19th IEEE International Working Conference on Source Code Analysis and Manipulation

Contextualizing Rename Decisions using Refactorings and Commit Messages

Anthony Peruma, Mohamed Wiem Mkaouer, Michael J. Decker, Christian D. Newman

RIT | Rochester Institute of Technology

BGSU Bowling Green State University

September 30-October 01, 2019

SCANL
Source Code Analysis And Natural Language Laboratory

Hello, we are SCANL Lab!

We study the latent connection between source code behavior and the natural language elements used to describe that behavior

Our research interest includes:

- Program Comprehension and Textual Analysis
- Program Transformation
- Static Source Code Analysis

For more information or to collaborate:

https://scanl.org/

How & Why Identifier Names Change

We know:

A rename can either preserve or change the semantic meaning of the identifiers name.

But:

What motivates a developer to update the semantics of the name? Moreover, can we empirically derive these motivations?

Our prior work[†]:

Contextualized the semantic update with the commit message, yielded interesting, but not concrete results around motivation

[†] Anthony Peruma, Mohamed Wiem Mkaouer, Michael J. Decker, and Christian D. Newman. 2018. **An empirical investigation of how and why developers rename identifiers**. In Proceedings of the 2nd International Workshop on Refactoring (IWoR 2018)

Research Goal & Contributions

Expand our knowledge of understanding the changes made to identifiers during renaming activities

Frequently occurring refactorings with identifier renames

Type of developers introducing performing renames

Replication package availability

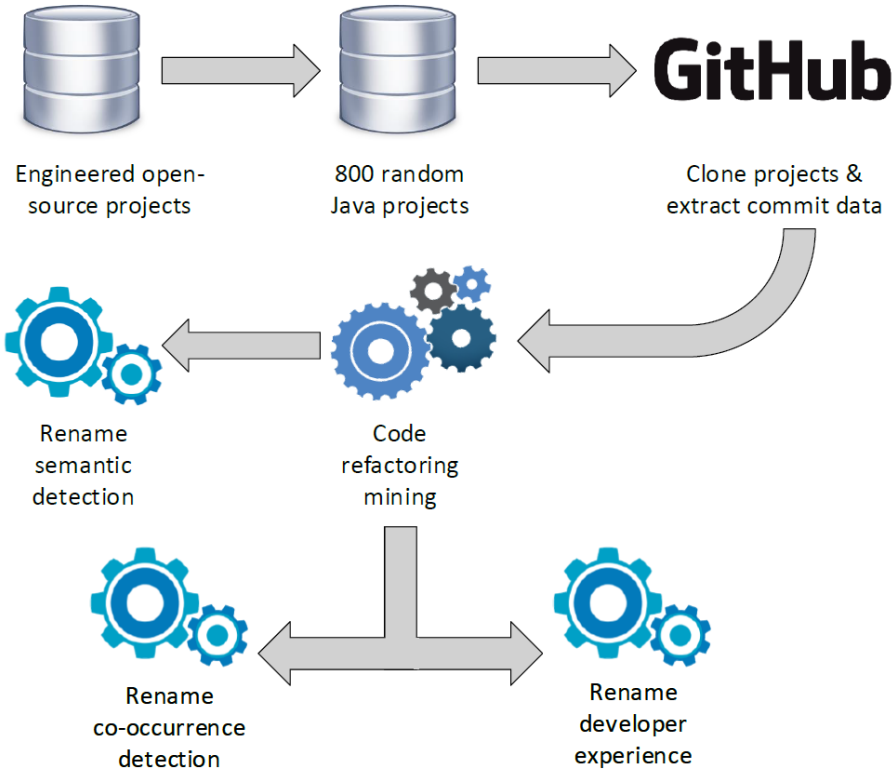
Research Questions

1. What is the distribution of **experience among developers** that apply renames?
    - Helps us understand the types of developers in our dataset that rename identifiers
2. What are the **refactorings that occur more frequently** with identifier renames?
    - Helps us identify relationships between renames and surrounding refactorings
3. To what extent can we use **refactoring occurrence and commit message analysis** to understand why different semantic changes were applied during a rename operation?
    - Pinpoint the reasons that motivated the developers to perform a rename

Semantic Change Categories

V. Arnaoudova, L. M. Eshkevari, M. D. Penta, R. Oliveto, G. Antoniol and Y. Guéhéneuc, "REPENT: Analyzing the Nature of Identifier Renamings," 2014, IEEE TSE

Dataset



748,001
total commits

732
avg. project commits

19
avg. team size

Distribution of the top five refactorings

Refactoring Type	Count	Percentage
Rename Attribute	137,842	19.37%
Rename Variable	84,010	11.81%
Rename Method	82,206	11.55%
Move Class	76,265	10.72%
Extract Method	47,477	6.67%
Others	*283,695*	*39.87%*

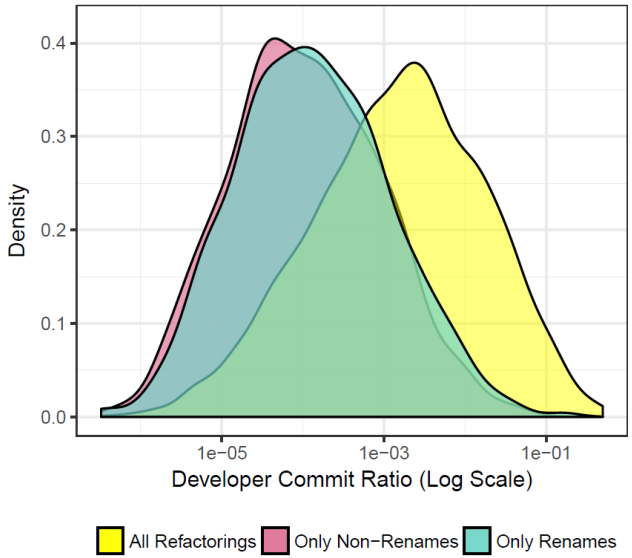
Developer Experience

Approach:

- Utilize a developers project contribution (i.e., # of commits) as a proxy for developer experience within a project

Findings:

- Developers with limited project experience are more inclined to perform only rename refactorings than other types of refactorings
- No difference between the user types with regards to the type of rename complexity



Co-occurrence of Renames With Other Refactorings

Findings:

- Developers frequently perform the rename in isolation (≃ 90% of renames)
- More operations occur before a rename. Common actions performed before a rename:
  - Class: moving the class
  - Attribute: moving the attribute
  - Method: rename or extract
  - Variable: rename operation
- In general, the majority of the refactorings that occur before a rename are related to changes/updates to functionality
- Refactorings occuring after a rename are are associated with some form of code reversal/reverting

Contextualizing refactorings during a rename

Approach:

- LDA and bi/trigrams analysis of commit messages of rename operations that occurred after a refactoring

Findings:

- Renames applied after a refactoring most frequently changes the meaning
  - A narrowing (i.e., specialization) of the name is the most common type of change
  - Preserve meaning was the least occurring semantic type
  - A Rename Variable followed by another Rename Variable tended to add meaning
- Contextualizing yielded in high-level LDA topics
  - E.g., Preserve meaning included the terms ‘fix’, ‘test’ and ‘work’ -- too high-level to determine the problem with the old name
  - ***Key Takeaway***: We need to find optimal techniques to analyze natural language-based project artifacts

Summary

- A study on the refactorings surrounding an identifier rename to derive the developer motivations for the rename
- A large-scale study on 800 engineered open-source Java systems
- Developers who perform only rename refactorings in a project contribute less to a project
- Developers perform renames in isolation
  - However, more renames are performed after a refactoring than before
- Contextualizing renames that occur after a rename yielded in high-level topics
  - Developers do not explicitly connect the semantic update they make to the identifiers name with the task they performed

Thanks!

https://scanl.org/

SCANL

Source Code Analysis And
Natural Language Laboratory