**OSS事业部 - 有效输出计算方法说明**

|  |  |
| --- | --- |
| 部 门： | 产品测试部 |
| 撰 写： | 施昊华 |
| 文档编号： | BOCO-QC-20170704 |

修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **作者/修改者** | **描述** | **审核人** |
| 2017-07-04 | 1.00 | 施昊华 | 创建 |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 概述

当前的有效输出，按照工作内容共分为三类：研发类有效输出、需求类有效输出、测试类有效输出；

其中：

**研发类有效输出包括：**

* 任务开发得分；
* 被驳回merge\_request的扣分；
* 审核merge\_request的得分；
* 解决bug得分；
* 引起Bug的纠责扣分；

**需求类有效输出包括：**

* 评审一次通过的需求得分；
* 提交的需求关联任务有效得分；
* 现场满足度得分<暂时未计算>；
* 引起Bug的纠责扣分；

**测试类有效输出包括：**

* 提交测试用例得分；
* 提交测试bug得分；
* 引起Bug的纠责扣分；

说明：有效输出以**工作内容**为区分，不以人员角色为区分；同一个人可以根据他的实际工作在这三项输出上都有分数。 比如，某同事即作为研发进行了开发工作，也作为需求在禅道上提交了需求，同时还进行了测试提交了发现的bug，那么他的有效输出则包含了研发有效输出、需求有效输出和测试有效输出三部分，他的有效输出总分是这三部分工作输出的和。即：

**个人总有效输出 = 研发工作输出 + 需求工作输出 + 测试工作输出；**

下表为禅道上呈现的有效输出信息，根据人员的实际工作情况进行呈现各部分的有效输出；若只有研发有效输出，则下面的需求和测试有效输出部分不会展现；



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **姓名：** | XXX | **任务数：** | X个 |
| **研发有效输出** | **任务有效输出：** | **新增行数 \* 代码类型权重 \* 0.9（新增操作占比） + 删除行数 \* 代码类型权重 \* 0.1（删除操作占比）**  **（注：代码类型权重见本文第2.1.1章节）** | | |
| **审核（PR）有效输出：** | **合并请求驳回次数 \*（-10）（驳回系数） + （作为审核人）审核通过的代码行数 \* 审核比重**  **（注：审核比重见本文第2.1.3章节）** | | |
| 审核被通过次数： | 2个 | 审核被驳回次数： | - |
| 作为审核人审核通过次数： | - | 作为审核人驳回次数： | - |
| **Bug有效输出：** | **解决的问题数量 \* 30（解决问题系数） + 纠责问题数 \* （-30）（纠责问题系数）** | | |
| 修改自测Bug数： | 2个 | 产生自测Bug数： | - |
| 修改现场Bug数： | - | 产生现场Bug数： | - |
| Bug报工工时： | - |  |  |
| **需求有效输出** | **评审通过得分：** | **评审一次通过的需求个数 \* 100（需求基础分）** | | |
| 评审一次通过数： | - | 评审多次通过数： | - |
| **需求复杂度得分（根据研发得分）：** | **此需求相关任务对应的研发任务有效输出 \* 0.5（需求复杂度系数）** | | |
| **现场bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 现场bug承担责任个数： | - |
| **内部bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 内部bug承担责任个数： | - |
| **现场满足度得分：** | **此需求相关任务对应的研发任务有效输出 \* 需求满足度系数**  **（注：需求满足度系数参见本文第2.2.3章节）** | | |
| **测试有效输出** | **测试用例得分：** | **测试用例个数 \* 20（测试用例系数）** | 测试人员用例个数： | - |
| **测试上报bug得分：** | **测试Bug个数 \* 测试Bug系数**  **（注：Bug系数见本文第2.3.2章节）** | 测试人员上报bug数量： | 2个 |
| **现场bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 现场bug承担责任个数： | - |
| **内部bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 内部bug承担责任个数： | - |

## 各有效输出详细说明

### 研发有效输出

目前的研发有效输出包含以下几部分：

* 任务开发得分；
* 被驳回merge\_request的扣分；
* 审核merge\_request的得分；
* 解决Bug得分；
* 引起Bug的纠责扣分；



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **研发有效输出** | **任务有效输出：** | **新增行数 \* 代码类型权重 \* 0.9（新增操作占比） + 删除行数 \* 代码类型权重 \* 0.1（删除操作占比）**  **（注：代码类型权重见本文第2.1.1章节）** | | |
| **审核（PR）有效输出：** | **合并请求驳回次数 \*（-10） + （作为审核人）审核通过的代码行数 \* 审核比重**  **（注：审核比重见本文第2.1.3章节）** | | |
| 审核被通过次数： | 2个 | 审核被驳回次数： | - |
| 作为审核人审核通过次数： | - | 作为审核人驳回次数： | - |
| **Bug有效输出：** | **解决的问题数量 \* 30（解决问题系数） + 纠责问题数 \* （-30）（纠责问题系数）** | | |
| 修改自测Bug数： | 2个 | 产生自测Bug数： | - |
| 修改现场Bug数： | - | 产生现场Bug数： | - |
| Bug报工工时： | - |  |  |

详细说明：

#### 任务开发得分

**前提条件：**

在禅道上有开发任务，例如某开发任务id为1234；

负责开发的研发在git相关项目上，建立feature-1234分支；注意分支命名符合规范；

研发开发完成并提交和推送此分支后，发起merge\_request，将此feature分支合并入develop分支，别表明审核人（assignee字段）；审核人对此merge\_request进行审核，通过或者驳回申请；

有效输出代码量只统计merge\_request中审核通过的部分；（对于驳回的merge\_request按次进行扣分，参见第2）点被驳回merge\_request的扣分）

计算方法：

**Feature开发有效输出 =**

**新增行数 \* 代码类型权重 \* 0.9（新增操作占比） + 删除行数 \* 代码类型权重 \* 0.1（删除操作占比）**

代码类型权重当前设置如下：

Java:5 python:5 cs:5 c:5 h:5 m:5 mm:5 md:5

as:3 mxml:3 perl:3 swift:3 js:3 jsp:3 css:3

htm:2 html:2 gv:2

xml:1 scala:2 sql:1 sh:1 properties:1 other:1

其他文件类型等同代码行数:

Jpg:2 png:2;

添加、删除操作类型:

新增操作占比: 0.9 删除操作占比: 0.1；

#### 被驳回merge\_request的扣分

研发人员所提交的merge\_request，经过审核人审核，被驳回（closed），则对研发进行扣分。

**被驳回merge\_request分数 =** **合并请求驳回次数 \*（-10）**

#### 审核merge\_request的得分

作为研发负责人审核他人或自己所提交的merge\_request，根据审核的代码量分别计算：

**代码审核有效输出=**

（作为审核人）**审核通过的代码行数 \* 审核比重**

其中：

当审核代码小于50行，则认为审核行数为最小值50,审核比重为0.1,得分为：50\*审核比重0.1=5分；

当审核代码行数大于50行但小于1000行，则审核比重为0.12，得分为：实际审核行数\*审核比重0.12；

当审核代码大于1000行，则认为审核行数为封顶1000行，审核比重为0.15，得分为：1000\*审核比重0.15=150分；

#### 解决Bug的得分

研发人员所解决的bug，根据来源不同分别乘以不同的系数算分。

**问题解决有效输出 = 解决的问题数量 \* 解决问题系数**

Bug来源分为：产品线内部和现场；

目前两种来源的系数均为：30，即每解决一个bug得30分。

#### 引起Bug的纠责扣分

对禅道上的简单问题，在禅道上若对此问题中的研发责认人填写了某人，则对此人进行问题纠责扣分，根据来源不同分别乘以不同的系数算分。

**问题纠责扣分 = 纠责问题数 \* 纠责问题系数**

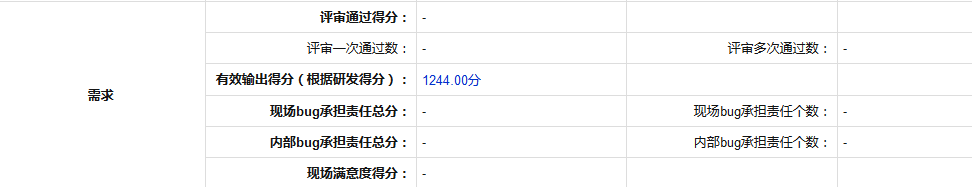
Bug来源分为：产品线内部和现场；

目前两种来源的纠责系数均为：-30，即每引起一个bug扣掉30分。

### 研发有效输出

目前的需求有效输出包含以下这几部分：

* 评审一次通过的需求得分；
* 需求复杂度得分；
* 现场满足度得分<暂时未计算>；
* 引起Bug的纠责扣分；



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **需求有效输出** | **评审通过得分：** | **评审一次通过的需求个数 \* 100（需求基础分）** | | |
| 评审一次通过数： | - | 评审多次通过数： | - |
| **需求复杂度得分（根据研发得分）：** | **研发开发此需求的有效输出 \* 0.5（需求复杂度系数）** | | |
| **现场bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 现场bug承担责任个数： | - |
| **内部bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 内部bug承担责任个数： | - |
| **现场满足度得分：** | **研发开发此功能的有效输出 \* 需求满足度系数**  **（注：需求满足度系数参见本文第2.2.3章节）** | | |

详细说明：

#### 评审一次通过的需求得分

在禅道上的需求，最终确认为评审一次通过，则每个需求乘以需求基础系数为该需求得分；

**评审一次通过需求得分 = 评审一次通过的需求个数 \* 100（需求基础分）**

#### 需求复杂度得分（提交的需求关联任务有效得分）

需求的功能完成开发后，研发人员就此功能的有效输出乘以需求复杂度系数，则为此需求的复杂度得分；

**需求复杂度得分 = 研发开发此功能的有效输出 \* 0.5（需求复杂度系数）**

#### 现场满足度得分<暂时未计算>

需求的功能完成开发后，研发人员就此功能的有效输出根据现场反馈乘以需求满足度系数，则为此需求的满足度得分；

**需求满足度得分 = 研发开发此功能的有效输出 \* 需求满足度系数**

满足度系数按照现场反馈评级：

A: 0.5

B: 0.3

C: 0.2

D: 0

E: -0.2

#### 引起Bug的纠责扣分

对禅道上的简单问题，在禅道上若对此问题中的需求责认人填写了某人，则对此人进行问题纠责扣分，根据来源不同分别乘以不同的系数算分。

**问题纠责扣分 = 纠责问题数 \* （-20）纠责问题系数**

Bug来源分为：产品线内部和现场；

目前两种来源的纠责系数均为：-20，即每引起一个bug扣掉20分。

### 测试有效输出

目前的需求有效输出包含以下这几部分：

* 提交测试用例得分；
* 提交测试bug得分；
* 引起Bug的纠责扣分；



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试有效输出** | **测试人员用例得分：** | **测试用例个数 \* 20（测试用例系数）** | 测试人员用例个数： | - |
| **测试人员上报bug得分：** | **测试Bug个数 \* 测试Bug系数**  **（注：Bug系数见本文第2.3.2章节）** | 测试人员上报bug数量： | 2个 |
| **现场bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 现场bug承担责任个数： | - |
| **内部bug承担责任总分：** | **纠责问题数 \* （-20）纠责问题系数** | 内部bug承担责任个数： | - |

详细说明：

#### 提交测试用例得分

根据在禅道上提交测试用例个数，计算测试用例得分；

**测试用例得分 = 测试用例个数 \* 20（测试用例系数）**

#### 提交Bug得分

根据在禅道上创建的bug个数，根据Bug的严重程度乘以不同系数计算测试Bug得分；

**测试Bug得分 = 测试Bug个数 \* 测试Bug系数**

其中，测试Bug系数按照Bug的严重程度分别为：

一级Bug：100分

二级Bug：80分

三级Bug：30分

四级Bug：20分

五级Bug：10分

#### 引起Bug的纠责扣分

对禅道上的简单问题，在禅道上若对此问题中的测试责认人填写了某人，则对此人进行问题纠责扣分，根据来源不同分别乘以不同的系数算分。

**问题纠责扣分 = 纠责问题数 \* 纠责问题系数**

Bug来源分为：产品线内部和现场；

目前两种来源的纠责系数均为：-20，即每引起一个bug扣掉20分。

<说明：以上系数或权重值后续有可能根据实际计算结果情况有所调整>