computer Application and Simulation Technology, Beijing* 100854, *China*)

E-mail: wuji@safepro.buaa.edu.cn; bicast@public3.bta.net.cn

Abstract: To overcome the imperfections of the Function Points (FP), the Extended Function Points (EFP) is presented, which complies with the FSM (functional size method), an international standard of ISO/IEC 14143. So EFP can be used to all kinds of software applications and has excellent practical operability.

Key words: software engineering; software metrics and measurement; software size; functional measurement; function point

* Received November 12, 1998; accepted November 15, 1999