*论文题目：跨项目缺陷修复的实证分析（原文标题：How do Developers Fix Cross-project Correlated Bugs? A case study on the GitHub scientific Python ecosystem）

*作者：马皖王莹[1]，陈林[1]，张翔宇[2]，周毓明[1]，徐宝文[1]

*单位：1.南京大学，2.Purdue University（普渡大学）

联系方式：wwyma@smail.nju.edu.cn,lchen@nju.edu.cn

*文章发表信息：Proceedings of the 39th International Conference on Software Engineering, Pages 381-392, Buenos Aires, Argentina — May 20 - 28, 2017, IEEE Press Piscataway, NJ, USA

*原文链接地址：http://dl.acm.org/ft_gateway.cfm?id=3097414

文章简介：对 Github 上 Python 科学计算软件生态系统中的跨项目关联缺陷进行了实证分析，聚焦于开发者对缺陷根源的追踪和上下游项目开发者修复缺陷的协作。通过定性和定量的分析，揭示了影响这类缺陷定位与修复的因素，以及开发者应对它们的常见行为。

*正文：

从一个例子开始：

Astropy 是 Python 科学计算软件生态系统中的核心项目之一，它建立在该生态系统中的一个基础项目——NumPy 之上，使用了 NumPy 提供的数据结构和基础功能。

有一天，有人在 Astropy 的 GitHub 缺陷跟踪系统中报告了一个性能缺陷（astropy/astropy#4259），Astropy 的程序员们立刻对该缺陷展开了调查。他们发现该缺陷似乎由上游项目 NumPy 的缺陷引发，而 numpy/numpy#6467 记录了这个 NumPy 缺陷。于是，Astropy 的程序员们参与了这个上游缺陷的修复，并且主动提供了测试用例。经过两个项目的开发者们的共同努力，这个 NumPy 的缺陷很快被修复，Astropy 也摆脱了影响。

我们为什么要研究它？

我们把上述分别报告在上游项目 NumPy 和下游项目 Astropy 中的一对缺陷称为“跨项目关联缺陷”（Cross-project correlated bugs）。图 1 简要记录了这对缺陷的修复过程，该过程可以大致分成两步：首先下游项目程序员进行跨项目的缺陷根源追踪，然后上下游项目协作修复缺陷。

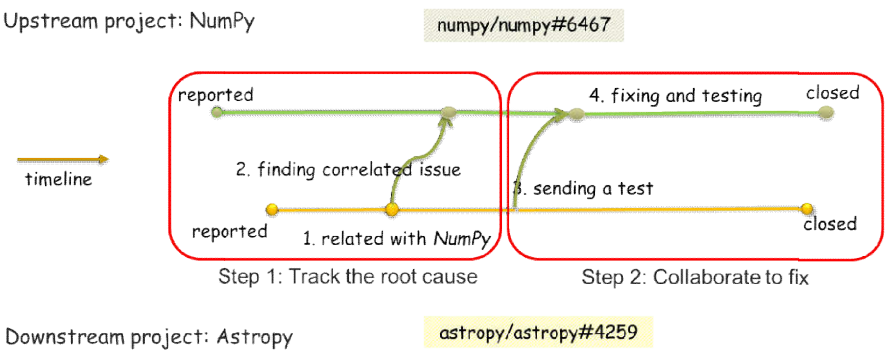

图 1. 跨项目关联缺陷的修复过程

随着软件生态系统的发展壮大，项目之间的相互依赖关系越加复杂，跨项目缺陷的数量必然会逐步增加。直觉来看，跨项目缺陷要比项目内缺陷更麻烦，因为程序员们要在另一个不太熟悉的项目中寻找“罪魁祸首”，还要和另一个项目的程序员配合解决问题。

为了验证这一直觉，我们对 GitHub 上 Python 科学计算生态系统的程序员们开展了问卷调查。调查结果显示，在程序员们遇到的缺陷中，平均 17.3%是跨项目缺陷。相比于项目内缺陷，超过半数的受访者认为跨项目缺陷更难修复，且 40.7%的受访者认为跨项目缺陷会造成更严重的影响。

此外，我们收集了 Python 科学计算生态系统中七个核心项目（Astropy, Ipython, Matplotlib, NumPy, Scikit-learn 和 SciPy）的缺陷报告，并针对修复时间、缺陷报告的评论数量和参与人数，对跨项目缺陷和项目内缺陷进行了比较（Wilcoxon 秩和检验）。结果表明，在大多数情况下，跨项目缺陷在这三个指标上显著高于项目内缺陷。这在一定程度上说明了跨项目缺陷比项目内缺陷更难修复。

结合问卷调查和统计分析，我们可以认为：跨项目缺陷数量不容忽视、修复难度大、影响严重。虽然已有很多工作对项目内缺陷的修复过程进行了详细分析，然而跨项目缺陷的修复过程与之有显著不同，迫切需要对此进行深入研究。

我们想知道什么？

从图 1 可以看出，与项目内缺陷的修复相比，在跨项目缺陷的修复过程中程序员会遇到两个特有的挑战：

1. 跨项目缺陷根源追踪：跨项目缺陷的“罪魁祸首”是在上游项目之中，对于下游项目的开发者来说，他们需要在一个不太熟悉的上游项目中确定缺陷根源，这无疑更加困难。

2. 协作修复：跨项目缺陷的修复进程是不受下游程序员控制的，因为修复需要上游程序员在上游项目中进行。因此在上游项目的修复过程中，为了尽可能减小跨项目缺陷对下游项目的影响，下游程序员往往不能“袖手旁观”，而要主动配合，这在项目内缺陷的修复活动中也是不需要的。

这两个挑战增加了跨项目缺陷的修复难度，因此我们的研究目的就是调查程序员们在面对跨项目缺陷时，如何处理这两个挑战。具体来说，我们设置了三个研究问题：

RQ1：程序员需要多长时间才能找到跨项目缺陷的罪魁祸首？

RQ2：哪些因素有利于程序员追踪到跨项目缺陷的罪魁祸首？

RQ3：在等待上游程序员修复缺陷的过程中，下游程序员又做了什么？

我们做了什么？

我们对 Python 科学计算生态系统中的跨项目缺陷展开了实证研究。我们以上文提到的七个核心项目为中心，共收集了 271 对跨项目关联缺陷，这些缺陷涉及到的上/下游项目共 204 个，图 2 显示了这些项目之间的依赖关系。

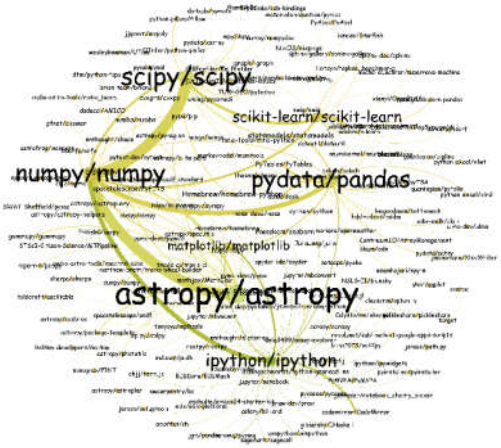

图 2. Python 科学计算生态系统中的部分项目间依赖关系

为回答上述三个研究问题，我们结合了以下两种分析方法：

1. 人工审查：三位研究者分别独立对每个缺陷报告进行详细阅读、记录关键信息，从中总结研究问题的可能答案，然后共同对比讨论各自的结论。

2. 问卷调查：我们对上游和下游程序员设计了不同的问卷，分别包含 9 个和 10 个问题，最后一题为开放性问题，其它均为选择题。为收集受访者的不同意见，对每一个选择题，我们额外增加了一个评论区域。我们向 GitHub 上 Python 科学计算生态系统中的 676 位活跃开发者发送了问卷邀请，最终收到 116 份回复（17.2%的回复率）。

我们得到了什么？

结合人工审查和问卷调查的结果，对于三个研究问题，我们得到以下结论：

1.对于近一半的跨项目缺陷，程序员们需要超过一天的时间来找到缺陷根源；

2.对于下游项目程序员来说，三个主要因素能够帮助他们找到跨项目缺陷的根源：缺陷报告中提供的堆栈信息、与上游程序员间的沟通、以及对上游项目的熟悉程度。

3. 在等待上游修复的过程中，下游程序员采取了多种措施来尽可能减小跨项目缺陷对自身项目的影响，其中，在自身项目注入一个临时解决方案（workaround）是最常见的做法。

跨项目缺陷的修复过程充分体现了开源软件生态系统中不同项目程序员间的协作，例如上下游程序员间的交流沟通、下游程序员对上游项目主动提供帮助等。然而当前的缺陷跟踪系统和缺陷调试修复工具在对程序员协作方面的支持仍不充足。在分析缺陷报告和问卷调查结果后，我们根据程序员的需求，为跨项目缺陷跟踪和支持工具的设计与实现提供了建议。

作者简介：

本文作者包括南京大学博士研究生马皖王莹，南京大学陈林副教授，美国普渡大学张翔宇教授，南京大学周毓明教授和徐宝文教授。