Feature Usage In Java*

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

Language features A programming language contains many constructs and features to allow the programmer to complete a task in an effective manner. However, most tasks can be compllteded while only using a small subset of these features (reference).

Behaviour of developers Beginner developers use less features(ref)? As a developers experience with a language increases... Use more features

Abstract Syntax Trees An Abstract Syntax Tree (AST) is a tree structrure which represents the language constructs contained in a file. Each node in the tree... Each child in the tree... Comparison of two trees

In this paper we will attempt to answer the following research questions: RQ1:

2. RELATED WORKS

General Description of prior work

Paper 1 has ... Use of java language features over time. There are millions of places features could potentially be used but weren't. Developers convert existing code to use new features They found thousands of instances of potential resource handling bugs. Are the use of languages anticipated? Are they being used before the release date? Some features widely used, others not so much Due to lack of opportunity? Check for situations before and after release of feature Need better tools or training to promote use of features Most popular features: annotation use, enhanced-for loops, and variables with generic types Team vs individual adoption rate of new features PER COMMITER use of features First person to commit a feature is marked as having used that feature

Paper 2 has ...

3. METHODOLOGY

Describe AST diff tools, histogram of constructs

3.1 Metric

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 [2] we were able to generate ASTs for each of the differences between

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

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

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.

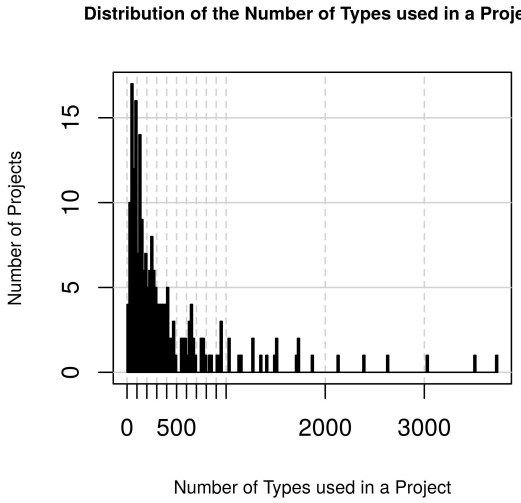

Figure 1: A sample black and white graphic.

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

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

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] 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.

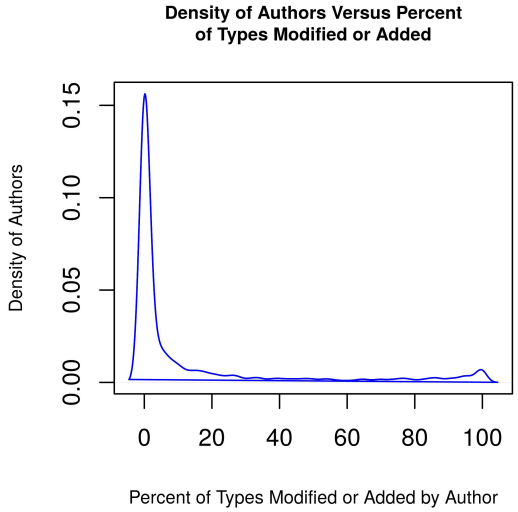

Figure 2: A sample black and white graphic.

APPENDIX

A. HEADINGS IN APPENDICES

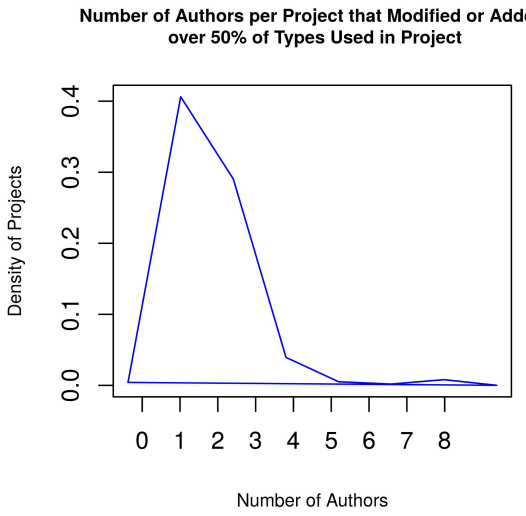

Figure 3: A sample black and white graphic.

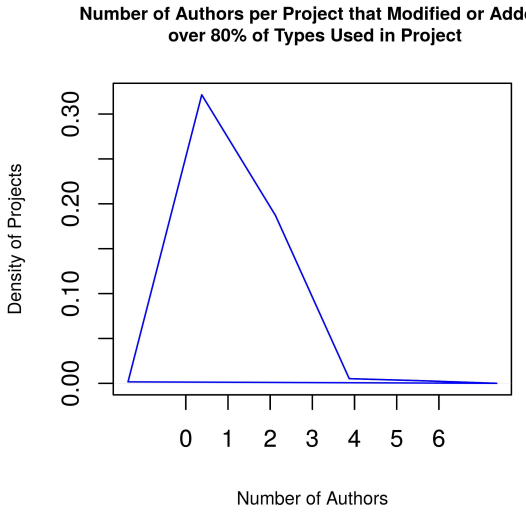

Figure 4: A sample black and white graphic.

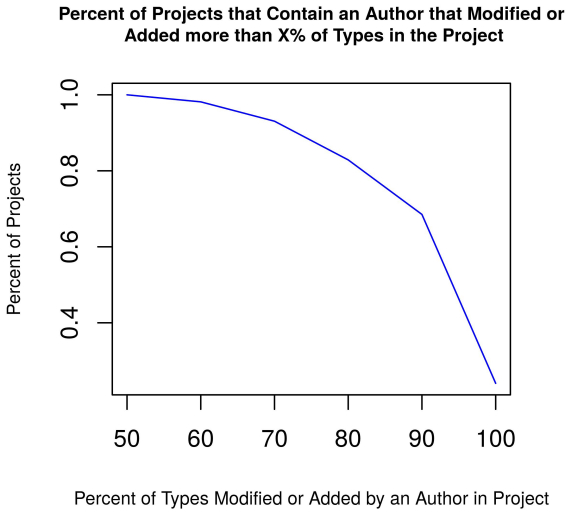

Figure 5: A sample black and white graphic.