软件需求工程-实验报告

Lab3：软件需求的跟踪分析

2019.12

一、 小组成员

学号	姓名	成绩分配比例
161220056	敬舒舒	0.25
161271030	张梦窈	0.25
161290019	吴雨昕	0.25
171860519	李培剴	0.25

二、 数据获取

本次实验基于 Intellij 项目的 Pull Request，对每条 Pr 进行需求分析并对需求变更进行追踪，共计 1194 条 Pull Request 记录。

每条 Pull Request 中的记录分为讨论内容 discuss 和推送记录 commit，而这些信息分别在两个页面中，因此对于每一条 pr 我们都要对两个页面进行爬虫。以 id 为 1261 的 pr 为例，需要爬取的页面为：

https://github.com/JetBrains/intellij-community/pull/1261
https://github.com/JetBrains/intellij-community/pull/1261/commits

对 pr 记录的发布时间和内容进行抓取，如果是 commit，还将抓取显示其变更对比的 url。在获取数据过程中利用 beautifulsoup 库中的 find 系列方法能帮我们准确定位到 html 中的标签。

需要注意，存在讨论文本为空的情况，这时候将在讨论内容中填充 None 作为标记。

三、 需求跟踪分析

需求处理程序 tidy_pr.py 中有 3 个类：item, record 和 Transaction。每个 item 一个 Pull Request 一个阶段的需求。一个 item 可能包含一条或若干条 record，一条 record 表示一次发言/一次提交。Transaction 类负责处理 Pull Request，并利用 item 和 record 进行对结果进行包装和写出。Transaction 类是整个处理程序的主体。

首先需要明确的是，每条记录都有 discuss 和 commit 二者之一的标签。在 Transaction 类中，先将所有记录按照时间先后排序。然后根据 discuss 和 commit 标签的交替来判断需求变更。具体操作为将所有连续的 discuss 或 commit 记录到同一个 item 中，出现不连续时停止在这个 item 中的记录，到新的 item 中进行记录。于是可以简单地认为，某一个 Pull Request 的需求变更次数与对其处理得的 item 数正相关。在将处理结果写出到 timeline 文件时，也会根据 item 数量输出不同的信息，如下图所示。

```
if (len(self.timeline) == 0):  # 没有具体内容
        f.write('timeline of ' + self.id + ' is empty.\n')
elif (len(self.timeline) == 1):  # 只有一次提交或者一次讨论，没有形成完整时间线
        if (self.timeline[0].typ != 'implements'):
                f.write('the requirement of ' + self.id + ' has no implements.\n')
        else:
                f.write('the requirement of ' + self.id + ' only has implements, no discussion.\n')
elif (len(self.timeline) == 2):  # 只有一次提交和一次讨论，也没有形成完整时间线
        f.write('*==========================================================================\n')
        f.write('* no changes, only the initial requirement, implements, and discussion.\n')
        f.write('*==========================================================================\n')
        for x in self.timeline:
                x.save(f)
else:  # 有多次commit和discussion交互的时间线
        i = 0
        while (i < len(self.timeline)):
                f.write('*==========================================================================\n')
                f.write('* round ' + str(int(i / 2))+'\n')
                f.write('*==========================================================================\n')
                self.timeline[i].save(f)
                try:
                        self.timeline[i + 1].save(f)
                except:
                        pass
                i += 2
```

四、 结果分析

实验中我们对于每条 pull request 均进行了跟踪分析，得到了大量需求及变更信息。这里以其中一条记录为例。

（1） 明确需求文本，获取需求的有关讨论文本

例如第 24 号 pull request 最初提出需求：需要语法高亮

```
**********************************************************************
* id: 24
* requirement: New highlights
*======================================================================
```

之后在讨论区我们可以看到针对该需求进行的一些讨论，明确语法高亮在某些情况下没有生效（如抽象方法和继承方法）

```
**********************************************************************
* type: discussion
*----------------------------------------------------------------------
* time: 2012-10-22T19:52:32Z
* content: As a former Eclipse user, I really miss highlights for abstract and inherited method calls. That's why I wrote some code to have it working.
* time: 2012-10-22T19:52:32Z
* content: Implementing abstract method calls hightlights was trivial, but inherited method calls was a little harder as PsiMethod.getMethodReceiver() isn't giving expected information.
* time: 2012-10-22T19:52:32Z
* content: Hope this helps.
* time: 2012-10-23T12:50:44Z
* content: Instead of all this code, please use enclosingClass.isInheritor(method.getContainingClass(), true)
* time: 2012-10-23T12:51:03Z
* content: PsiTreeUtil.getParentOfType(element, PsiClass.class)
* time: 2012-10-23T12:55:16Z
* content: This is wrong. You need to find a parent PsiMethodCallExpression for elementToHighlight, then get its method expression and check what kind of qualifier the method expression has.
* time: 2012-10-23T12:57:04Z
* content: Before we can merge your pull request, you'll need to submit a contributor license agreement: http://www.jetbrains.org/display/IJOS/Contributor+Agreement The easiest way to do so is to print out the PD
**********************************************************************
```

（2） 识别出实现需求 R 的代码

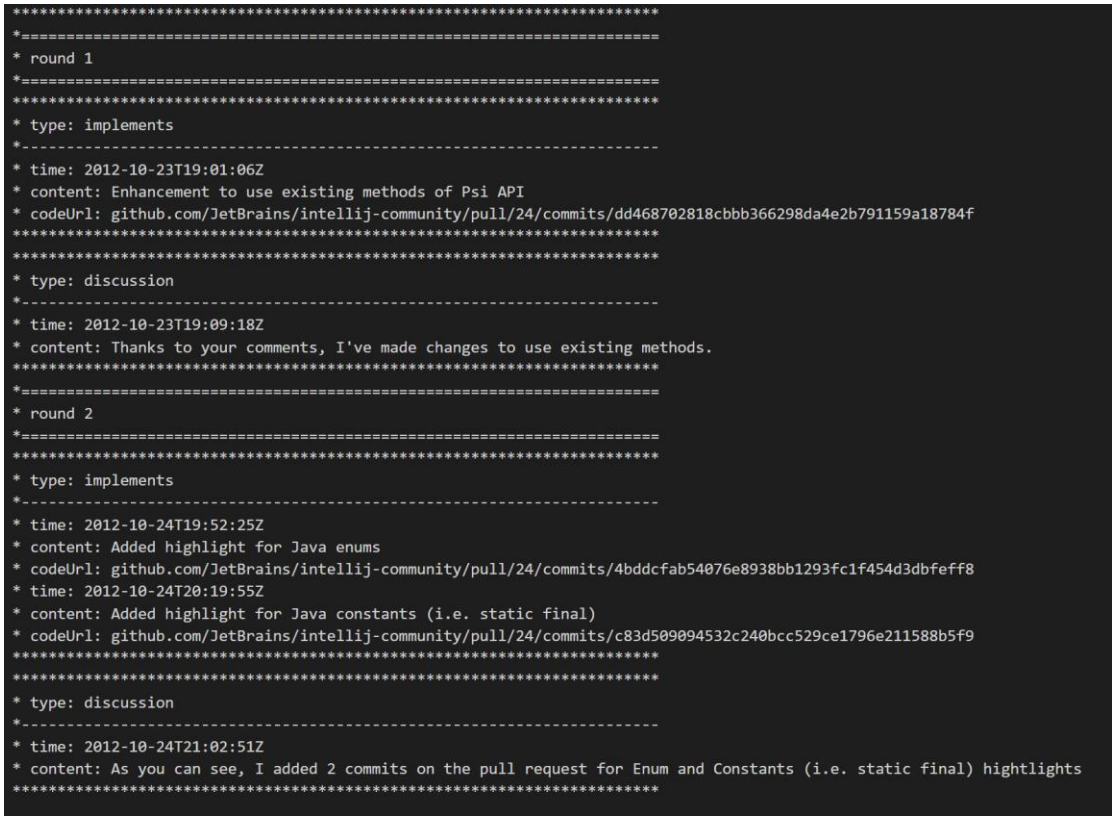
在我们结果中的 implements 部分，我们提取了实现了该需求的几次相关提交，可以看到，图中两次提交分别对之前讨论的抽象方法调用和继承方法调用的语法高亮需求进行了实现。

```
**********************************************************************
* type: implements
*----------------------------------------------------------------------
* time: 2012-10-19T20:44:53Z
* content: Added highlight for Java abstract methods calls
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/943e2d4d697180f268c442cd6003e0faf2eb9a72
* time: 2012-10-22T19:39:11Z
* content: Added highlight for Java inherited methods calls
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/b9cedbcb83242636449ab84b6a9bf5511983acd7
**********************************************************************
```

（3） 识别出需求变更

在实验中我们对于一个需求进行持续追踪，得到其需求变更的时间线，并将其分为了多个 round，而上述需求文本的提取和实现代码提取，均为最初的原始需求，记为 round0.

而对于图中例子，可以看到需求发生了多次变更，我们这里将其分为 round1 和 round2.



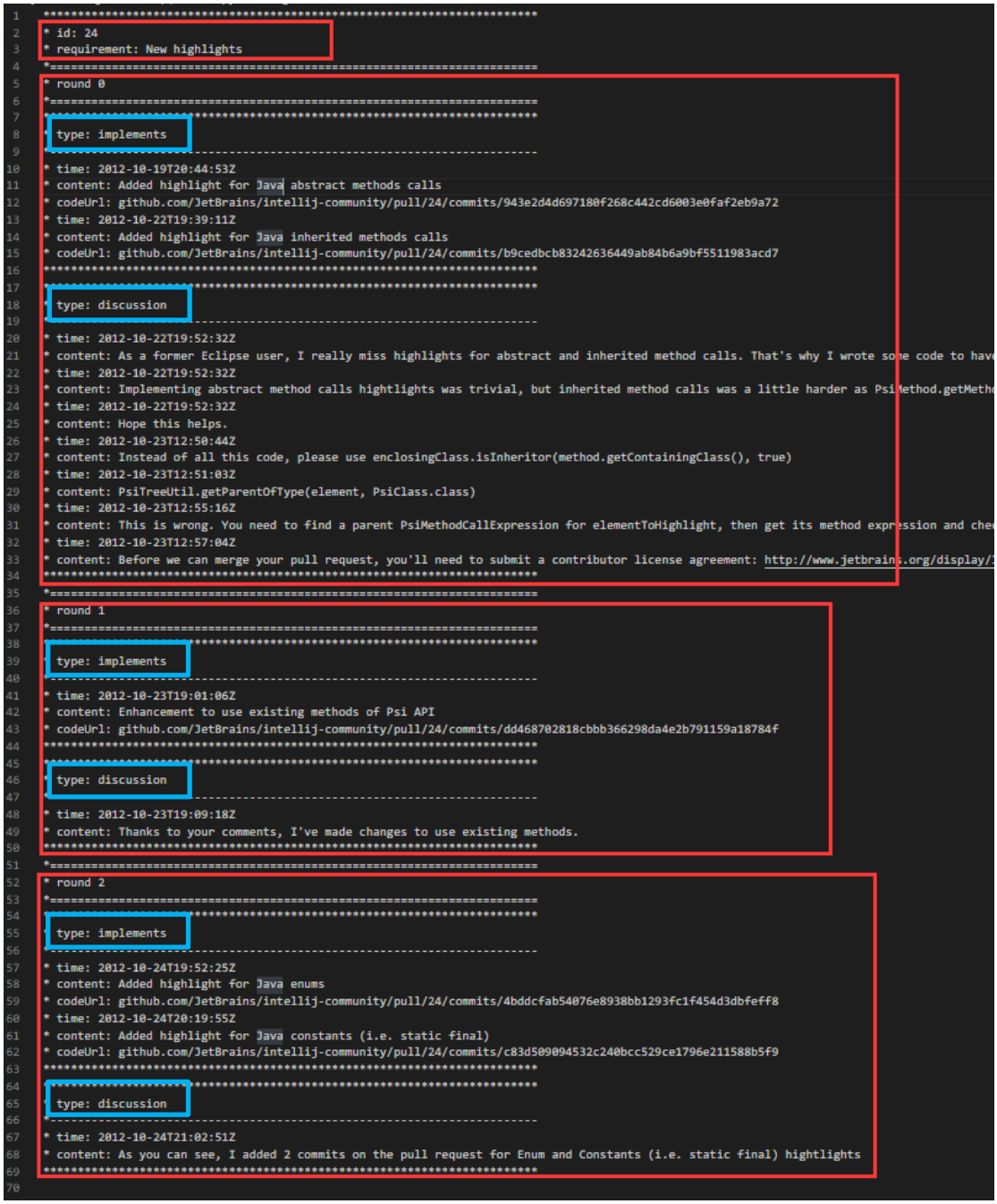
每一次需求变更，都伴随着一轮新的讨论，讨论中可能暴露了新的问题，或者对原有需求的改进。而 implements 表明了对于这次新的需求变更，进行代码实现的对应提交。

（4） 给出需求 R 的全生命周期的时间线

以第 24 号 pr 记录为例，该 pull request 最初提出的原始需求为：需要语法高亮。(图片见下页)

- 在第一轮讨论中，得到了对于特定函数的语法高亮需求，并进行了实现，最终提交是在 2012-10-22
- 第二轮讨论中，将语法高亮的底层实现进行了调整，改用了现有的 api 进行实现。时间为 2012-10-23
- 第三轮讨论中，额外追加了对 Java 中枚举类型的语法高亮。时间为 2012-10-24

在该需求的整个生命周期的时间线上我们可以看到，从最开始单纯需要语法高亮这一原始需求，中途经历了底层实现变更，额外需求的追加等。


* id: 24
* requirement: New highlights
* round 0
* type: implements
* time: 2012-10-19T20:44:53Z
* content: Added highlight for Java abstract methods calls
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/943e2d4d697180f268c442cd6003e0faf2eb9a72
* time: 2012-10-22T19:39:11Z
* content: Added highlight for Java inherited methods calls
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/b9cedbcb83242636449ab84b6a9bf5511983acd7
* type: discussion
* time: 2012-10-22T19:52:32Z
* content: As a former Eclipse user, I really miss highlights for abstract and inherited method calls. That's why I wrote so
* time: 2012-10-22T19:52:32Z
* content: Implementing abstract method calls hightlights was trivial, but inherited method calls was a little harder as Psi
* time: 2012-10-22T19:52:32Z
* content: Hope this helps.
* time: 2012-10-23T12:50:44Z
* content: Instead of all this code, please use enclosingClass.isInheritor(method.getContainingClass(), true)
* time: 2012-10-23T12:51:03Z
* content: PsiTreeUtil.getParentOfType(element, PsiClass.class)
* time: 2012-10-23T12:55:16Z
* content: This is wrong. You need to find a parent PsiMethodCallExpression for elementToHighlight, then get its method expr
* time: 2012-10-23T12:57:04Z
* content: Before we can merge your pull request, you'll need to submit a contributor license agreement: http://www.jetbrains.org/display/
* round 1
* type: implements
* time: 2012-10-23T19:01:06Z
* content: Enhancement to use existing methods of Psi API
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/dd468702818cbbb366298da4e2b791159a18784f
* type: discussion
* time: 2012-10-23T19:09:18Z
* content: Thanks to your comments, I've made changes to use existing methods.
* round 2
* type: implements
* time: 2012-10-24T19:52:25Z
* content: Added highlight for Java enums
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/4bddcfab54076e8938bb1293fc1f454d3dbfeff8
* time: 2012-10-24T20:19:55Z
* content: Added highlight for Java constants (i.e. static final)
* codeUrl: github.com/JetBrains/intellij-community/pull/24/commits/c83d509094532c240bcc529ce1796e211588b5f9
* type: discussion
* time: 2012-10-24T21:02:51Z
* content: As you can see, I added 2 commits on the pull request for Enum and Constants (i.e. static final) hightlights