The impact of licenses on software quality

Ben Cotton

All Things Open 2016

Outline

- Motivation
- License paradigms
- Methodology
- Results
- Future Work

Disclaimers

- I lean copyleft
- My employer produces (primarily) proprietary software

Caveats

- This is novel work
- This is unfunded work
- This is slightly-dated work

What this talk is

- A look at how license paradigms might affect the relative quality of projects that are going to be open source anyway

What this talk is not

- A comparison of open source and proprietary software
  - It’s been done
- About the philosophical merits of license paradigms

Motivation

- Open source is big business
  - Red Hat, Inc. generated $2 billion in revenue in 2015
  - Used and developed by companies like Google, Amazon, Apple, Facebook, and Microsoft

Motivation

- Improving quality
  - Reduces costs
  - Increases customer satisfaction
  - Aids reputation
- These are important for *gratis* projects, too

Motivation

- News stories highlight open source software quality, not always in a positive light
  - Heartbleed
  - Shellshock
  - Ghost

Motivation

- Aid selection of software inputs
  - Everybody is downstream of *something*
- License selection for open source projects

Motivation

- GRADUATE!

License paradigms

(Please put your flame throwers away)

License paradigms

- Copyleft
  - Share software under the same terms you received it
  - Considered “open source”, but often associated with the term “free software”
  - Associated with lower developer engagement and high user engagement
  - Considered “viral”

License paradigms

- Permissive
  - Few/no conditions on resharing
  - Considered “open source”
  - Associated with higher developer engagement and lower user engagement
  - Allows downstream users to incorporate into closed projects

Defining quality

- Philip B. Crosby: the degree to which software conforms to user requirements

Defining quality

- Open source projects often lack formal requirements
- Usage numbers are very difficult to collect
- We need to find some reasonable metric

Bug reports?

- No! Bug trackers are the worst
  - Gated
  - Duplicates
  - Not-a-bug?
  - Used as feature trackers
  - Dependent on user base
  - Might not exist

Metric selection

- Direct metrics are preferable
- Static analysis is easier to perform

Technical debt

- Introduced by Ward Cunningham in 1992
- Represents the cost of “not right” code
- Can be quantified
- Can be measured with static analysis

Methodology

- Selected 40 of the most popular projects for each paradigm from SourceForge.com
- Used the SonarQube software analysis tool to evaluate technical debt
- Tested the hypotheses with the Mann-Whitney test
  - Used a significance of 0.10

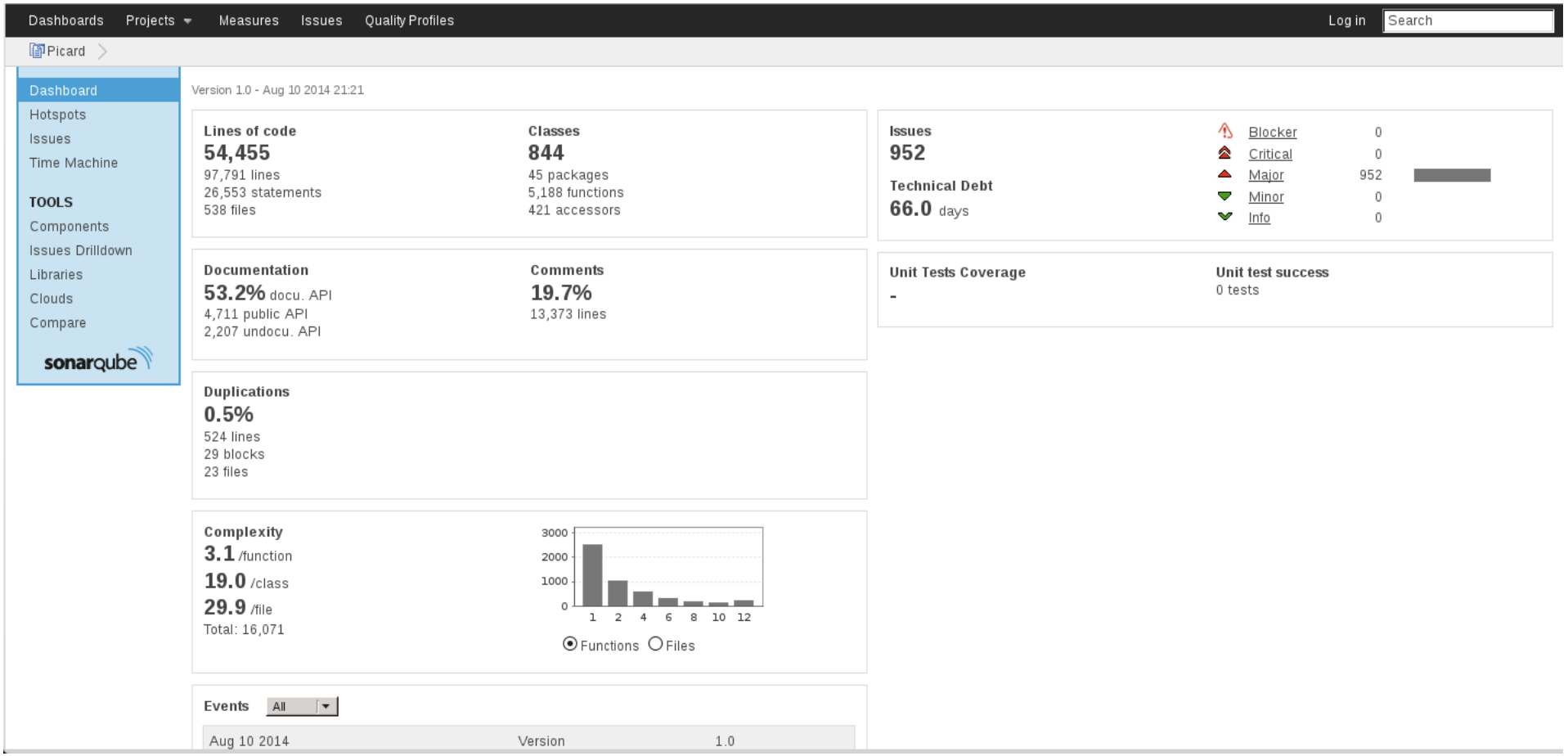
Constraints

- Assumptions
  - Quality variations within a paradigm are negligible
  - Selected projects are representative
  - Difference in quality is meaningful to project leaders
- Limitations
  - Impossible to create a random sample
  - Free SonarQube plugins do not exist for some popular programming languages

Data collection

- Downloaded all software packages
- Configured SonarQube based on language and directory structure
- Queued project for analysis
- Retrieved debt, complexity, and lines of code from SonarQube web interface

SonarQube project info

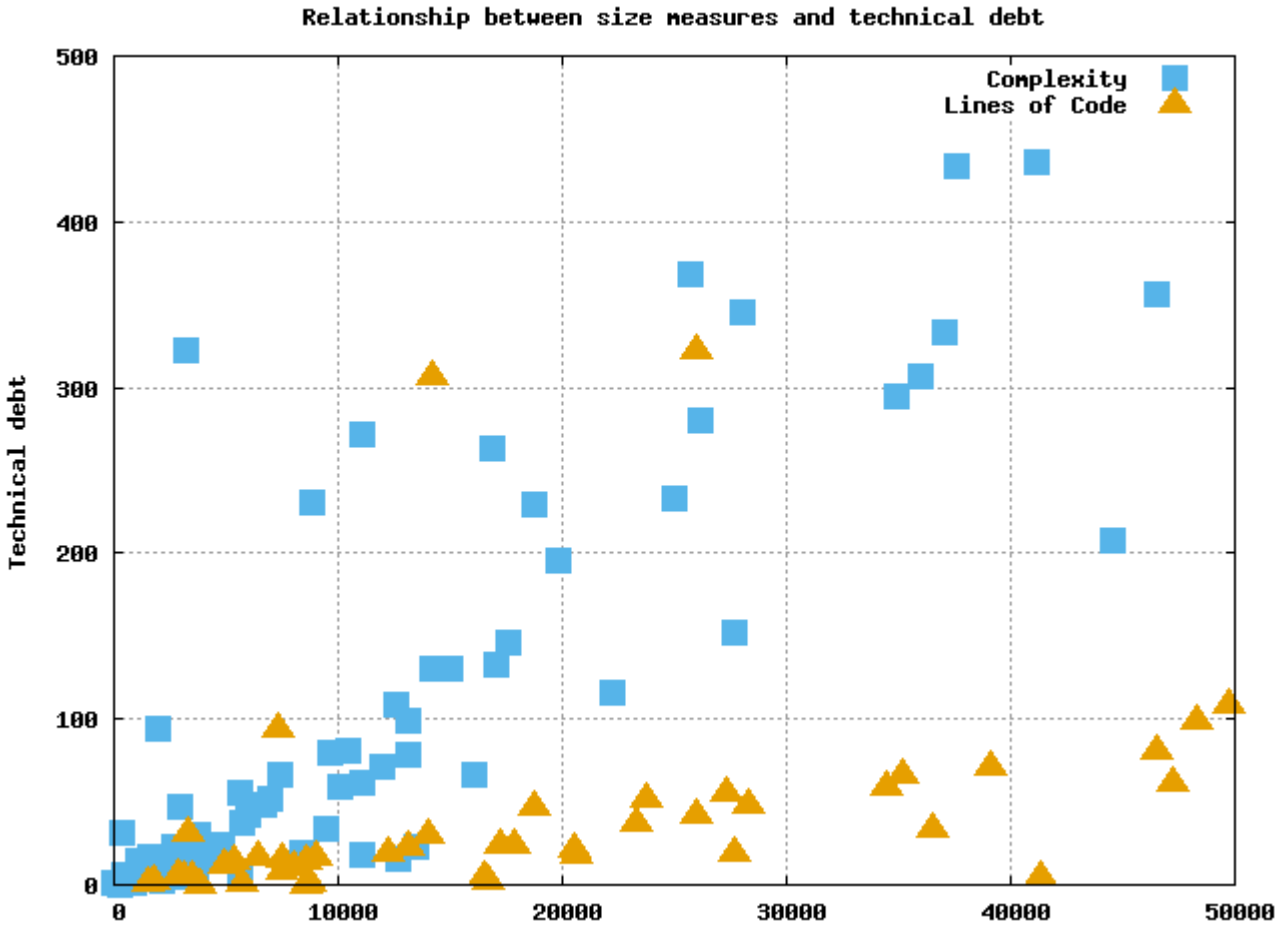


Methodology – test

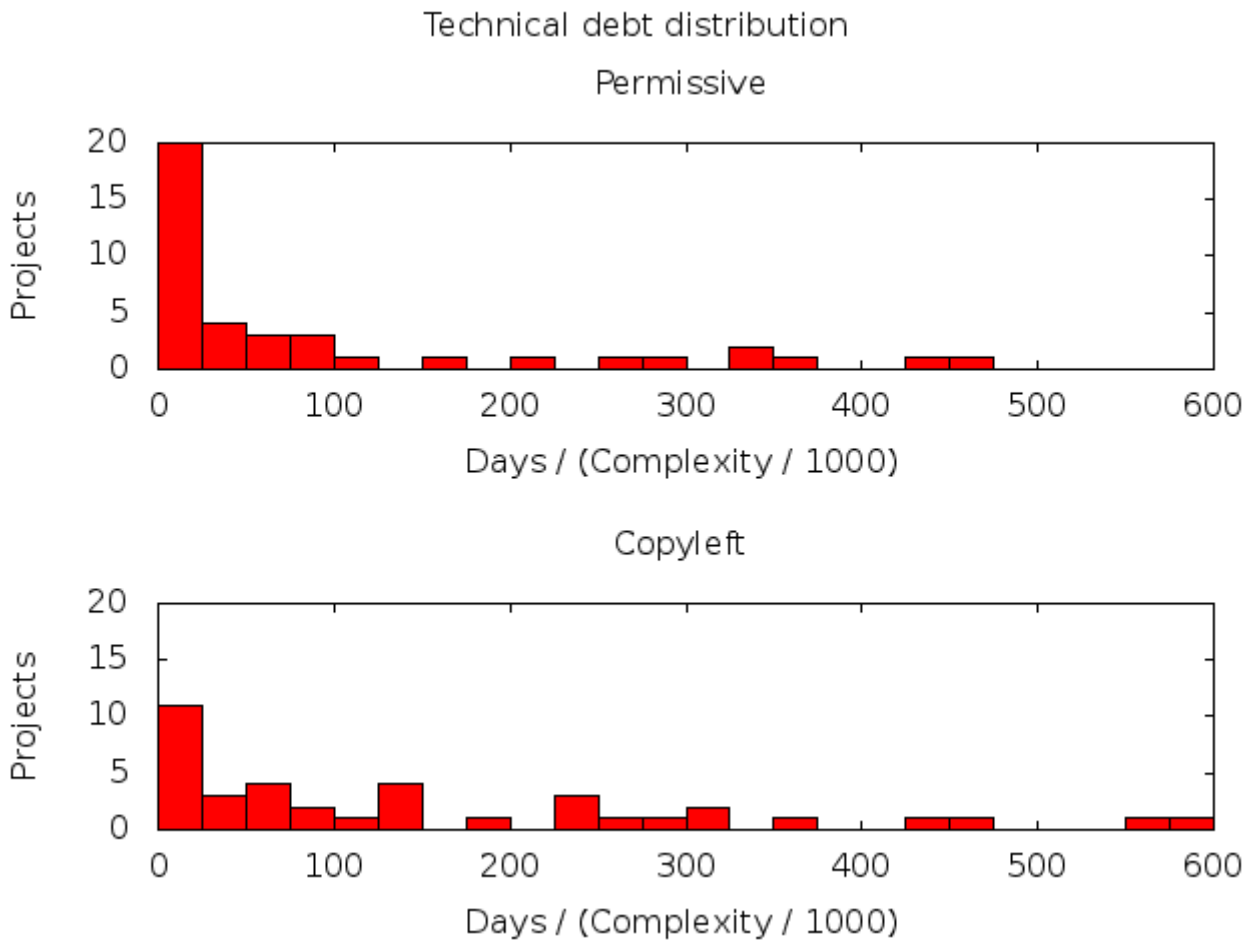
- Mann-Whitney test
  - Compares the sum of ranks
  - More robust against outliers
  - No assumption of equal variance or normally-distributed differences

Results

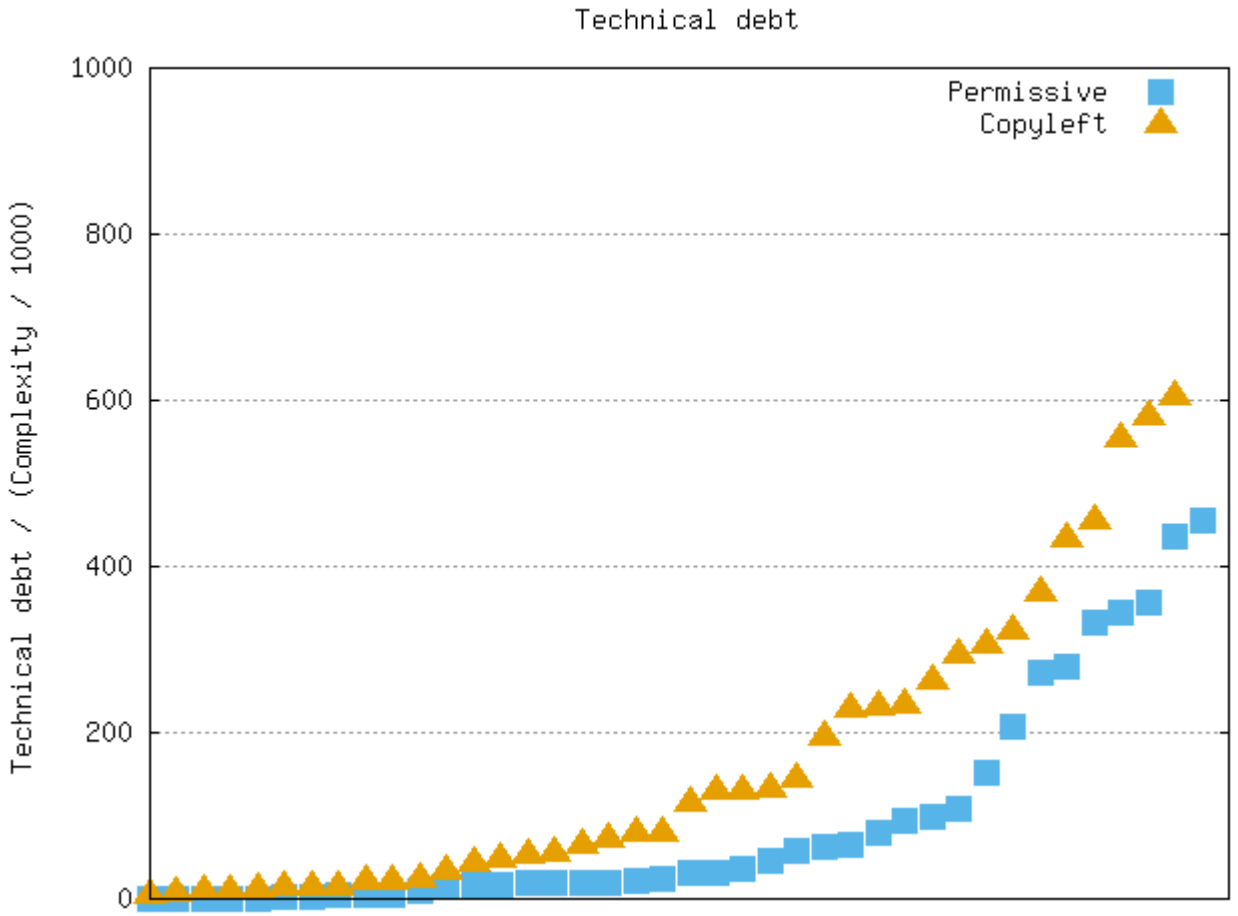
- Need to scale raw measures to project size



Scaled technical debt distribution



Scaled technical debt



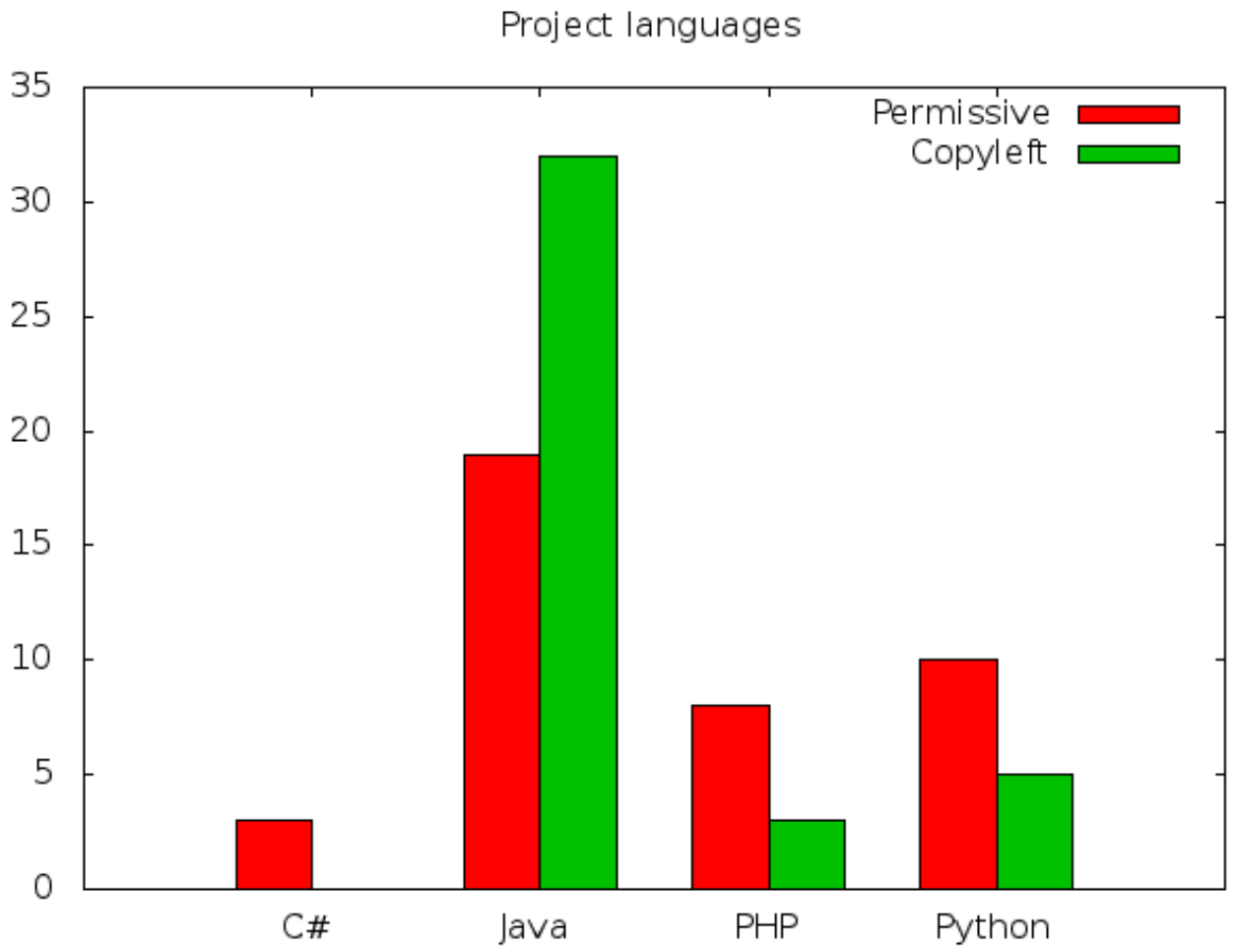
Results

- Mann-Whitney test
  - Medians: 8.64 (copyleft), 7.04 (permissive)
  - U = 531
  - p = .010
- Permissively-licensed software has less technical debt

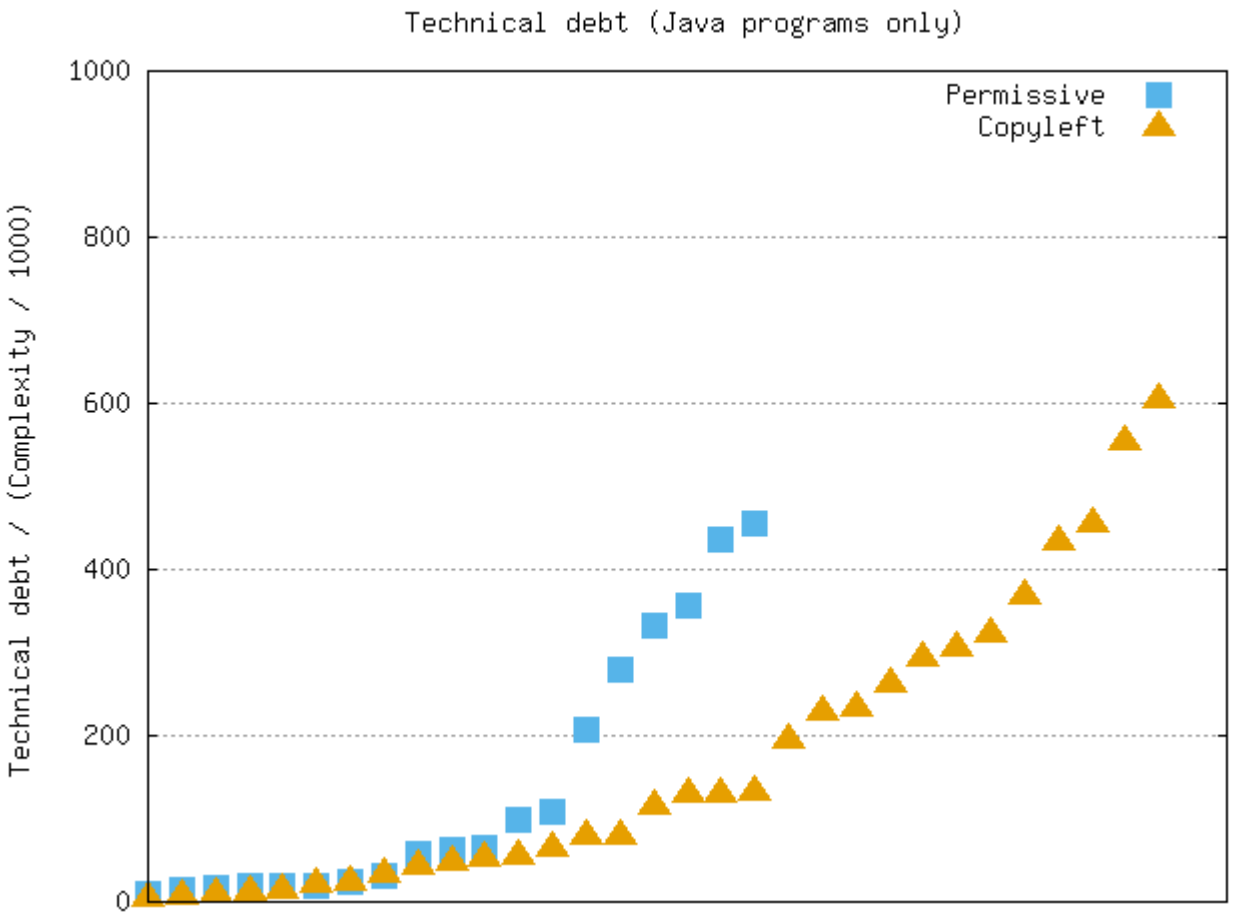
Results

- What about language effects?

Project languages



Java only


Technical debt (Java programs only)
Technical debt / (Complexity / 1000)
1000
800
600
400
200
0
Permissive
Copyleft

Results

- Mann-Whitney test
  - Medians: 8.80 (copyleft), 7.56 (permissive)
  - U = 258
  - p = .370
- Results are inconclusive

Future work (for someone else)

- Include projects written in C/C++
- Use a larger sample size to allow single-language comparisons
- Improve technical debt models to develop meaningful thresholds
- Examine project governance effects

Need more?

- Full thesis: http://bit.ly/ato2016-cotton
- Twitter: @funnelfiasco
- Email: bcotton@funnelfiasco.com

Thanks to my committee:

Prof. Kevin Dittman

Prof. Jeff Whitten,

Prof. Jeff Brewer