

Software Development in the Next Decade: Beyond the Dichotomy of Open-Source and Corporate Worlds

Jinglei Ren¹ and Hezheng Yin²

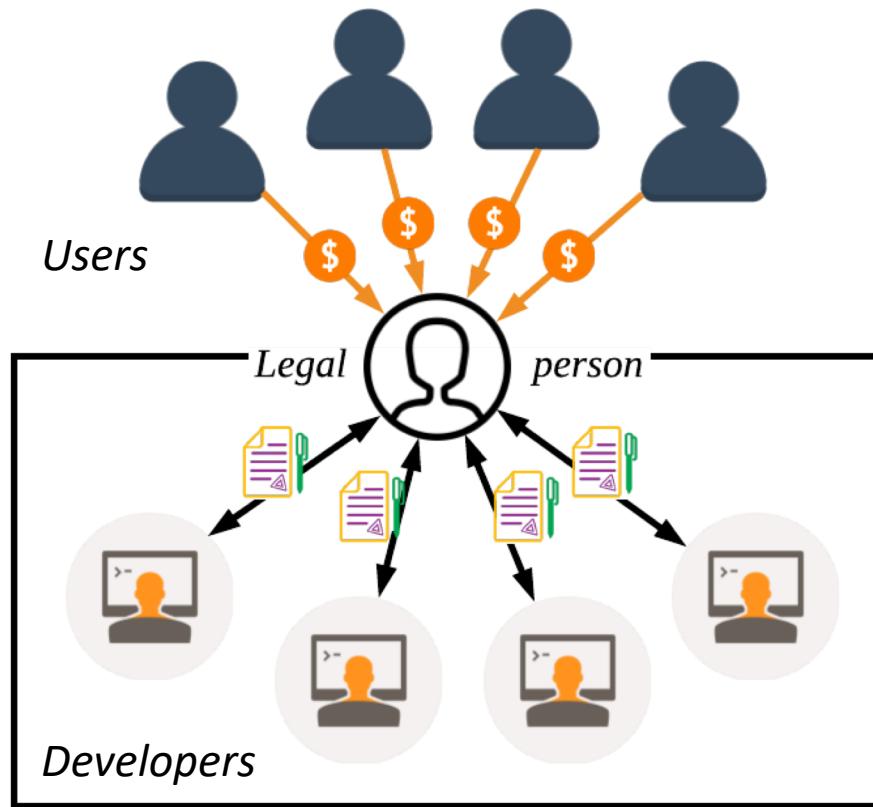
Persper Foundation

¹ Microsoft Research

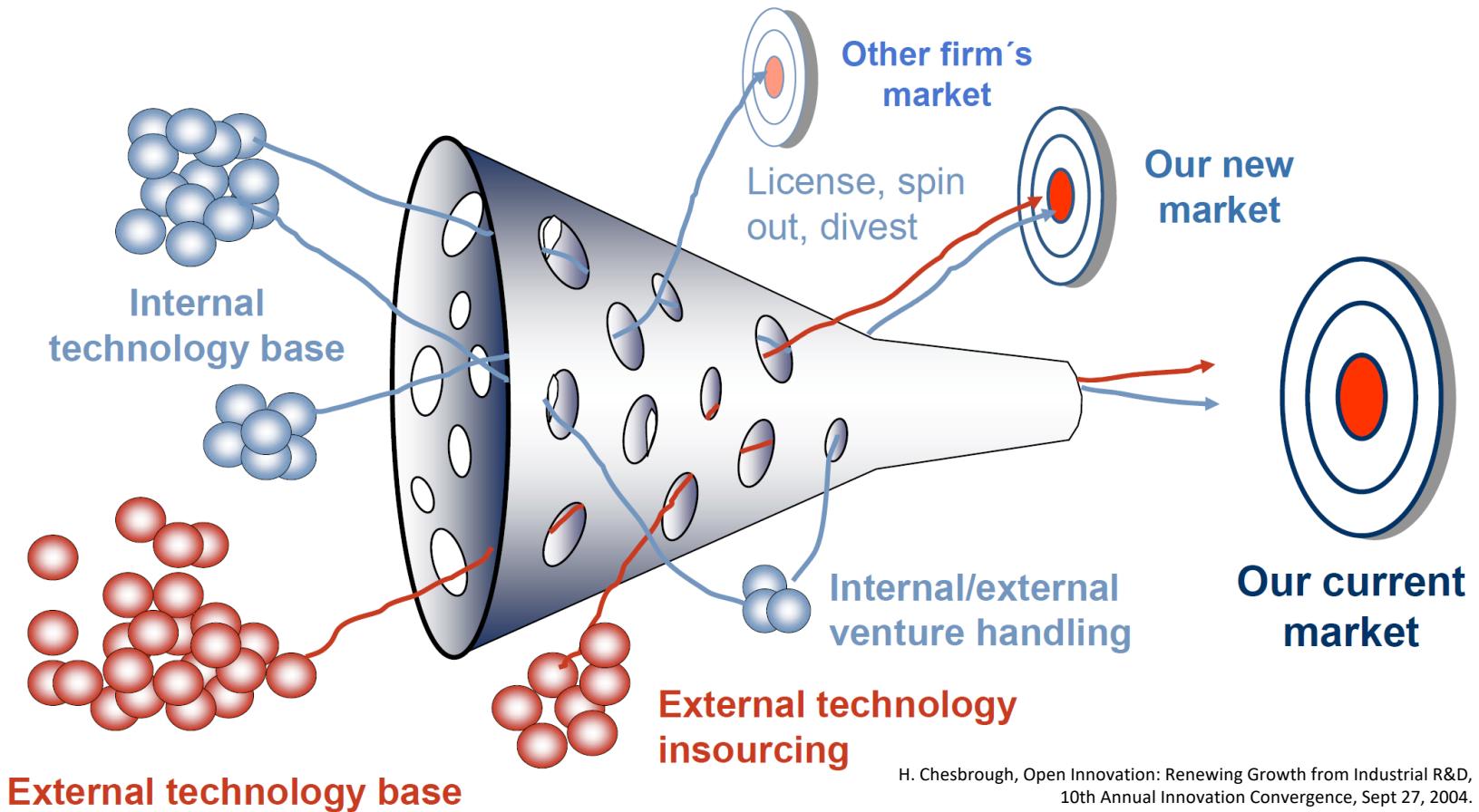
² UC Berkeley

(Advised by Armando Fox from UC Berkeley)

Corporate World vs. Open-Source World



Classic Open Innovation



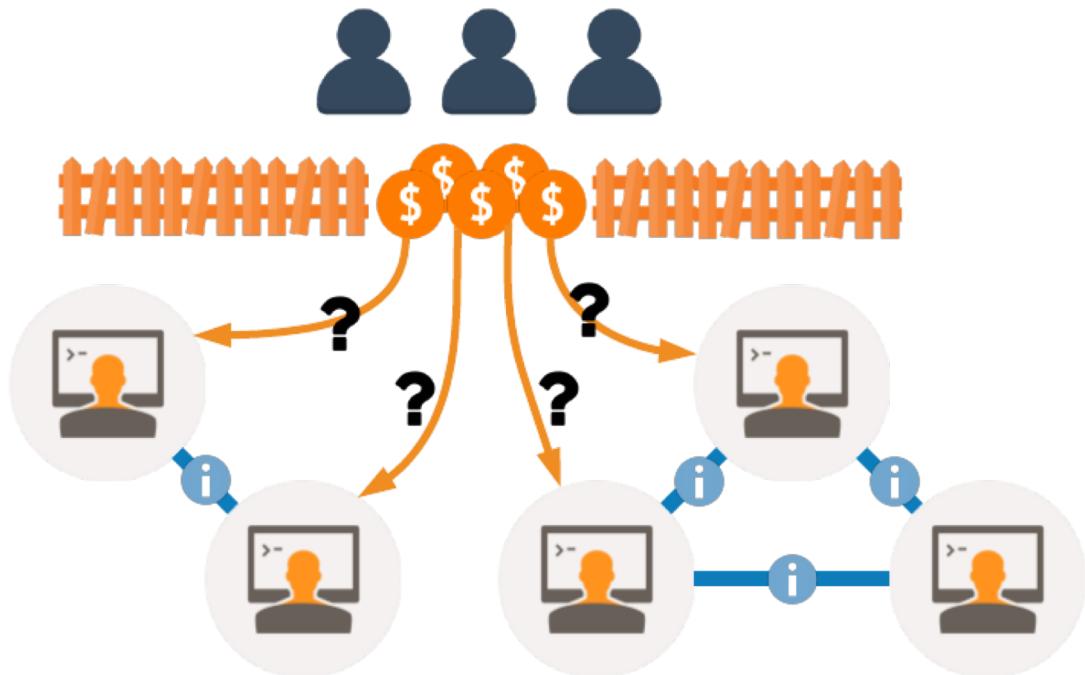
Why the dichotomy of the two worlds?

Overview

- Reasons leading to the two worlds
 - Information asymmetry
 - Transaction cost
- Imperfect solutions
- Technologies with potentials
 - Assessing the value of code contributions → Information asymmetry
 - Decentralized value sharing → Transaction cost
- Virtual shareholding: Beyond the dichotomy
 - Value-shared open source
 - Virtual open corporation
 - Virtual closed corporation

Information Asymmetry

- How to allocate value?
 - Users don't understand developers' work.
 - Individual developers lack a consistent global view.
- Practice in corporation:
Corporate structures
- Practice in open source:
Relinquish the value



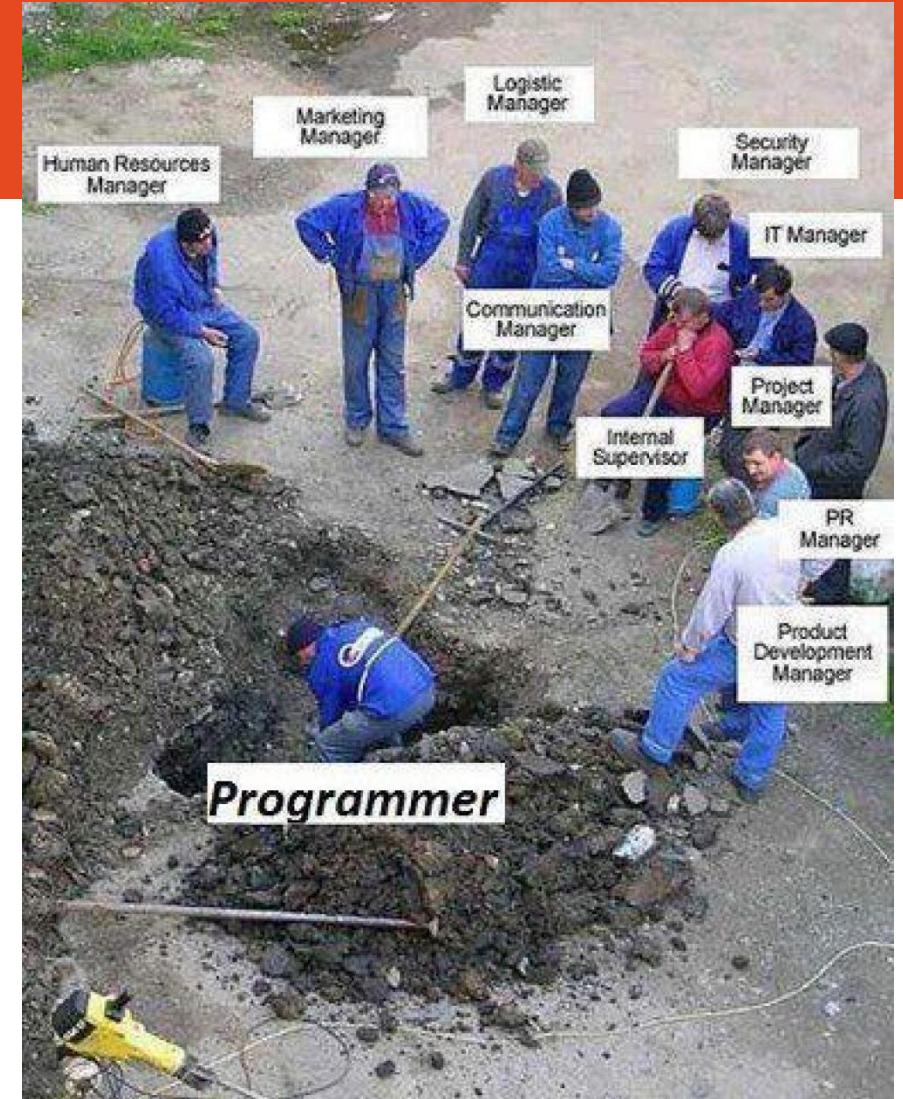
Transaction Cost

- Suppose n developers and m users
- # of contracts $\cong n^2 + nm$
- Dynamics of the network
 - Turnover of developers
 - Churn of users
- Practice in corporation:
Legal person
- Practice in open source:
Open source license



Corporate Issues

- Unfair value allocation
 - Bias on personality and social relationships
 - Why employees leave:
“low salary” (69.44%);
“overworked” (63.12%);
“lack of recognition or reward” (45.24%);
“boss didn’t honor commitments” (43.49%).
- Unnecessary operating overhead
- Your code is *not* yours!



Open-Source Issues

- Financial crisis

“The **worst vulnerability** found... since commercial traffic began to flow on the Internet.”
-- *Joseph Steinberg*,
Forbes.



“Free Can Make You Bleed.”
-- *John Walsh*, SSH Communications Security.

Open-Source Issues

- No general method for value allocation



“... if he took some of the money, then it seemed fair for other contributors ... [to take some].”
-- *Evan You*, creator of vue.js.

Linus' Law:
“given enough eyeballs, all bugs are shallow”

Corporation and open source are compromises.

Assessing the Value of Code Contributions

- Various definitions of value (e.g., market price)
- Development value
 - Development effort embodied in a code contribution
 - Development effort that the code contribution saves
- Abstract development effort in general
 - Assume a hypothetical average developer
 - Similar to the abstract human labor in classic economics
- A coherent concept for building the model

Assessing the Value of Code Contributions

- Basic metric of development effort
 - Commits, additions, and deletions
 - Our metric:
of changed LOCs

The screenshot shows a GitHub commit page for a repository named 'drepo'. The commit is titled 'Adjust functions of Entities' and has four items in its list:

1. Add permission query.
2. Add validity query.
3. Remove required permission check.
4. Support only one owner in construct.

The commit was made by a user named 'basicthinker' 26 days ago. It has 1 parent commit (885ac69) and a commit hash of d17e4c6ef78086fd112e560d31a0d7d9971754df. The interface shows a 'Unified' view of the diff, highlighting 23 additions and 20 deletions in the file 'drepo/contracts/Entities.sol'. The diff shows the following changes:

```
43 43 --- drepo/contracts/Entities.sol
44 44 @@ -13,18 +13,25 @@
45 45     function construct(
46 46         Entity storage entity,
47 47         address[] holders,
48 48         address holder,
49 49         bytes32[] data
50 50     )
51 51 }
```

Assessing the Value of Code Contributions

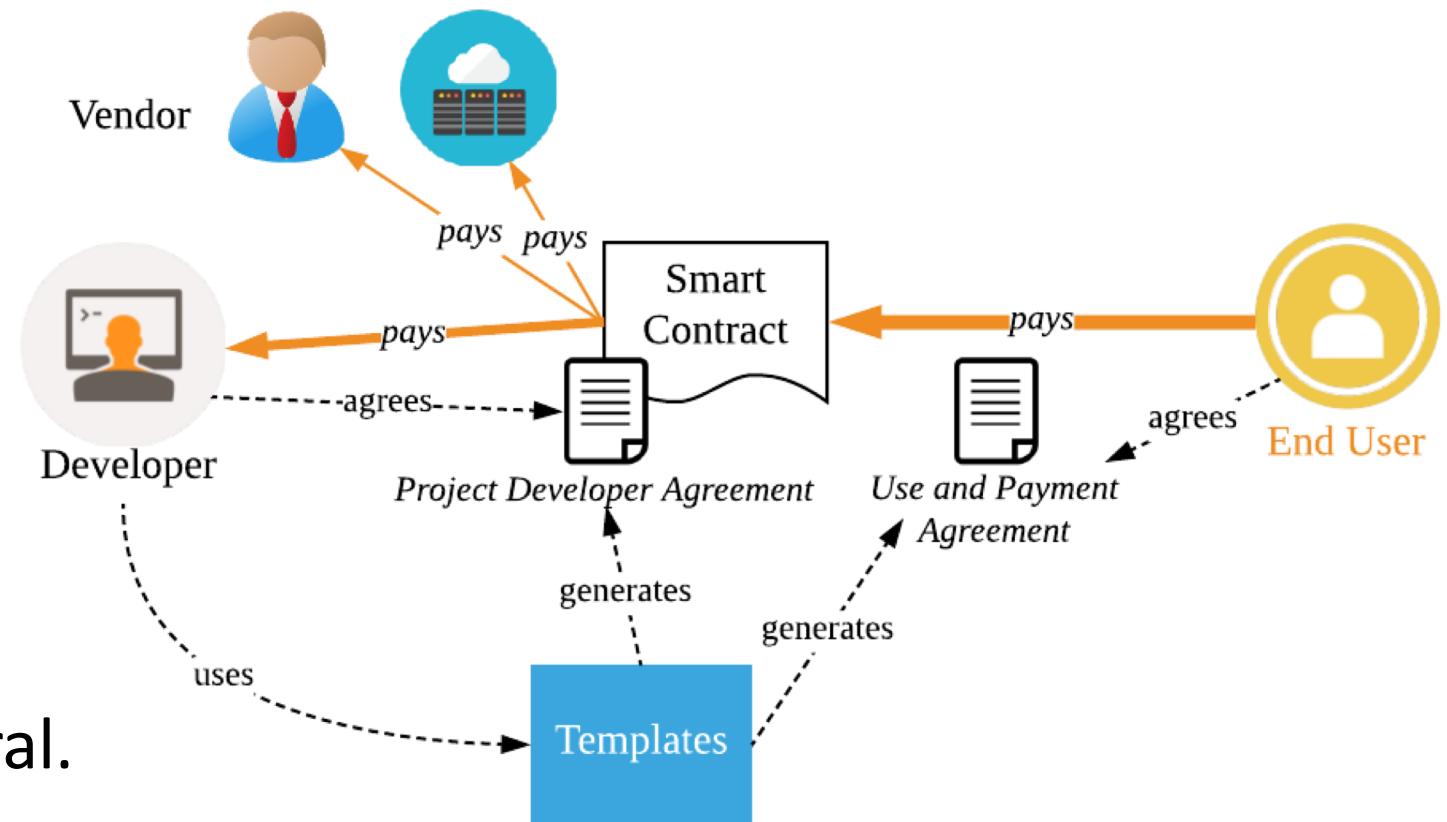
- Commit-specific calibration
 - E.g.: a fix to a very hard-to-track bug
 - A few lines solve a huge issue!
 - Such a commit requires an large effort of an average developer
- Program analysis and natural language processing/machine learning
 - Code structure → Reuse of code
 - Commit messages + large developer surveys → Commit-specific calibration

Decentralized Value Sharing

- A blockchain is an *immutable* ledger of user-defined records
- Smart contracts
 - E.g., “store received tokens in the current account; upon the 1st day of every month, divide the balance by total shares and transfer dividends to shareholders.”
 - Transparent
 - Unstoppable
 - Tamper-proof
- Reduce the cost of trust-building, contracting and enforcement
- Much more expressive than open source licenses

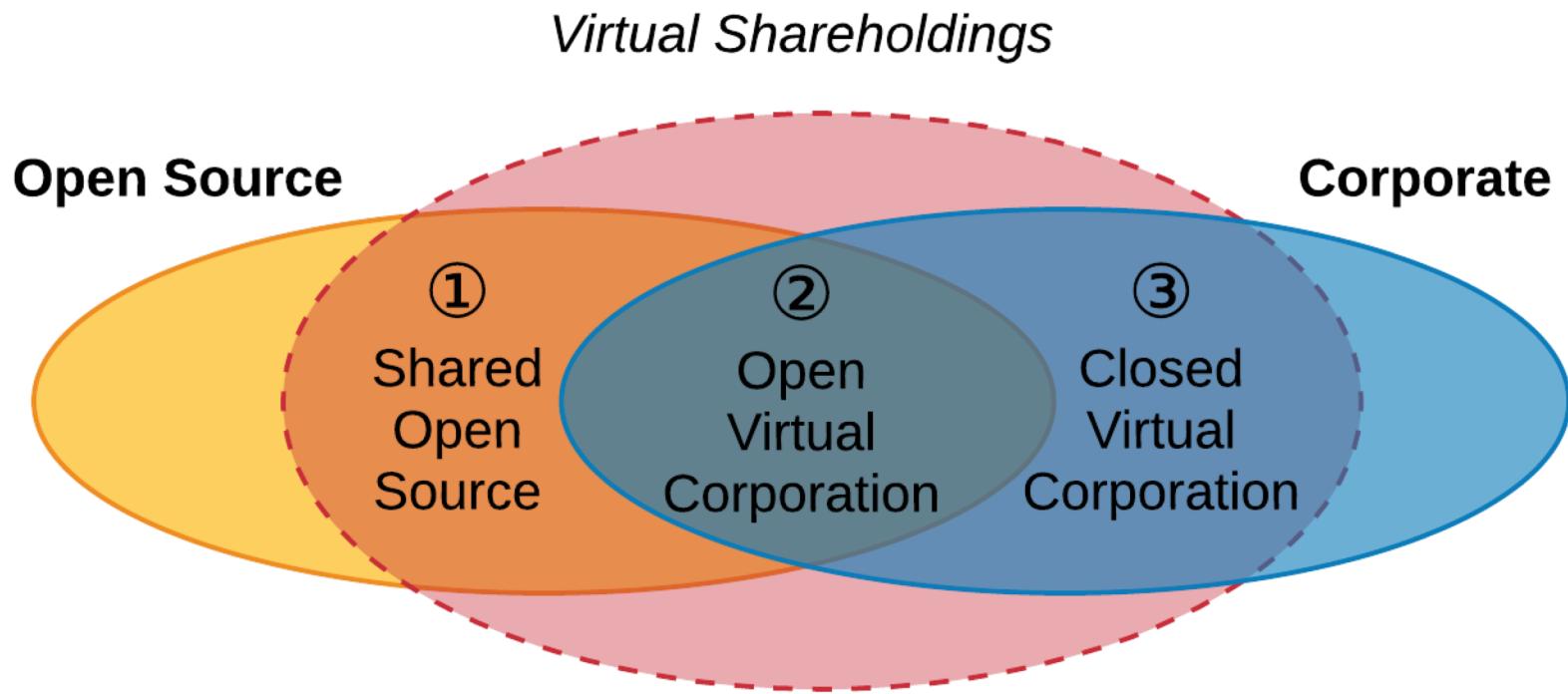
Decentralized Value Sharing

- Framework/workflow



- Value assessment and value sharing are integral.

Virtual Shareholding: Beyond the Dichotomy



Virtual Shareholding: Beyond the Dichotomy

	Traditional open source	Traditional corporation	Value- shared open source	Open virtual corporation	Closed virtual corporation
Financial risk	😢	😊	😢	😊	😊
Fair allocation	😢	😢	😊	😊	😊
Global talent	😊	😢	😊	😊	😢
Long-term return	😢	😢	😊	😊	😊

Outlook: Open Research Questions

- Gaming behavior and counteraction
- Influence on the project governance
- *Real application!*
 - Persper Foundation
 - Sponsor selected open source projects with \$10,000 per month for 6 months.
 - *How would developers react?*

Assessing the Value of Code Contributions

Goal

“... and it becomes a problem how to properly quantify and measure the work that each contributor does.”

-- Evan You, creator of Vue.js

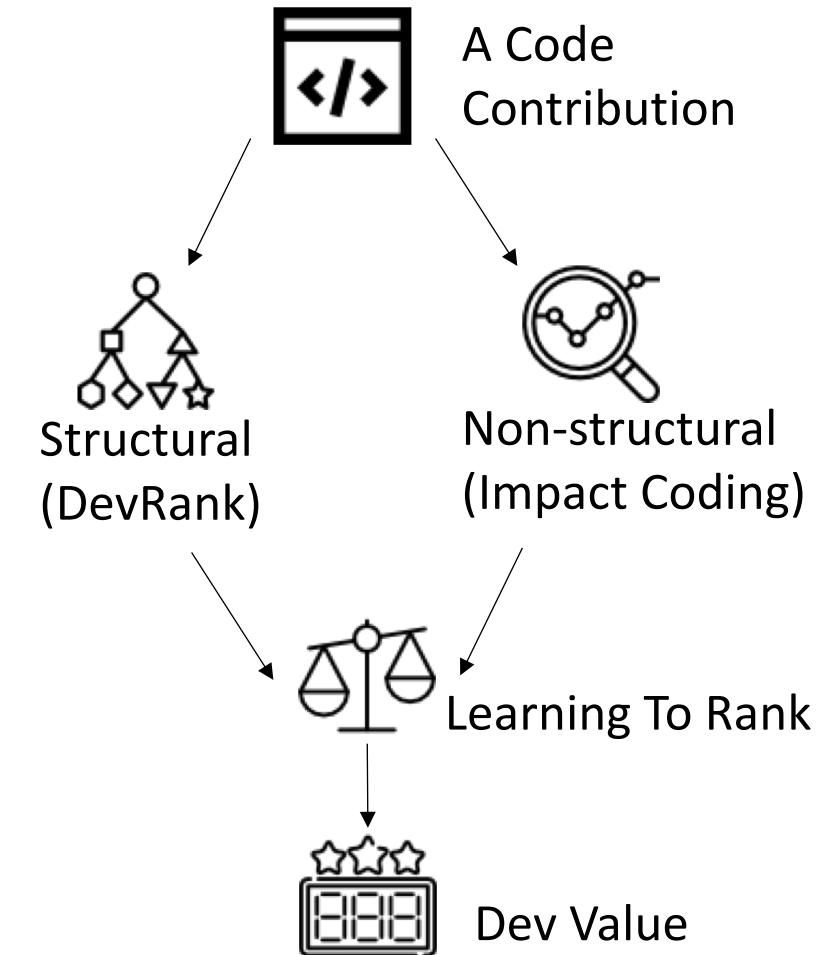
- Goal: Assess the value of code contributions in a way that's
 - Consistent (produce same results for same codebase each time)
 - Efficient (automatable so minimal human effort required)
 - Accurate (**true value -> expert developer's assessment**)

Development Value (Dev Value)

- Observation: Two types of code tend to have high value
 - Addresses a time-consuming dev task
 - Saves a huge amount of development effort for other developers
- Definition: Dev value of a code contribution is the sum of
 - Development effort embodied in itself
 - Development effort that it saves other developers

Dev Value = Structural + Non-structural Value

- Structural value: role of code in codebase structure
- Non-structural value: other impact of code on the team
- Learning To Rank (L2R): combining structural and non-structural value



Measure Structural Value: PageRank

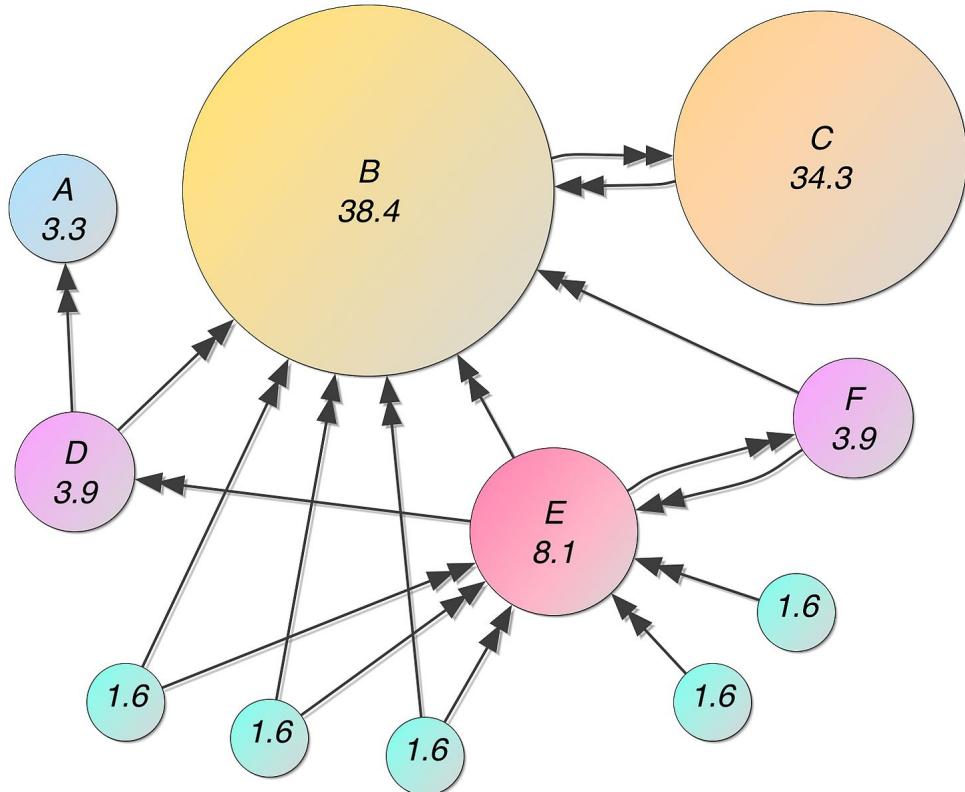


Figure: PageRank for a simple web graph (in %)

- What is PageRank?

- Used by Google to rank websites
- Assigns a numerical score to any given webpage
- Higher score -> more important

- How it works?

- Runs on a directed graph representing the web structure
- All nodes start with equal score
- In each iteration...
- Repeat until reaches a fixed point

Measure Structural Value: Call Graph

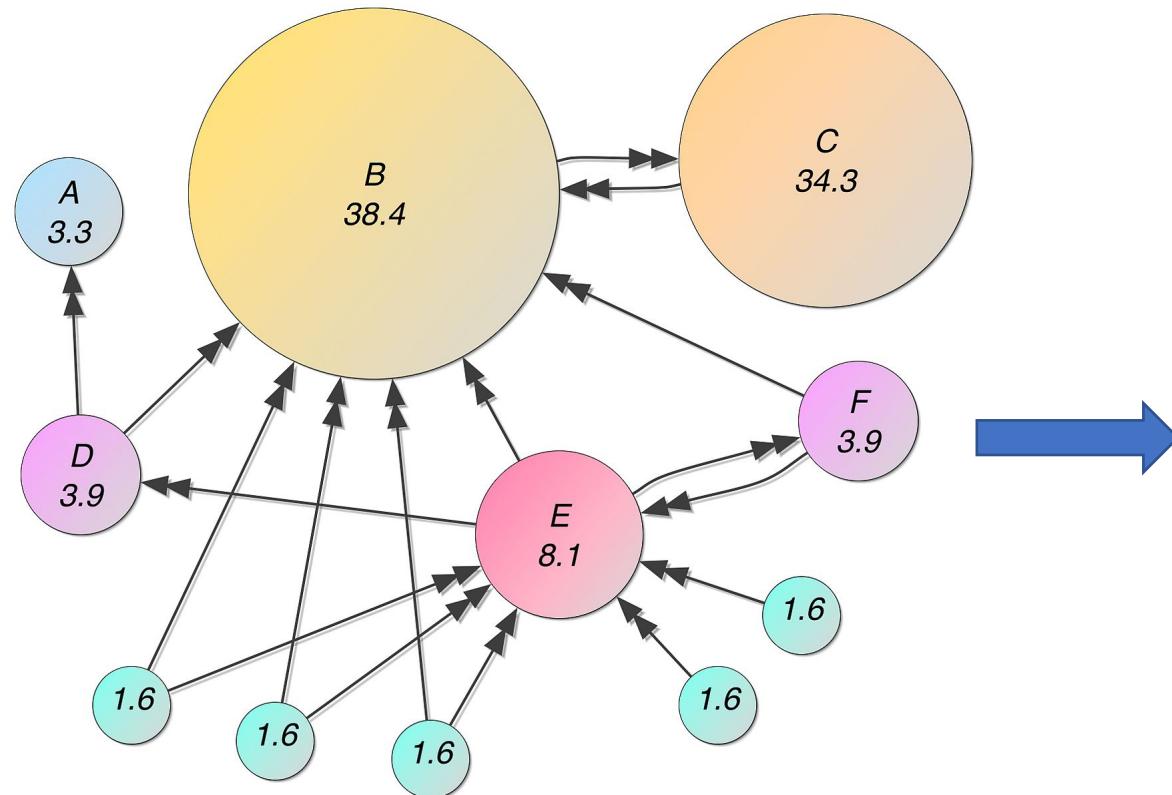


Figure: PageRank for a simple web graph (in %)

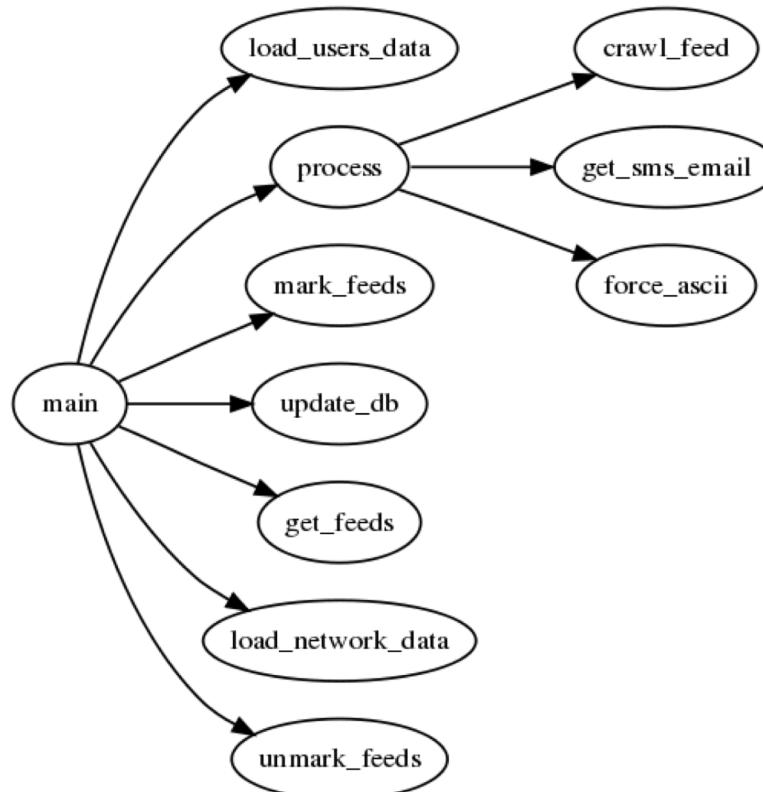


Figure: A simple Python Call Graph

<http://blog.prashanthellina.com/generating-call-graphs-for-understanding-and-refactoring-python-code.html>

Measure Structural Value: DevRank

- As a result, two qualities would contribute to high DevRank value
 - High in-degree
 - High dev effort
- Align well with our definition of development value

	PageRank	DevRank
<i>Node</i>	Website	Function
<i>Edge</i>	Hyperlink	Function call
<i>Weight on edge</i>	Equal weight	Proportional to dev effort

DevRank Results on Linux Kernel Project

	#	Function	File
DevRank	1	slob_free	mm/slob.c
	2	mempool_alloc	mm/mempool.c
	3	dma_pool_free	mm/dmapool.c
	4	kasan_slab_free	mm/kasan/kasan.c
	5	mempool_free	mm/mempool.c
LOC	1	shrink_page_list	mm/vmscan.c
	2	shmemp_getpage_gfp	mm/shmem.c
	3	__vma_adjust	mm/mmap.c
	4	balance_dirty_pages	mm/page-writeback.c
	5	__alloc_pages_slowpath	mm/page_alloc.c

	slob_free (DevRank)	shrink_page_list (LOC counting)
# lines	81	412
# times called	≈ 7,000	2

Note: Calls to slob_free are mostly through kfree.

Table: Most valuable functions under mm directory

Measure Non-structural Value: Impact Coding

- Not all dev value is captured by the code structure
- Interviewed 3 veteran open source developers
 - Author of a popular Twitter client
 - Two FreeBSD developers
- Q: “How would you compare the value of two commits?”
- A: "...classify commits by its impact..."

Ranking	Commit Type
1	Fix for build errors
2	Fix for severe non-build errors
3	Important new features
4	Fix for severe speculative errors
5	Fix for minor errors
6	Regular new features
7	Cosmetic errors
8	Source code hygiene

Table: Hierarchy of commit value, by a FreeBSD developer

Can We Automate Commit Classification?

- Solution: text classification on commit messages
- Classify N = 267,446 JIRA issues into one of four categories
 - Maintenance
 - Feature
 - Improvement
 - Bug
- Trained 3 classifiers on 2 types of inputs (msg title/entire msg)

	Maint.	Feature	Improv.	Bug
# commits	329	1517	14136	28818
BoW-title	0.28	0.34	0.63	0.85
BoW-message	0.2	0.33	0.62	0.85
CNN-title	0.37	0.39	0.65	0.86
CNN-message	0.46	0.39	0.64	0.87
RNN-title	0.4	0.35	0.67	0.86
RNN-message	0.33	0.36	0.68	0.87

Table: Performance (F1 score) of three NLP + ML models for JIRA experiment

Combine Structural and Non-structural Value

- Notations
 - d: devrank value
 - t: impact type of a commit
 - v: overall development value
- Our approach: Learn weight vector w from data provided by expert developers
- Ask many developers to compare random pairs of commits, identifying which in each pair is more valuable
- Use Learning to Rank to learn from this “pairwise ground truth”

$$\varphi(d, t) = \mathbf{w}^T \begin{bmatrix} d \\ t \end{bmatrix}$$

Results: Commit Value Comparison Task

- Our dataset
 - 772 commit pair comparisons
 - Contributed by 35 developers from FreeBSD & Linux Kernel

Feature	Model	Accuracy
LOC	-	53.2%
DevRank	-	59.0%
Coding	Ranking SVM	69.1%
Coding + LOC	Ranking SVM	70.2%
DevRank + Coding	Ranking SVM	73.1%

Conclusion

- Assessing the value of code contributions is difficult for developers
 - Inherent subjectivity
 - Few developers have a wide enough view of the entire project
- We postulated that a given code contribution has both
 - Structural value
 - Non-structural value
- We proposed a combination of
 - A PageRank-inspired algorithm
 - An impact coding scheme

Thank You!



<https://twitter.com/dPersper>



<https://medium.com/persper>



<https://fb.com/persper.foundation>

Recent articles

[What We Talk about When We Talk about a Decentralized GitHub](#)

[Light on the Dark Side of Network Effects](#)

Backup

- Non-code contributions

“The most important feature of Linux, however, was not *technical* but *sociological*.”

—*Eric S. Raymond*, The Cathedral & the Bazaar, 2001.

Backup

- Other incentives than money

