**功能点(Funciton Point FP)估算法(一) 识别项目范围和数据复杂度**

2014年03月31日 14:57:43 [极杰子](https://me.csdn.net/wj3319) 阅读数 10827更多

分类专栏： [项目管理](https://blog.csdn.net/wj3319/article/category/2155957)

功能点估算法是软件项目管理众多知识中比较有技术含量的一个。在软件项目管理中项目计划制定的优劣直接关系到项目的成败，项目计划中对项目范围的估算又尤为重要。如果项目负责人对项目的规模没有一个比较客观的认识，没有对工作量、所需资源、完工时间等因素进行估算，那么项目计划也就没有存在的意义。

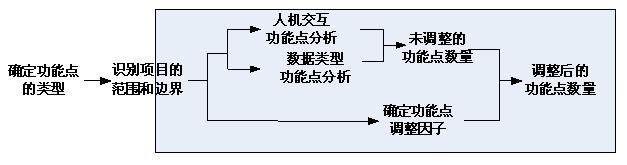
功能点估算法的特点

    项目范围的估算在CMMI的“MA”度量分析管理和“PP”项目计划中均有涉及。对软件项目范围的估算有很多种方法，常见的是LOC代码行和FP功能点法。它们之间的区别和关系如下：

* 功能点估算法常用在项目开始或项目需求基本明确时使用，这时进行估算其结果的准确性比较高。假如这个时候使用LOC代码行估算法，则误差会比较大。
* 使用功能点估算法无需懂得软件使用何种开发技术。LOC代码行估算法则与软件开发技术密切相关。
* 功能点估算法是以用户为角度进行估算，LOC代码行估算法则是以技术为角度进行估算。
* 通过一些行业标准或企业自身度量的分析，功能点估算法是可以转换为LOC代码行的。

    在项目刚开始的时候进行功能点估算可以对项目的范围进行预测。在项目开发的过程中由于需求的变更和细化可能会导致项目范围的蔓延，计算出来的结果会与当初估计的不同。因此，在项目结束时还需要对项目的范围情况重新进行估算，这个时候估算的结果才能最准确反映项目的规模。  
　　功能点分析的步骤

    本文将以国际标准IFPUG（International Function Point Users Group）组织提供的功能点估算法V4.1.1为基础进行讲解。如下图所示，首先大家应该了解功能点估算法的使用步骤。

  
图1 功能点估算法的步骤

    具体步骤包括：  
    1. 识别功能点的类型。  
    2. 识别待估算应用程序的边界和范围。  
    3. 计算数据类型功能点所提供的未调整的功能点数量。  
    4. 计算人机交互功能所提供的未调整的功能点数量。  
    5. 确定调整因子。  
    6. 计算调整后的功能点数量。

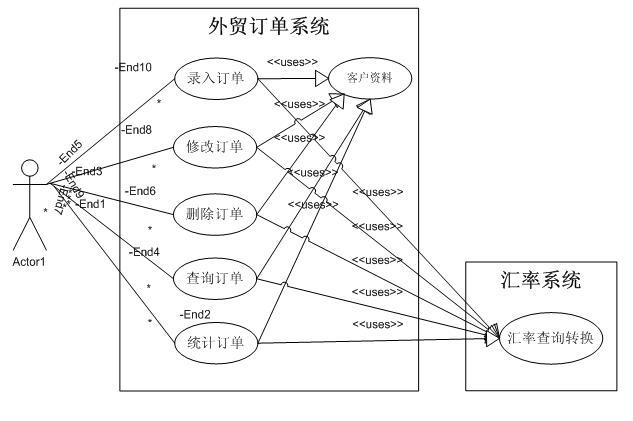
识别项目的类型

    国际IFPUG组织将软件项目分为三类，功能点估算法适用于任何一类项目：

* 新开发项目
* 二次开发的项目
* 功能增强的项目

识别项目的范围和边界

    使用UML的“UseCase”用例图是以用户角度进行识别项目范围和边界的最好方法，在画用例图时就必须明确系统的边界。通过系统的边界，我们可以知道哪些功能要计算功能点，哪些功能点是外部系统负责计算的。以图2为例：一个外贸订单系统只包含录入、修改、删除、查询和统计订单的功能，而汇率查询转换服务是不属于该系统的。  
    应用程序边界的识别规则大家一定要牢记，不能从技术角度去思考，必须从用户角度来定义；如果项目牵扯到多个系统，那么必须将这多个系统的边界全部描述清楚。

  
图2 外贸订单系统用例图

功能点估算分类

    功能点估算法将功能点分为以下5类：  
    1. ILF：Internal Logical File内部逻辑文件  
    2. EIF: External Interface File外部接口文件  
    3. EI: External Input外部输入  
    4. EO: External Output外部输出  
    5. EQ: External Inquiry外部查询

    其中，ILF和EIF属于数据类型的功能点，EI、EO、EQ属于人机交互事务类型的功能点。  
    以外贸订单系统项目为例：

* 录入订单、修改订单、删除订单是EI；
* 查询订单是EO
* 统计订单是EQ
* 汇率查询转换系统为EIF
* 订单和客户是ILF

识别功能点的重要原则

    ILF、EIF要与EI、EO、EQ分开计算。对ILF和EIF复杂度的计算可以简单理解为对数据库复杂度的计算。对EI、EO、EQ复杂度的计算可以理解为对程序开发复杂度的计算。一般软件项目都是由数据和程序构成的，因此计算ILF、EIF和计算EI、EO、EQ之间没有任何关系。

内部逻辑文件与外部接口文件

    ILF内部逻辑文件  
    内部逻辑文件是指一组以用户角度识别的、在应用程序边界内且被维护的逻辑相关数据或控制信息。ILF的主要目的是通过应用程序的一个或多个基本处理过程来维护数据。

    EIF外部接口文件  
    外部接口文件是指一组在应用程序边界内被查询，但在其他应用程序中被维护的、以用户角度来识别的、逻辑上相关的数据。因此，一个应用程序中的EIF必然是其他应用程序中的ILF。EIF的主要目的是为边界内的应用程序提供一个或多个通过基础操作过程来引用的一组数据或信息。  
    EIF所遵循的规则：

* 从用户角度出发识别的一组逻辑数据。
* 这组数据是在应用程序外部，并被应用程序引用的。
* 计算功能点的这个应用程序并不维护该EIF。
* 这组数据是作为另一个应用程序中的ILF被维护的。

ILF和EIF的复杂性计算

    ILF和EIF的复杂性是取决于RET（Record element type）和DET（Data element type）的数量。DET是一个以用户角度识别的、非重复的、有业务逻辑意义的字段。  
    DET计算的规则如下：

* 通过一个基本处理过程的执行，对ILF进行维护，或从ILF/EIF中返回一个特定的、用户可识别的、非重复的字段，那么每个这样的字段算一个DET。

    例如：添加一个外贸订单时需要保存“订单号码、订单日期、地址、邮编”，那么对于ILF订单来说它的DET就是4个。  
    再如：保存订单时还会保存订单的明细。订单的明细往往作为一个子表进行保存，那么“订单号码”在主表和子表中都同时存在（主外键）。但以用户角度来识别时，存盘操作是一个最小的单位，那么订单号码只能算做一个DET。

* 当两个应用程序维护和/或引用相同的ILF/EIF，但是每个应用程序分别维护/引用它们相应的DET时，这些DET在这两个应用程序的维护/引用中将单独计算。

    例如，一个应用程序的两个“Elementary Process”基本处理过程都需要使用到“地址”的信息，地址信息又可以细分为“国家、城市、街道、邮编”。那么对于其中一个基本处理过程来说，它将整个地址信息作为一个整体进行处理，只算一个DET；另外一个基本处理过程使用每个地址的详细信息，那么DET就是4个。

    RET计算的规则如下：  
    RET是指一个EIF/ILF中用户可以识别的DET的集合。如果把DET简单理解为字段的话，那RET就可以简单理解为数据库中的表。RET在ILF/EIF中分为两种类型：可选的（Optional）和必选的（Mandatory）。计算RET的规则为以下两点：

* 在一个ILF/EIF中每一个可选或必选的集合都被计算为一个RET。
* 如果一个ILF/EIF没有子集合，则ILF/EIF被计算为一个RET。

    例如：在外贸订单系统中添加一个订单时会保存“订单信息、客户的ID、部门的ID”。那么订单系统ILF中的RET为：  
    1. 订单信息（必选的）  
    2. 客户信息（必选的）  
    3. 部门信息（可选的）  
    因此ILF中RET的个数为3个。

    ILF/EIF复杂度的矩阵如下：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1~19个DET | 20~50个DET | 超过51个DET |
| 1个RET | 低 | 低 | 中等 |
| 2~5个RET | 低 | 中等 | 高 |
| 6个以上RET |  |  |  |

第一部分EI、EO、EQ

EI是处理来自于应用程序边界外部的一组数据的输入，它的主要目的是维护一个或多个ILF，以及/或者更改系统的行为。 EO是输送数据到应用程序边界外部的过程。它的主要目的是通过逻辑处理过程向用户呈现信息。该处理过程必须包含至少一个数学公式或计算方法，或生成派生数据。一个EO也可以维护一个或多个ILF，并/或改变系统行为。 EQ是向应用程序边界外发送数据基本处理的过程。其主要目的是从ILF或EIF中通过恢复数据信息来向用户呈现。该处理逻辑不包括任何数学公式或计算方法，也不会生成任何派生数据。EQ不会维护任何一个ILF，也不会改变应用程序的系统行为。

l   EO和EQ的共同点其主要目的都是通过基本操作过程展现数据给用户看。

Ø  主要目的

目的 EI EO EQ

改变应用程序的属性或行为 主要目的 次要目的 不允许

维护一个或多个ILF 主要目的 次要目的 不允许

显示信息给用户 次要目的 主要目的 主要目的

 Ø  主要行为

行为 EI EO EQ

数学公式或计算被执行 可以 至少选择一次 不可以

至少一个ILF被修改 至少选择一次 至少选择一次 不可以

至少一个ILF或EIF被引用 可选 可选 必选

数据被重新恢复 可选 可选 必选

派生数据被创建 可选 至少选择一次 可选

应用程序的行为或属性被修改 至少选择一次 至少选择一次 可选

准备或呈现信息到系统边界外 可选 必选 必选

接受进入系统边界内的数据的能力 必须 可选 可选

 计算规则

在IFPUG的定义中有一个重要的单词“Elementary Process”基本处理过程。该过程对用户来说是一个有意义的最小的活动单位，并且是一个自包含的活动。功能点的分类EI、EO、EQ的识别都是基于“Elementary Process”基本处理过程的。

●   EI的计算规则：

1.    从应用边界之外收到数据。

2.    如果进入系统边界内的数据不是一个改变系统行为的控制信息，那么至少一个ILF应该被改变。

3.    对于已识别的处理过程，至少满足下面三个条件之一。

ü   该基本处理过程的逻辑与本应用系统中其它基本处理过程的逻辑不同。该基本处理过程应该具有唯一性。例如：不能存在两个完全一模一样的存盘操作。

ü   在应用程序边界内，该基本处理过程所使用的这组数据应该与其他基本处理过程所使用的数据不同。

ü   在应用程序边界内，基本处理过程所引用的ILF或EIF是不同于其它基本处理过程所引用的ILF或EIF。

●   EO和EQ通用计算规则必须全部满足以下内容才能被视为一个EO或EQ：

1、  从外部发送数据或控制信息到应用程序边界内。

2、  为了识别这个过程，以下三点必须满足一个：

ü   该基本处理过程逻辑上必须是唯一的，该唯一性是指其在应用程序中与其他EO或EQ的逻辑性上保持唯一。

ü   该基本处理过程所使用的数据应该是唯一的，该唯一性是指其在应用程序中与其他EO或EQ所使用的数据不同。

ü   该基本处理过程所引用的ILF或EIF文件应该是唯一的，该唯一性是指其在应用程序中与其他EO或EQ所引用的ILF或EIF文件不同。

●   EO补充的计算规则除了要满足上面的通用规则外，还要满足下面其中一条：

ü   在基本操作过程中至少包含一个数学公式或计算方法

ü   在基本操作过程中要产生派生数据

ü   在基本操作过程中至少要维护一个ILF

ü   在基本操作过程中要改变系统的行为。

●   EQ补充的计算规则除了要满足上面的通用规则外，还要满足下面其中一条：

ü   基本操作过程从ILF或EIF中获取数据。

ü   基本操作过程不能包含数学公式或计算方法。

ü   基本操作过程不能生成派生数据

ü   基本操作过程不能维护任何一个ILF

ü   基本操作过程不能改变系统的行为

第二部分：EI、EQ和EO的技术复杂的计算

复杂性取决于FIRs和DETs的数量。FTR是被一个事物操作读取或维护的一个ILF，或者是被一个事物操作读取的一个EIF。

EI中识别FTR规则

●   每一个ILF应该算做一个FTR。

●   通过EI读取操作的每个ILF或EIF都应该被计算为一个FTR。

●   即被EI维护又被读取的ILF仅计算一个FTR。

EI中识别DET规则

●   在EI的过程中，以用户角度识别的，通过应用系统边界输入系统内部的非重复的字段，那么该字段应算一个DET。

●   如果在EI过程中，只要没有通过系统边界输入，就算它存在于系统内的一个ILF中，也不能算为一个DET。

ü   例如外贸订单系统中，订单的金额是被单价和数量自动计算的，那么金额是没有通过系统边界输入的，因此在EI操作中就不应该算做一个DET。

●   在应用程序的EI操作时，系统提示的错误信息或完成操作的信息，应该被分别计算为一个DET。

ü   例如在网站注册用户信息时，由于输入错误系统会显示提示信息，那么这些提示信息应该被逐个计算为一个DET。

ü   当EI操作完成时系统提示并显示出来的信息，应该被计算为DET。

●   在EI操作中如果遇到主外键的字段，应该算作一个DET。

EO和EQ计算FTR的规则

●   通用规则：每个在EO/EQ处理过程中读取的ILF和EIF算一个FTR

●   EO额外的FTR计算规则

ü  在EO处理过程中每个被维护的ILF算一个FTR

ü  在EO处理过程中既被读取又被维护的ILF算一个FTR

EO和EQ计算DET的通用规则

●   用户可识别的非重复的字段，进入应用边界并且指明处理什么，何时处理或处理方式，并且由EO/EQ返回或产生，那么这样的每个字段算一个DET

ü   例如在报表中的每个字段都是一个DET

●   在应用边界内以用户角度识别的，非重复字段算一个DET。

ü   例如在报表上起到解释或备注作用的文字信息，不管它是一个字、一个词或一段话，都当作一个DET

ü   例如某种编号或日期，就算它被物理存储在不同字段中，但从用户角度来看是一个整体的信息，因此被算作一个DET

ü   例如在饼图中百分比和分类算作不同的DET。

●   在EO或者EQ操作中，如果对系统进行输入或读取操作时，相同的字段只计算一个DET。

ü   例如在报表查询时，输入的字段在报表上也有显示，那么将算作同一个DET

●   在应用程序的EO或EQ操作时，系统提示的错误信息或完成操作的信息，应该被计算为DET。

ü   例如用户查询一个列表时被拒绝，那么拒绝的提示信息就算为一个DET。

●   在EO或EQ操作中如果遇到主外键的字段，应该算作一个DET。

●   如果在EO或EQ过程中，只要没有通过系统边界输入，就算它存在于系统内的一个ILF中，也不能算为一个DET。

ü   在公司发工资的时候，员工对应的状态信息被更新，但这个状态信息的更新是没有通过系统边界输入的，因此也不能算做一个DET。

●   页面的标题等类似的信息不计算DET

●   系统字段生成的记号不能被算作一个DET。

ü   例如：页码、位置信息、时间、上一页、下一页等信息。

EI复杂度计算矩阵

  1~4个DET 5~15个DET 多于16个DET

0~1个FTR 低 低 中等

2个FTR 低 中等 高

大于2个FRT 中等 高 高

 EO和EQ复杂度计算矩阵

  1~5个DET 6~19个DET 多于20个DET

0~1个FTR 低 低 中等

2~3个FTR 低 中等 高

多于4个FTR 中等 高 高

 未调整前功能点对应矩阵EI、EO、EQ、ILF和EIF计算出来的技术复杂度对应的功能点如下表所示

  低 一般 高

EI 3 4 6

EO 4 5 7

EQ 3 4 6

ILF 7 10 15

EIF 5 7 10

————————————————

版权声明：本文为CSDN博主「zhang\_jim」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/zhang\_jim/article/details/2578766