Git Repository Health

As measured by the bus factor and other metrics

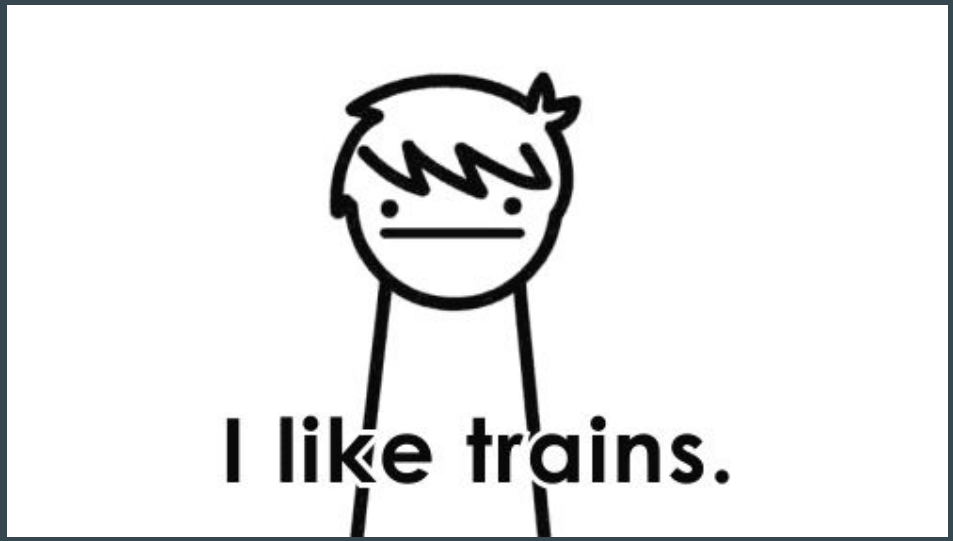
Introduction

Definition

- **Bus Factor** (a.k.a. Truck Factor or Lottery Number):

  > *The minimum number of team members that have to suddenly disappear from a project before the project stalls due to lack of knowledgeable or competent personnel.* (Wikipedia)

- Additional relevance in open-source



Examples

- Open-source project FindBugs:

  > *I'm really sorry to say, but FindBugs project in its current form is dead. . . . It looks like the project leader is not interested in the project anymore, and we can't reach him. . . . We requested his help for the project many times (via direct mails, postings to the list and to the GitHub issues) but haven't received any sign of life from him since a year.*

- Python and its creator Guido van Rossum

Outline

I. Different algorithms to compute the Bus Factor

II. Comparing these approaches

III. Good practices to overcome the problem

Different algorithms to compute the BF

AVL - Avelino et al.

- Calculate and normalize Degree-of-Authorship (DOA) for each file
  - Your commits to a file increase your DOA, others' commits decrease it (mathematical formula)
  - A dev is considered author of a file if their DOA > 0.75
- Algo:
  - Input: Ordered list A of top authors (mapped to their related authored files)
  - F ← getSystemFiles(A)
  - busFactor ← 0
  - While A ≠ ∅ do
    - coverage ← getCoverage(F, A)
    - If coverage < 0.5 then
      - break
    - A ← remove_top_author(A)
    - busFactor ← busFactor + 1
  - return busFactor

CST - Cosentino et al.

- Calculate line/file knowledge using one of following metrics:
  - *Last Change Takes it all*
  - *Multiple Changes Equally Considered*
  - *Non-Consecutive Changes*
  - *Weighted Non-Consecutive Changes*
- Propagate to assign knowledge for directories and finally for the project itself
- Define two sets of developers:
  - Primary devs (P), with minimum knowledge K_p, where p = 1 / (number of contributors to file)
  - Secondary devs (S), with knowledge K_p / 2
- Bus factor = $P \cup S$

RIG - Rigby et al.

- Use *git-blame* command to help define abandoned lines of code and files.
- Simulate many scenarios with random groups of devs leaving the project.
- Increments the size of randomized group after a fixed number of iterations.
- If a simulation results in at least 50% abandoned files, define the bus factor as the number of devs in the simulated group.

Comparing these approaches

Method

Ferreira et al., 2017. *A Comparison of Three Algorithms for Computing Truck Factors*

Gathering a sample	Cleaning the repos	Running the 3 algos	Comparing error
Selecting popular, recent, open-source projects. Surveying the developers via an issue on GitHub. **=> oracle of 35 repos**	Removing third-party code, handle GitHub aliases.	With the recommended parameters.	$error = BF_{algo} - BF_{oracle}$

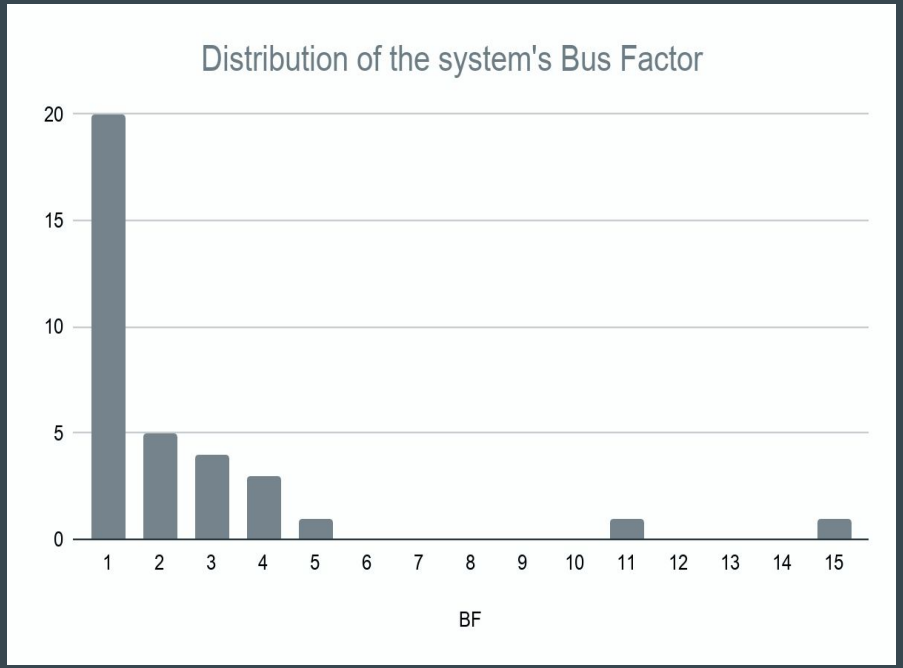
Results

- Accuracy
  - AVL: 71.4% correct estimation
  - CST: 68.6%
  - RIG: 34.3%
- Developers identification: AVL, CST then RIG
- RIG non determinist

- Limits
  - Subjectivity of the concept
  - Files not as equally important
  - No account of last commit

Bus factor of some GitHub repositories

Some known systems...	BF
junit4	4
atom-shell	1
d3	1
RXJava	1
symphony	15
ipython	5
netty	2



65% repo have BF≤2

Avelino et al., among 133 popular projects on GitHub

For fun...

What's the Bus Factor of KTH/devops-course repo ?

Answer: BF = 2

and...

```
TF = 2 (coverage = 30,12%)
TF authors (Developer;Files;Percentage):
Martin Monperrus;1083;88,41
Kartik Mudaliar;850;69,39
```

Why?

Commit cdfc740: *Final Submission for Open Task (#235)*

836 changed files with
199,384 additions and 0 deletions.

Good practices to overcome the problem

Good practices to overcome the problem

- Good documentation
- Active community
- Automatic tests
- Code legibility
- Code comments
- Founding/Paid developers

Conclusion

Conclusion

- Bus factor is a easy to understand, difficult to measure
- High BF == low risk
- Several algorithms have been proposed
- AVL performs best
- 65% repos have BF≤2
- Good coding practices can mitigate the risks of having a low bus factor

Take-home: document your code before you get hit by a bus!

References

Cosentino, V., Izquierdo, J.L.C., Cabot, J., 2015. Assessing the bus factor of Git repositories

Ferreira, M., Valente, M.T., Ferreira, K., 2017. A Comparison of Three Algorithms for Computing Truck Factors

Rigby, P.C., Zhu, Y.C., Donadelli, S.M., Mockus, A., 2016. Quantifying and Mitigating Turnover-Induced Knowledge Loss: Case Studies of Chrome and a Project at Avaya

Avelino, G., Passos, L., Hora, A., Valente, M.T., 2016. A novel approach for estimating Truck Factors