**Actor的权重计算**

Actor权重参考表

|  |  |
| --- | --- |
| 接口类型 | Actor权重 |
| 类DOS型界面 | 0.8 |
| 简单的对话型界面 | 1.6 |
| 复杂的对话型界面 | 2.3 |
| 简单的GUI界面 | 2.4 |
| 复杂的GUI界面 | 3.0 |

Actor的权重合计表

|  |  |  |
| --- | --- | --- |
| Actor | Actor权重 | 理由 |
| 用户 | 3.0 | 复杂GUI界面 |
| 管理员 | 3.0 | 复杂GUI界面 |
| 二级管理员 | 2.4 | 简单的GUI界面 |
| 合计 | 8.4 |  |

**UseCase权重的计算**

UseCase权重参考表

|  |  |  |
| --- | --- | --- |
| 类型 | 意义 | 权重（系数） |
| 简单型 | 3个以下事务/4个以下分析类 | 4～7 |
| 平均型 | 4至7个事务/5至10个分析类 | 8～12 |
| 复杂型 | 8个以上事务/11个以上分析类 | 13～17 |

UseCase权重合计表

|  |  |  |
| --- | --- | --- |
| UseCase名 | 权重 | 理由 |
| 查看事件地图  事件搜素 | 10  5 | 平均型  简单型 |
| 浏览事件详细信息 | 4 | 简单型 |
| 事件可信度标记  事件上报  订阅  事件类型管理  事件管理  推送事件  管理自己上传的数据  提供报案信息 | 4  7  4  8  12  4  8  4 | 简单型  简单型  简单型  平均型  平均型  简单型  简单型  简单型 |
| 合计 | 70 |  |

**UUCP计算(UUCP：Unadjusted Use Case Point)**

UUCP＝∑Actor权重+∑UseCase权重＝78.4

**技术复杂度（TCF）计算**

将待开发系统所涉及的一些技术因素系数根据涉及程度赋予从0到6之间的值(0：未涉及； 3：一般性；5：本质性)。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 类目 | 权重 | 系数 | 值 | 理由 |
| T1 | 分布式系统 | 0 | 1 |  |  |
| T2 | 系统吞吐量大、处理时间短 | 1 | 1 |  |  |
| T3 | 在线用户使用效率高 | 0 | 1 |  |  |
| T4 | 内部处理复杂 | 2 | 1 |  |  |
| T5 | 代码可供后续项目复用 | 3 | 1 |  |  |
| T6 | 易安装调试 | 1 | 1 |  |  |
| T7 | 易用性高 | 1 | 1 |  |  |
| T8 | 移植性高 | 1 | 1 |  |  |
| T9 | 易扩展和修改 | 5 | 1 |  |  |
| T10 | 并行处理 | 3 | 1 |  |  |
| T11 | 需要特别考虑系统的安全性 | 3 | 1 |  |  |
| T12 | 第三方软件能够直接访问 | 0 | 1 |  |  |
| T13 | 需要对用户进行特别培训 | 0 | 1 |  |  |
| 合计 | TFactor=∑权重×系数 | 20 |  |  |  |

由上表得到TFactor的值，根据下面的算式得出该项目技术复杂度的权重

TCF(Technical Complexity Factor)＝0.6 + (0.01×TFactor) ＝0.8

**环境复杂度（EF）权重的计算**

根据项目开发的具体环境，给下列环境因素的系数赋予0-6之间的值

E1-E4   0 无经验 3 一般 5 专家级

E5     0 无热情 5 高度热情

E6     0 极端不安定 3 平均 5 不变

E7     0 无复用    5 高度复用（复用度80%以上）

E8     0 效率低下  2 一般 5 开发高度自动化

E9     0 公司内/全职开发 5 外包/兼职开发

E10    1 简单开发语言 3 平均 5 非常难的开发语言

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 类目 | 权重 | 系数 | 值 | 理由 |
| E1 | 开发人员对开发流程的熟悉度 | 1 | 1 |  |  |
| E2 | 开发人员对该类应用的开发经验 | 0.5 | 1 |  |  |
| E3 | 开发人员对开发方法的习惯程度 | 0.5 | 1 |  |  |
| E4 | 项目经理的能力 | 3 | 1 |  |  |
| E5 | 开发人员对该项目的热情度 | 3 | 1 |  |  |
| E6 | 开发式样的稳定性（很少变更） | 3 | 1 |  |  |
| E7 | 该项目的复用程度 | 2 | 1 |  |  |
| E8 | 采用开发工具的开发效率 | 1 | 1 |  |  |
| E9 | 开发人员的来源及工作方式 | -0 | 1 |  |  |
| E10 | 程序设计语言的难度 | -2 | 1 |  |  |
| 合计 | EFactor=∑权重×系数 | 12 |  |  |  |

由上表得到EFactor的值，根据下面的算式得出该项目技术复杂度的权重

EF(Environmental Factor)＝1.6 - (0.03×EFactor) ＝1.24

**UCP(Use Case Point)的计算**

UCP=UUCP×TCF×EF=78.4×0.8×1.24≈77.8

**计算项目风险系数**

（风险系数 = E1-E8中2以下的值与E9-E10中4以上的值的总和）

风险系数为3 → 1UCP = 26人时

**估算工作量**

1人月 = 20人日 = 80人时

|  |  |  |
| --- | --- | --- |
| **规模** | **77.8** | **UCP** |
| **生产率** | **6.5** | **人日/UCP** |
| **工作量** | **25** | **人月** |

**估算总时间**

Time（月）= C\*工作量^P

C为进度调整系数（取3），P为组织内部开发能力调整系数（取0.4）

Time = 3\*25^0.4 = 11月

**估算方法简介**

用例分析技术为软件需求规格化提供了一个基本的元素，而且该元素是可验证、可度量的用例通常由评估业务领域的战略目标和场景组成，它们也用来说明软件应用的复杂度。用例用来描绘一个系统外在可见的需求情况，是代表系统中各个项目相关人员（[风险](http://www.csai.cn/incsearch/search.asp?key=%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD)承担人，Stakeholder）之间就系统的行为所达成的契约用例可以作为项目计划、[进度控制](http://www.csai.cn/incsearch/search.asp?key=%EF%BF%BD%EF%BF%BD%C8%BF%EF%BF%BD%EF%BF%BD%EF%BF%BD)、测试等环节的基础。而且用例还可以使开发团队与客户之间的交流更加顺畅。

A、用例数估算前提条件

通过分析用例的角色和场景，可以得到可靠的一个应用程序的规模和工作量的估算值。用例建模是一种得到广泛使用的技术，用来获取软件应用的业务过程和需求。用例模型的建立帮助划定项目的功能边界及功能结构，它为实现自上而下的估算提供基础。用例点数是一个项目估算方法，它使用项目的用例来准确的估算项目规模和工作量。

B、用例数估算步骤

角色复杂度等级划分及计数

用例复杂度等级划分及计数

计算未平衡用例点数

使用技术复杂度因子和环境复杂度因子平衡

估算项目规模和开发工作量

（1）角色复杂度等级划分及计数

（2）用例复杂度等级划分及计数

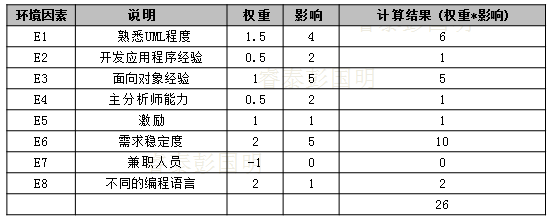
（3）计算未平衡用例点数

将UAW与UUCW相加得出未平衡用例点数（ Unadjusted Use Case Point ，UUCP）

UUCP=UAW+UUCW

（4）使用技术复杂度因子平衡

[](http://photo.blog.sina.com.cn/showpic.html#blogid=93c2f6700101384z&url=http://s2.sinaimg.cn/orignal/93c2f670gc3e814effa01)  
 TCF = 0.6 + (.01\*Total Factor). 图1中, TCF = 1.07

（5）使用环境复杂度因子平衡  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=93c2f6700101384z&url=http://s2.sinaimg.cn/orignal/93c2f670g79fd9bb09671)  
 ECF = 1.4 + (-0.03\*Total Factor). ECF = 0.62

（6）软件规模用例点数计算公式如下

  软件规模举例：UCP=TCF\*ECF\*UUCP

C、总结

如果没有历史数据可供参考，行业专家推荐使用15-30之间的一个数字，典型的，你可以使用20人时/UCP。

与专家估算法相比，用例点数估算可能偏高。为了得到准确结果，公式中的变量需要进行调整，特别是刚开始这样估算时。

软件功能分解（WBS）对估算的有效性有很大影响，应以3至5层之间为宜。

场景中的步数。很多的步数将使复杂性偏高，用例点数增大。较少的步数将使复杂性偏低。

包含和扩充用例增大了复杂性。将这些记为单独的用例。

用例中的角色数也影响估算。如果可能，将角色归纳为更高层次的角色。这将减少复杂性而不影响用例。

技术和环境因素值需要根据不断获得的实际数据来做调整。越多项目使用用例点数来估算，估算将越准确。

生产力因素只能从历史资产中获得。在没有得到符合公司实际的生产力基准数据前，将生产力划为3个等级：高级工程师15人时/UCP；中级工程师20人时/UCP；初级工程师30人时/UCP；