Measuring Project Exposure Using Types used in a Java Project*

Michael Feist
University of Alberta
mdfeist@ualberta.ca

Ian Watts
University of Alberta
watts@ualberta.ca

Lars Thørväld
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

ABSTRACT

A Java project contains many different types which are defined by the language, the developers and included libraries. This paper examines how many of those types are being used by developers in Java projects. A tool for computing the difference between Abstract Syntax Trees (AST) is used to compare revisions in GitHub repositories to find what libraries and types are contributed and changed by different authors. From these differences, the number of types used by an individual developer is calculated. A developer's exposure to different parts of a project is measured by how many out of the total number of types they have used. A comparison between developers is used to see if programmers specialize type usage in their contributions. We find that most projects use a large number of types and most developers only touched a small portion of the total types. We also propose the use of this metric to encourage developers to increase their exposure to more parts of a project.

CCS Concepts

•**Software and its engineering** → *Software organization and properties;*

Keywords

Software Engineering; Mining Software Repositories; Programming Languages; Abstract Syntax Trees

1. INTRODUCTION

From analysing open source software repositories, it can be seen that most contributors make few changes and only touch a small part of the program. The majority of the changes come from a set of core developers. There are also many more people watching repositories than actually contributing to them. Increasing the activity level and encouraging aspiring developers would be beneficial to the overall health of the repository [11]. An active developer might make changes to all areas of a repository or focus on a specific part. There are some metrics for measuring the contributions of a developer to a project, but these do not measure how much of the project a developer has been exposed to and instead measures the size of their contribution [7]. The number of files added and lines of code contributed by an author are not a good metric since the total number is heavily affected by other developers. We propose that if a developer has used most of the different types present in a project, they have likely seen most parts of it. Since most tasks can be completed while only using a small subset of these types, this does not measure the size of their contribution but rather the exposure to the entire scope of the project. In Java, a type can be a primitive type or an object reference. Primitive types are those that are built into the language such as an int or char. Object references are classes that have been defined in the project or through an included library. Different types and libraries are used in different parts of a program. When writing or contributing, the author's choice of which part of the program to edit will in turn affects what types they end up using or editing. The number of types a programmer uses could depend on the level of of experience with a language or the size and scope of the project. When collaborating on open source software, there is the extra considerations of how a developer's contributions mesh with those of their collaborators. Developers typically have a role in development process that is defined by the project, their collaborators and their experience level [11]. A programmer may choose to stay within their comfort areas and not contribute certain features or leave features to be developed by others. As a developer's experience with a project increases they may be more inclined to use more types. We can learn about the types developers are using by examining each revision and finding the types which are being used by the developer making the revision.

In this paper we will attempt to answer the following research questions:

RQ1: Do developers stick to the same types in a project?

RQ2: How many developers in each project had large coverages?

2. RELATED WORKS

The usage of language features in Java and the behaviour of developers has been studied before. Robert Dyer *et al.* [2] mined AST nodes to look at the use of new Java language

*For use with SIG-ALTERNATE.CLS. Supported by ACM.

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

features over time. They found that all new Java features do get used, but there were many places that they could have been used but were not. Some features were not used very often whereas others became widely used. Developers would also modify old code to use new features as they were released. They also checked what were the most popular features and how teams adopted new features. They did not check to see how much a developer used a feature but instead only checked to see if they used it at all.

Grechanik *et al.* [5] examined the structure of Java programs mined from 2080 programs. 32 small research questions are proposed and answered regarding Java usage. They find that most methods do not have any arguments or return values and that inheritance is usually shallow. This paper looks at the breakdown of syntactic structures in open source repositories, however it does not look into per author statistics.

Meyerovich and Rabkin [9] surveyed developers and examined repositories to learn about language adoption and usage. It mentions that developers learn and forget languages over time depending on their education. Developers care more about expressiveness than correctness and feel that certain language features are more important than others.

Hoppe and Hanenberg [6] found that the usage of generic instead of raw types resulted in a an improvement in usability of undocumented APIs, no improvement in the fixing of type errors and a decrease in extensibility. Interaction between developers and IDEs could also be important for the use of language features.

Parnin *et al.* [10] mined repositories to see how Java generics have been used in open source projects. They found that generic usages were often introduced by a single developer in a project. Generics usage was also fairly narrow, with the primary use of them being collecting and traversing lists of objects.

Gorschek *et al.* [4] performed a large-scale empirical study of practitioner's use of object-oriented concepts and found that developers are not consistent with their beliefs or practices. Many developers did not follow what was good in theory and perhaps instead used what worked for them.

Patrick Wagstrom *et al.* [11] explored the roles that developers take in a networked, social development environments like Github. These role were defined as their level of contribution as well as the types of issues they dealt with. Developers will take on multiple roles in a project and will sometimes fulfill the same role across different projects.

Gousios *et al.* [7] came up with metrics to measure a developer's contributions to a repository. They used metrics like number of commits, documentation and activity on bug trackers as positives and bugs committed or poor quality code as negatives to calculate a contribution factor for a developer.

Lämmel *et al.* [8] used ASTs to examine the usage of APIs in Java projects. They find what APIs are popular as well as if they are used in a framework like manner.

These studies look at the usage of language features, APIs and object oriented structures but did not look at the types used in Java projects.

3. METHODOLOGY

3.1 Metric

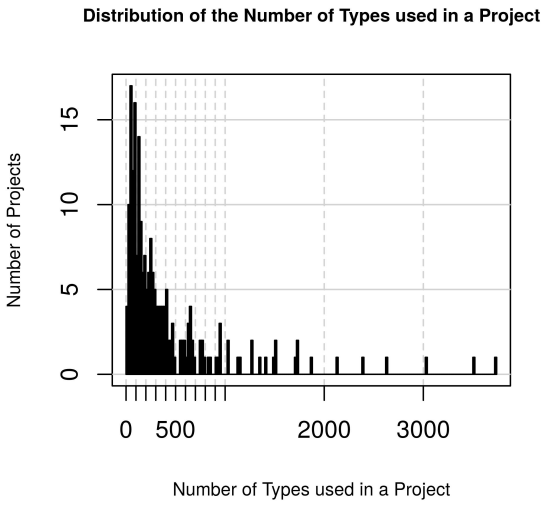

Figure 1: Histogram showing the number of different types declared in a project.

3.2 Data Set

Our data set consists of 216 Github repositories. To ensure that we only look at Java repositories of reasonable size we first queried BOA [1] for eligible repositories. We ran our BOA query over the 2015 Github September dataset. The criteria for a repository to be accept was that it include at least 10 Java files, 3 different committers, 30 commits, and at least one commit from after 2014. This allowed us to narrow our search down to sizable Java projects that were recently active. Out of the possible repositories returned from BOA we randomly chose the 216 repositories and pulled them from Github.

3.3 Approach

To determine what types in a project an author declared we looked at all revisions in each of the 216 Github repositories. Using the Gumtree algorithm with Spoon [3] we were able to generate ASTs for each of the differences between the revisions. Next we only looked at additions and modifications as deletions do not necessarily show that an author is using that type. We then counted the number of times a type is declared in the ASTs. By adding up the types used by each author in a given project we were able to determine the total number of types used in each project. This then allowed us to calculate the percent of types an author modified or added compared to the total number of types in a project.

4. ANALYSIS AND FINDINGS

We found that more than half of the projects used less than 250 different types. But there were a few projects that used thousands of different types.

4.1 RQ1: Do developers stick to the same types in a project?

We found that the majority of developers declare less than 30% of the types declared in the project. With a small group

declaring more than 80% of the types. This is shown in Figure 2.

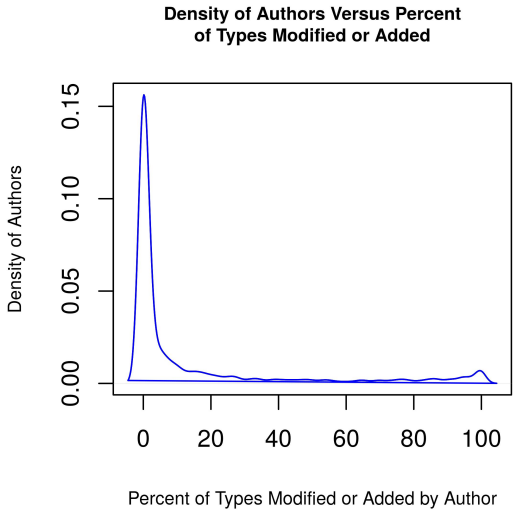

Figure 2: A sample black and white graphic.

4.2 RQ2: How many developers in each project had large coverages?

We found that on average a project has about 7 authors. In these projects on average only two developers modified or added over 50% of the types used in the project. Furthermore on average only one developer modified or added over 80% of the types. This is shown if Figures 3 and 4 respectively.

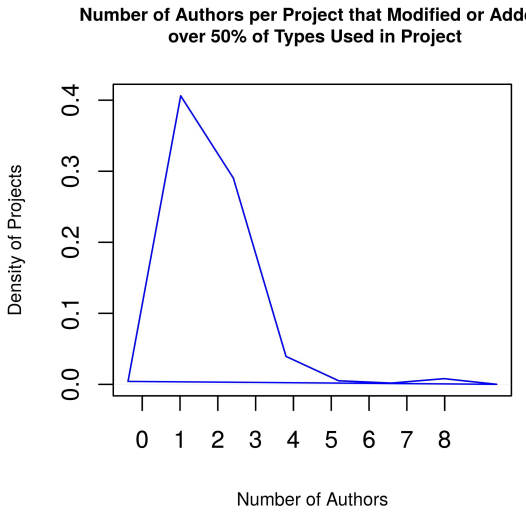

Figure 3: A sample black and white graphic.

Looking at Figure 5 we can see that approximately 70% of the projects have at least one developer who has modified or added over 90% of the types used in a project. Furthermore

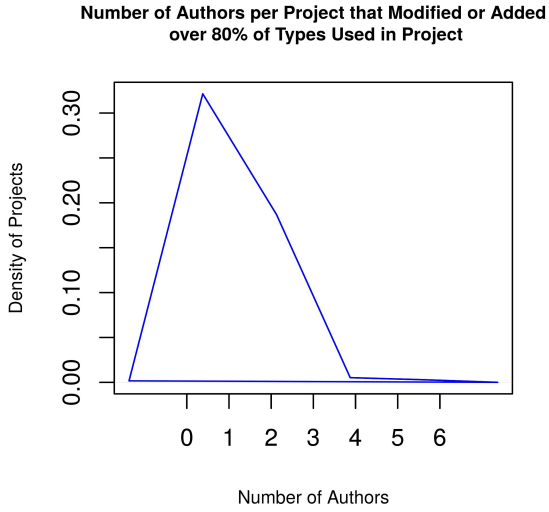

Figure 4: A sample black and white graphic.

all projects have at least one developer that modified or added over 50% of the types.

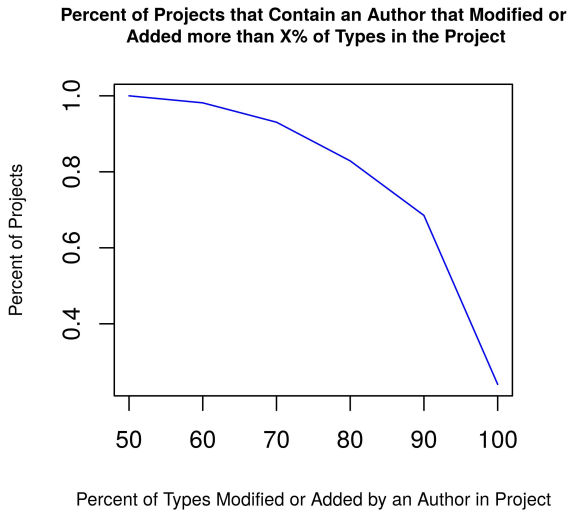

Figure 5: A sample black and white graphic.

5. CONCLUSIONS AND FUTURE WORK

6. ACKNOWLEDGMENTS

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. Boa: A language and infrastructure for analyzing

ultra-large-scale software repositories. In *35th International Conference on Software Engineering*, ICSE 2013, pages 422–431, May 2013.

[2] R. Dyer, H. Rajan, H. A. Nguyen, and T. N. Nguyen. Mining billions of ast nodes to study actual and potential usage of java language features. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 779–790, New York, NY, USA, 2014. ACM.

[3] J.-R. Falleri, F. Morandat, X. Blanc, M. Martinez, and M. Monperrus. Fine-grained and Accurate Source Code Differencing. In *ASE 2014*, page 11 p., France, 2014.

[4] T. Gorschek, E. Tempero, and L. Angelis. A large-scale empirical study of practitioners' use of object-oriented concepts. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 115–124, New York, NY, USA, 2010. ACM.

[5] M. Grechanik, C. McMillan, L. DeFerrari, M. Comi, S. Crespi, D. Poshyvanyk, C. Fu, Q. Xie, and C. Ghezzi. An empirical investigation into a large-scale java open source code repository. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 11:1–11:10, New York, NY, USA, 2010. ACM.

[6] M. Hoppe and S. Hanenberg. Do developers benefit from generic types?: An empirical comparison of generic and raw types in java. *SIGPLAN Not.*, 48(10):457–474, Oct. 2013.

[7] E. Kalliamvakou, G. Gousios, D. Spinellis, and N. Pouloudi. Measuring developer contribution from software repository data. In A. Poulymenakou, N. Pouloudi, and K. Pramatari, editors, *MCIS 2009: 4th Mediterranean Conference on Information Systems*, pages 600–611, Sept. 2009.

[8] R. Lämmel, E. Pek, and J. Starek. Large-scale, ast-based api-usage analysis of open-source java projects. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 1317–1324, New York, NY, USA, 2011. ACM.

[9] L. A. Meyerovich and A. S. Rabkin. Empirical analysis of programming language adoption. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA '13, pages 1–18, New York, NY, USA, 2013. ACM.

[10] C. Parnin, C. Bird, and E. Murphy-Hill. Java generics adoption: How new features are introduced, championed, or ignored. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 3–12, New York, NY, USA, 2011. ACM.

[11] S. Wagstrom, Jergensen. *Roles in a Networked Software Development Ecosystem: A Case Study in GitHub*, 2012.

APPENDIX

A. HEADINGS IN APPENDICES