Research and Startup

Prof. Dr. Dirk Riehle

Friedrich-Alexander University Erlangen-Nürnberg

COSS D02

Licensed under CC BY 4.0 International

Agenda

1. Research vs. startup
2. Project management
3. Product management
4. Lead programming
5. Open source governance
6. Open source strategy
7. Setting-up shop
8. Getting contributions

© 2020 Dirk Riehle - Some Rights Reserved

1. Research vs. Startup

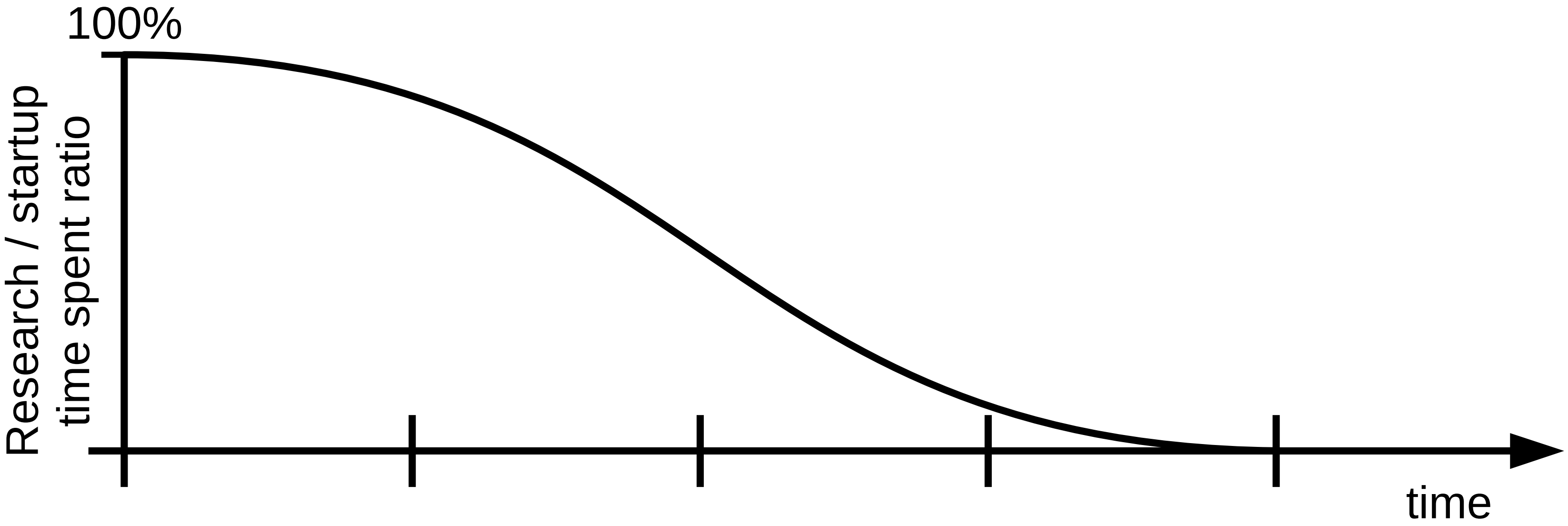
Goal Conflict

- Research
  - Desired output
    - Research papers
    - A dissertation

- Startup
  - Desired output
    - Software product
    - A startup

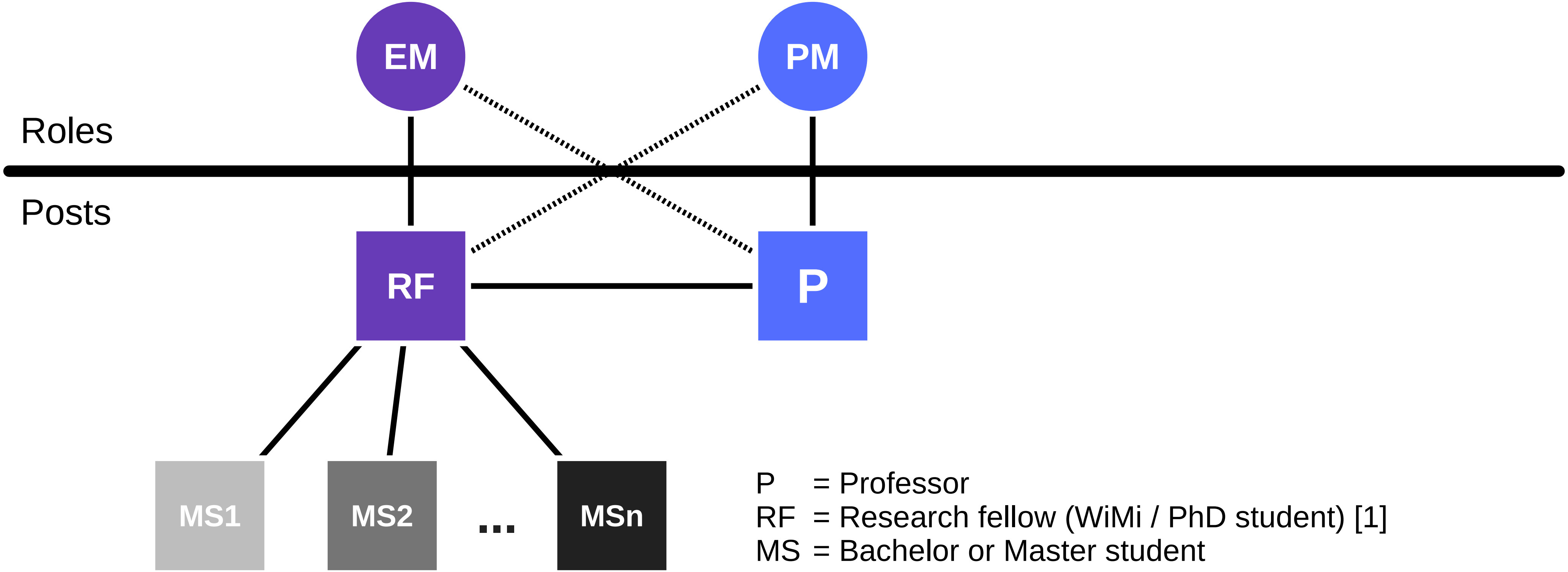
© 2020 Dirk Riehle - Some Rights Reserved

Resulting Time Allocation



2. Project Management

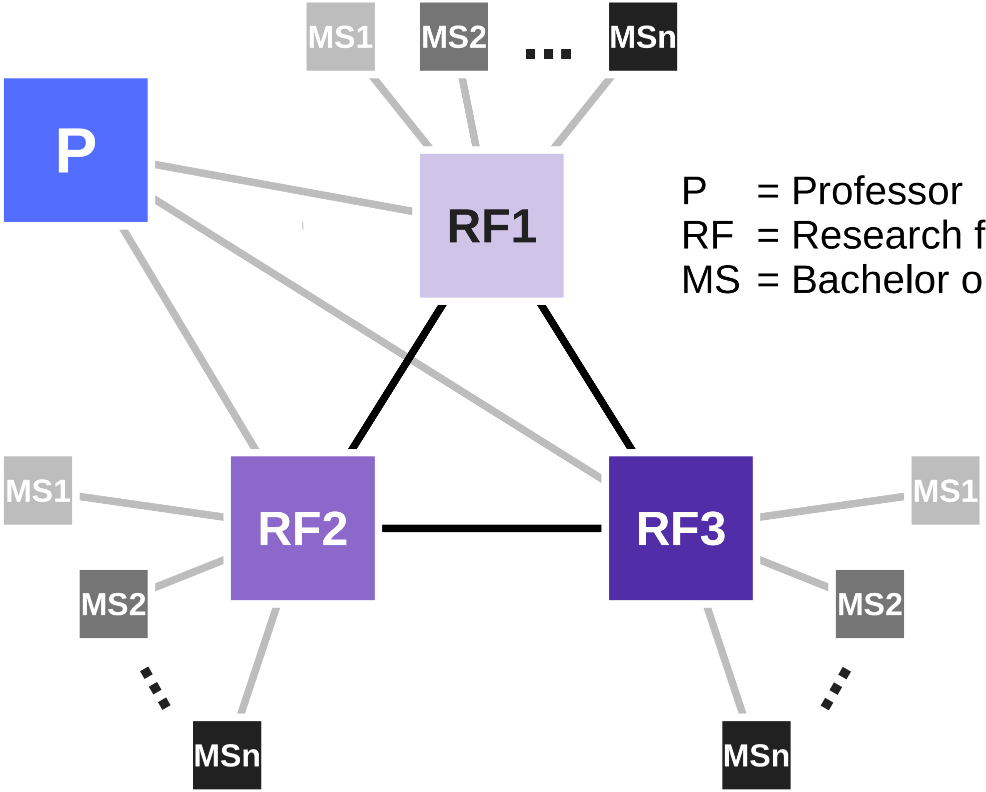
The Solo Founder Model



P = Professor
RF = Research fellow (WiMi / PhD student) [1]
MS = Bachelor or Master student

[1] The research fellow / PhD students acts as “lead programmer”

The Peer Group Model



P = Professor
RF = Research fellow (WiMi / PhD student)
MS = Bachelor or Master student

© 2020 Dirk Riehle - Some Rights Reserved

End of Project

- Research project finished
- Startup created (or failed)

© 2020 Dirk Riehle - Some Rights Reserved

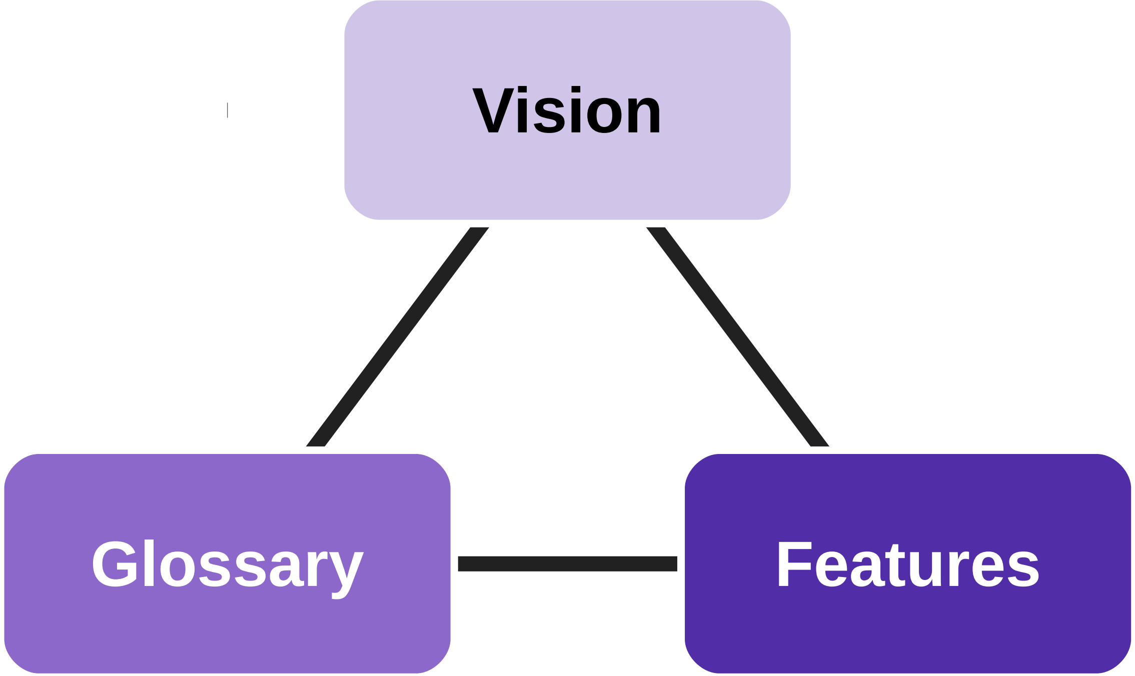
3. Product Management

Who is the Product Manager?

- Research fellow (German: WiMi)
- Professor

© 2020 Dirk Riehle - Some Rights Reserved

Product Vision, Glossary, and Features



© 2020 Dirk Riehle - Some Rights Reserved

Product Releases

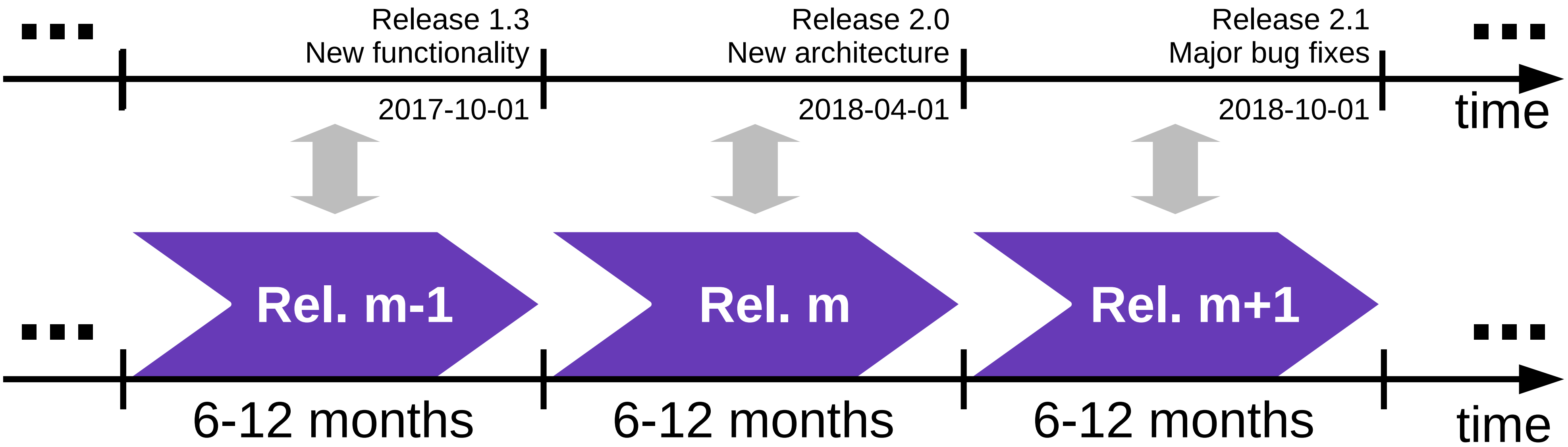
- Product release
  - A deployable product increment
    - That provides new significant business value to market / customer
- Iteration release
  - A demonstratable product increment
    - That serves to gather relevant feedback from market / customer

© 2020 Dirk Riehle - Some Rights Reserved

Product Roadmap

- External roadmap
  - Overview-oriented; few details are provided
  - Serves communication needs
  - Serves external stakeholders
- Internal roadmap
  - As detail oriented as needed (richer than external roadmap)
  - Serves planning and communication needs
  - Serves internal stakeholders

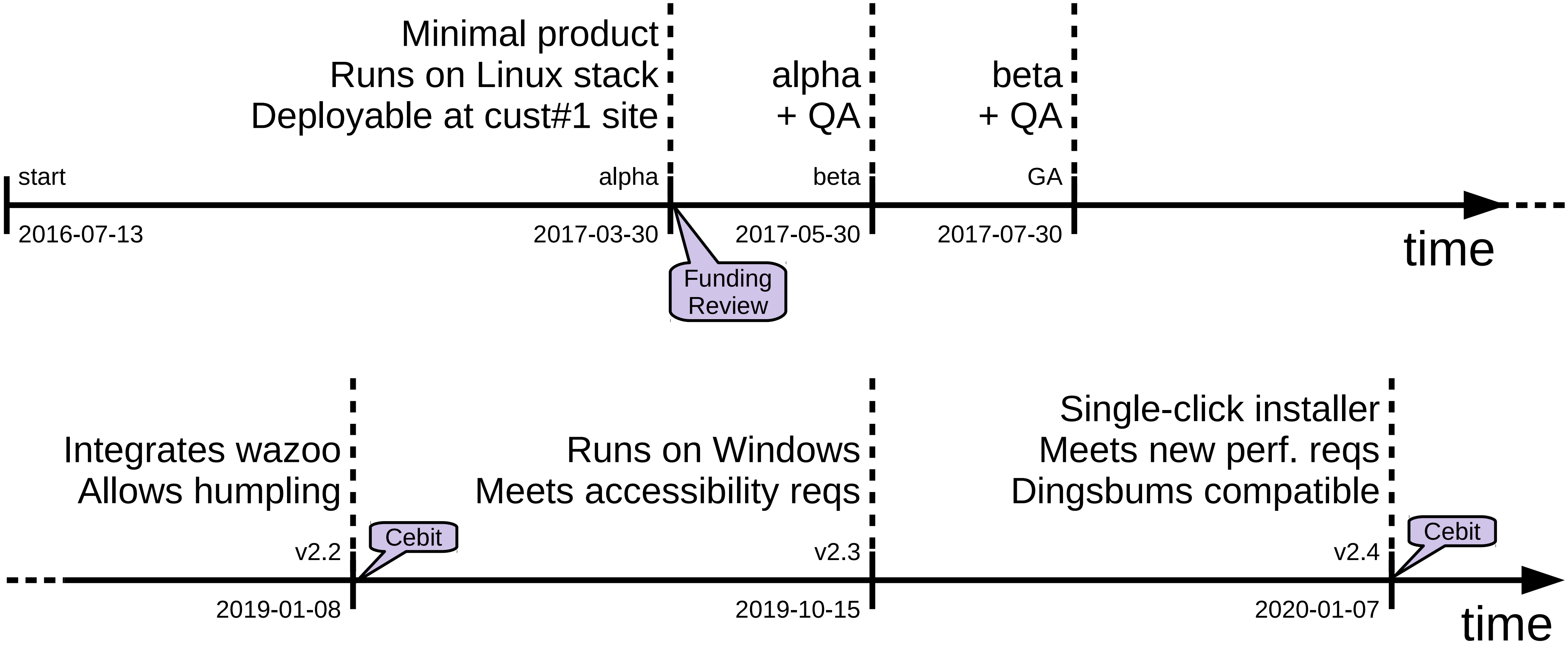
Roadmap as Sequence of Releases [1]



[1] The release rhythm depends on the software’s position in the tech. stack

© 2020 Dirk Riehle - Some Rights Reserved

Example Product Roadmap



Minimal product
Runs on Linux stack
Deployable at cust#1 site

alpha
+ QA

beta
+ QA

start
2016-07-13

alpha
2017-03-30

Funding Review

beta
2017-05-30

GA
2017-07-30

time

Integrates wazoo
Allows humpling

Runs on Windows
Meets accessibility reqs

Single-click installer
Meets new perf. reqs
Dingsbums compatible

v2.2
2019-01-08

Cebit

v2.3
2019-10-15

v2.4
2020-01-07

Cebit

time

Product Vision, Roadmap, and Release

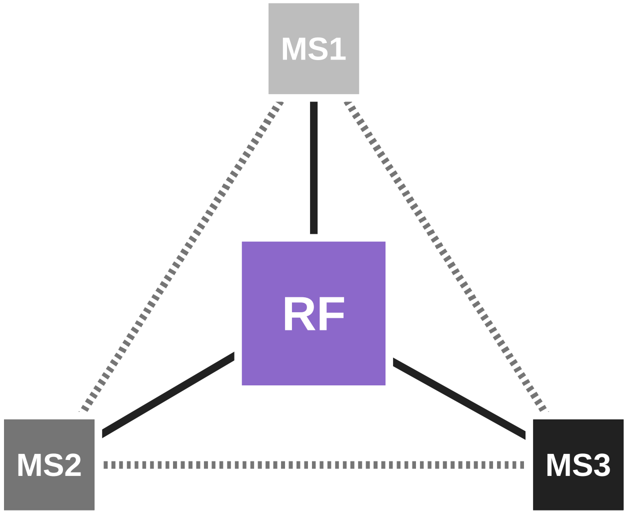
	Timeframe	Content	Certainty	(Responsible) Owner
Vision	Long-term (3+ years)	High-level ideas	Low	CEO and / or strategic product manager
Roadmap	Medium (1-5 years)	Themes and epics	Medium	Strategic product manager
Release	Short-term (months)	Epics and features	High	Technical product manager

© 2020 Dirk Riehle - Some Rights Reserved

4. Lead Programming

Lead Programmer [1]

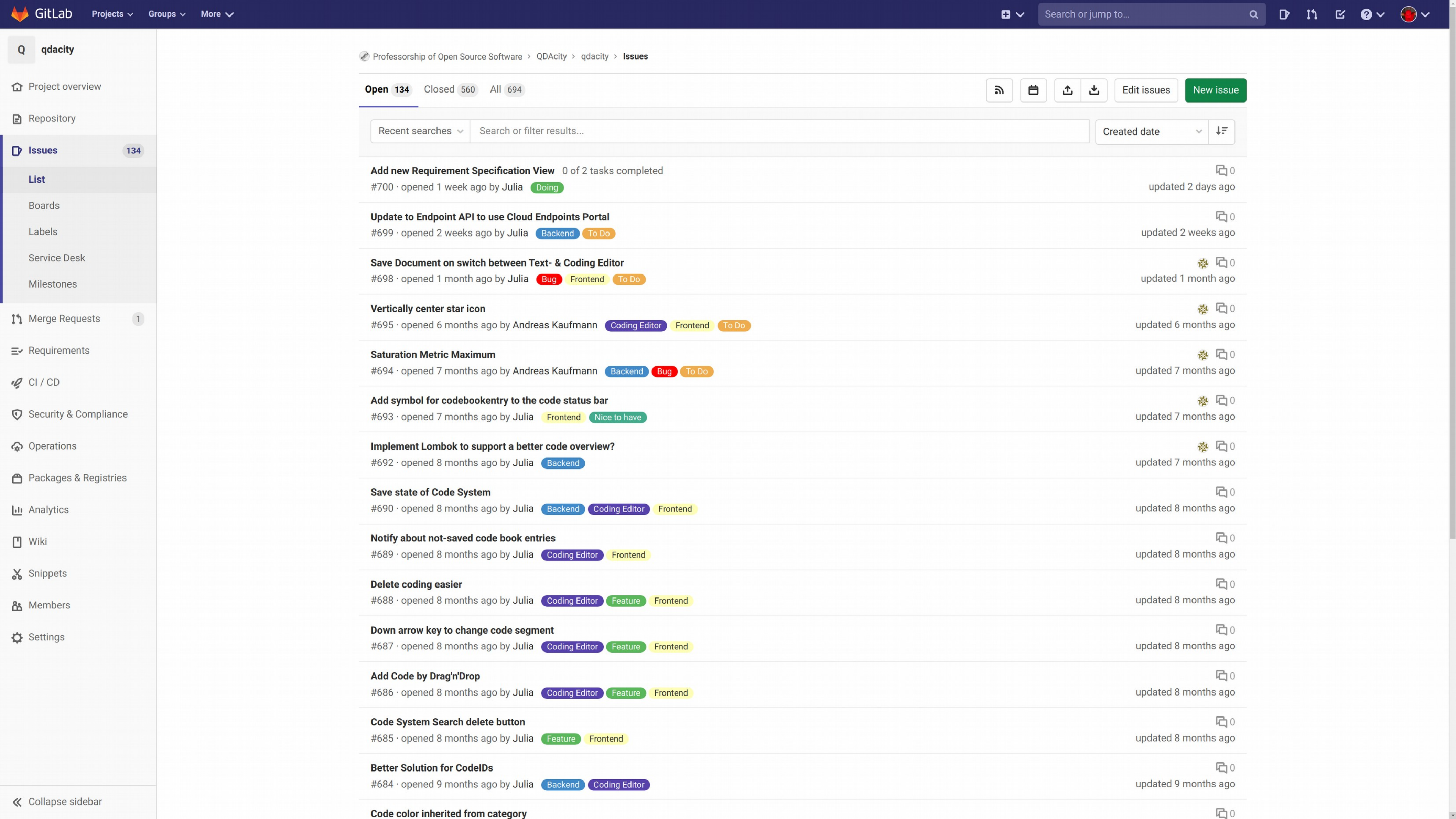
- A lead programmer
  - Is an engineer who leads the development of a component / system
    - In case of solo founder: The system
    - In case of peer group: A key component
  - W.r.t. the component, is responsible for maintaining
    - Product vision and getting the features done
    - Architectural integrity and code quality
  - Manages other people as they contribute to the component
    - In the beginning, they write most of the code themselves
    - Later, they review and integrate code more than they write
- Other / helper engineers are
  - Students at the university (e.g. Master thesis students)
  - Open source volunteers



[1] A pun on Mills' "chief programmer teams"

Lead Programmer / Helper Engineer Collaboration

- Lead programmer provides feature
  - If necessary breaks it down into tasks
  - Assigns tasks of waits for volunteers
- With helper, discusses design and implementation
  - Helps break down work into tasks / increments
  - Ideally, discussion is public (in open source anyway)
- In multiple increments, reviews and merges code
  - Helper provides sufficiently small commit
  - Lead programmer reviews, comments, integrates
- Until feature is fully implemented



Add new Requirement Specification View 0 of 2 tasks completed
#700 · opened 1 week ago by Julia Doing — 0 — updated 2 days ago

Update to Endpoint API to use Cloud Endpoints Portal
#699 · opened 2 weeks ago by Julia Backend To Do — 0 — updated 2 weeks ago

Save Document on switch between Text- & Coding Editor
#698 · opened 1 month ago by Julia Bug Frontend To Do — 0 — updated 1 month ago

Vertically center star icon
#695 · opened 6 months ago by Andreas Kaufmann Coding Editor Frontend To Do — 0 — updated 6 months ago

Saturation Metric Maximum
#694 · opened 7 months ago by Andreas Kaufmann Backend Bug To Do — 0 — updated 7 months ago

Add symbol for codebookentry to the code status bar
#693 · opened 7 months ago by Julia Frontend Nice to have — 0 — updated 7 months ago

Implement Lombok to support a better code overview?
#692 · opened 8 months ago by Julia Backend — 0 — updated 7 months ago

Save state of Code System
#690 · opened 8 months ago by Julia Backend Coding Editor Frontend — 0 — updated 8 months ago

Notify about not-saved code book entries
#689 · opened 8 months ago by Julia Coding Editor Frontend — 0 — updated 8 months ago

Delete coding easier
#688 · opened 8 months ago by Julia Coding Editor Feature Frontend — 0 — updated 8 months ago

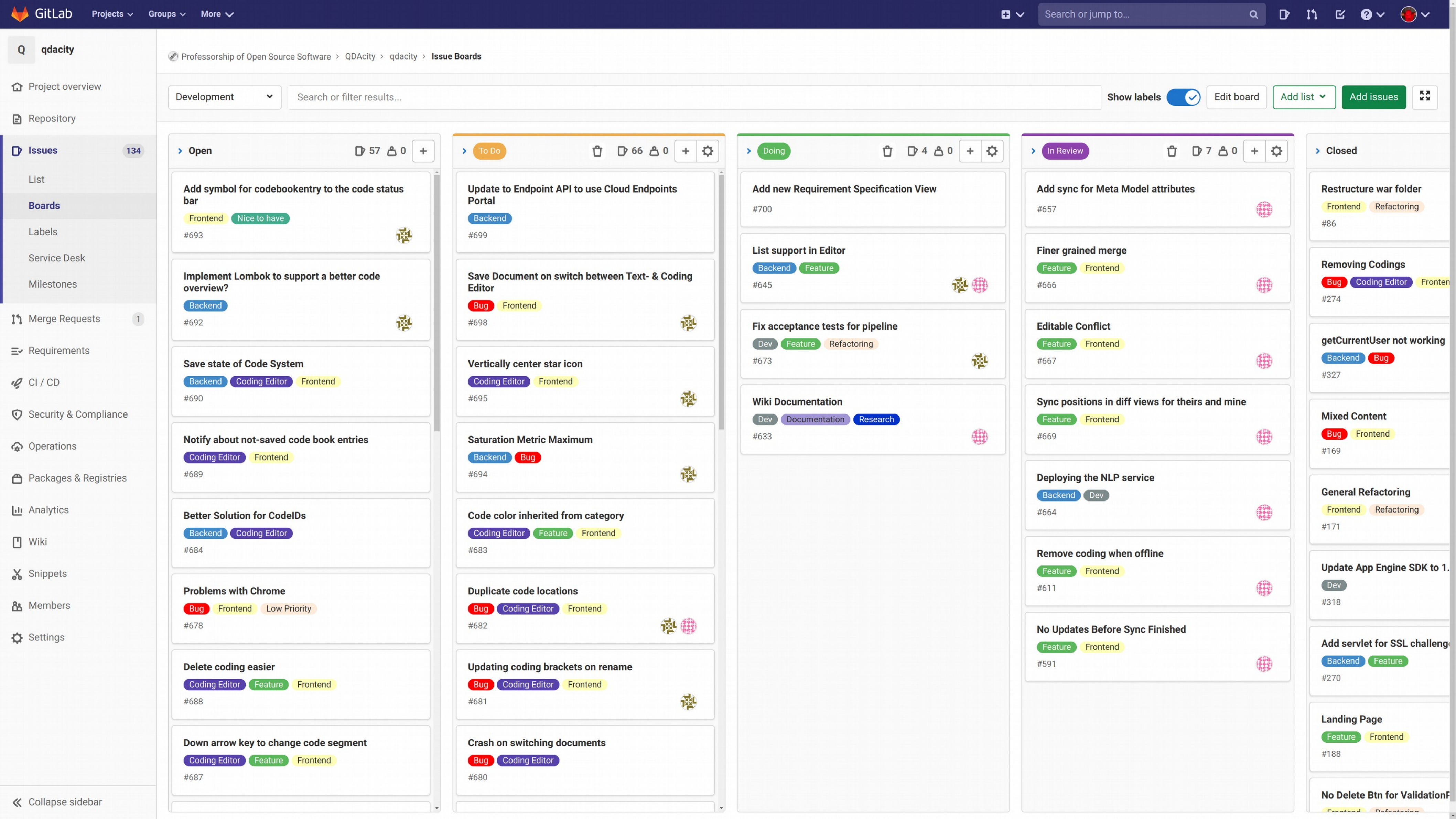
Down arrow key to change code segment
#687 · opened 8 months ago by Julia Coding Editor Feature Frontend — 0 — updated 8 months ago

Add Code by Drag'n'Drop
#686 · opened 8 months ago by Julia Coding Editor Feature Frontend — 0 — updated 8 months ago

Code System Search delete button
#685 · opened 8 months ago by Julia Feature Frontend — 0 — updated 8 months ago

Better Solution for CodeIDs
#684 · opened 9 months ago by Julia Backend Coding Editor — 0 — updated 9 months ago

Code color inherited from category — 0



Open 57 0

- Add symbol for codebookentry to the code status bar — Frontend, Nice to have — #693
- Implement Lombok to support a better code overview? — Backend — #692
- Save state of Code System — Backend, Coding Editor, Frontend — #690
- Notify about not-saved code book entries — Coding Editor, Frontend — #689
- Better Solution for CodeIDs — Backend, Coding Editor — #684
- Problems with Chrome — Bug, Frontend, Low Priority — #678
- Delete coding easier — Coding Editor, Feature, Frontend — #688
- Down arrow key to change code segment — Coding Editor, Feature, Frontend — #687

To Do 66 0

- Update to Endpoint API to use Cloud Endpoints Portal — Backend — #699
- Save Document on switch between Text- & Coding Editor — Bug, Frontend — #698
- Vertically center star icon — Coding Editor, Frontend — #695
- Saturation Metric Maximum — Backend, Bug — #694
- Code color inherited from category — Coding Editor, Feature, Frontend — #683
- Duplicate code locations — Bug, Coding Editor, Frontend — #682
- Updating coding brackets on rename — Bug, Coding Editor, Frontend — #681
- Crash on switching documents — Bug, Coding Editor — #680

Doing 4 0

- Add new Requirement Specification View — #700
- List support in Editor — Backend, Feature — #645
- Fix acceptance tests for pipeline — Dev, Feature, Refactoring — #673
- Wiki Documentation — Dev, Documentation, Research — #633

In Review 7 0

- Add sync for Meta Model attributes — #657
- Finer grained merge — Feature, Frontend — #666
- Editable Conflict — Feature, Frontend — #667
- Sync positions in diff views for theirs and mine — Feature, Frontend — #669
- Deploying the NLP service — Backend, Dev — #664
- Remove coding when offline — Feature, Frontend — #611
- No Updates Before Sync Finished — Feature, Frontend — #591

Closed

- Restructure war folder — Frontend, Refactoring — #86
- Removing Codings — Bug, Coding Editor, Fronten — #274
- getCurrentUser not working — Backend, Bug — #327
- Mixed Content — Bug, Frontend — #169
- General Refactoring — Frontend, Refactoring — #171
- Update App Engine SDK to 1. — Dev — #318
- Add servlet for SSL challeng — Backend, Feature — #270
- Landing Page — Feature, Frontend — #188
- No Delete Btn for ValidationF

5. Open Source Governance

Open Source Governance

- Open source governance
  - Is the process of governing the use of open source software in your projects
    - Initial selection of components
    - Management of dependency
    - Eventual replacement

Open Source Don’ts (Example Governance Rules 1 / 3)

- Do not copy source code with unsure license into your project codebase
  - Random code on the web without a license is proprietary code
- Do not copy source code that is copyleft-licensed (from wherever) into your codebase.
  - Do not copy from Stack Overflow (code is copyleft-licensed)
  - Do not copy and paste from open source projects
- Do not include copyleft-licensed libraries or other components into your project
- Do not blindly trust the license that an open source component is labeled with
- Do not combine software components with contradicting licenses

© 2020 Dirk Riehle - Some Rights Reserved

Open Source Dos (Example Governance Rules 2 / 3)

- Use permissively licensed open source components.
- Prefer governed sources like Apache or Eclipse over ungoverned sources like Github
- Maintain a bill-of-materials

© 2020 Dirk Riehle - Some Rights Reserved

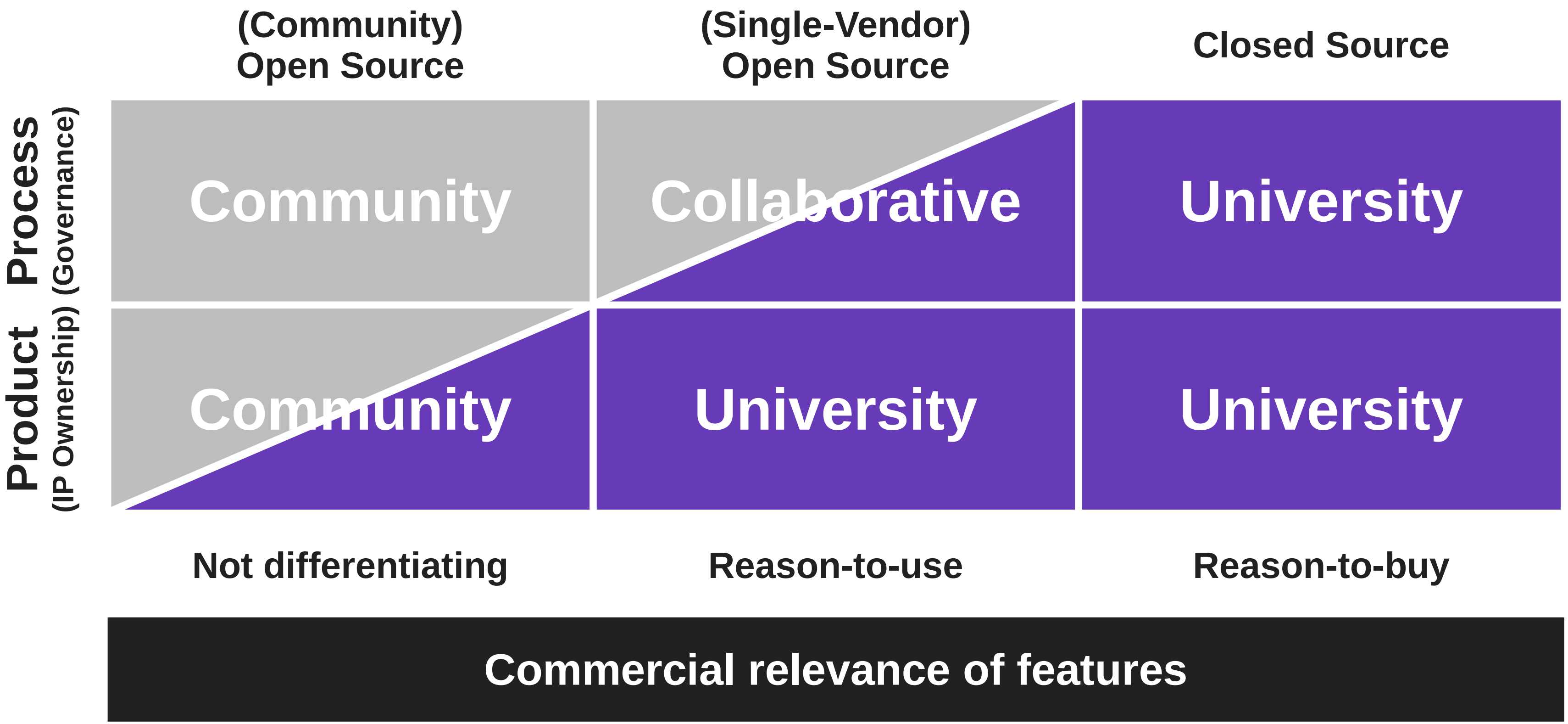
Projects, Licenses, and Sources

	Allowed	Must-ask	Denied
Projects	• PostgreSQL		
Licenses	• MIT • Apache 2.0 • All BSD variants	• EPL 1.1, EPL 2.0	• Any GPL license
Sources	• ASF website • Google Github repo • FB Github repo	• Linux Foundation • Eclipse Foundation	• Stack Overflow • Random website

© 2020 Dirk Riehle - Some Rights Reserved

6. Open Source Strategy

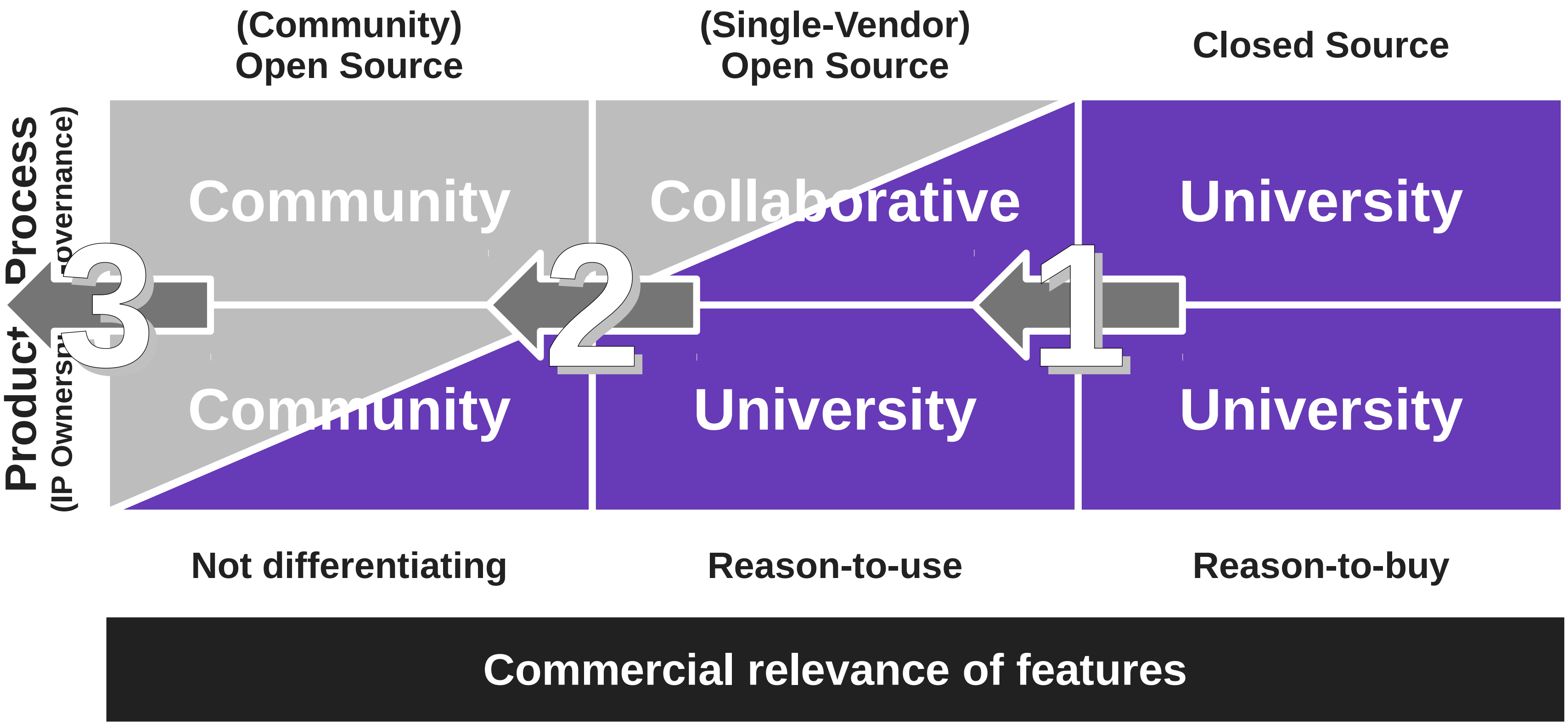
Open Source Community vs. University / Company



	(Community) Open Source	(Single-Vendor) Open Source	Closed Source
Process (Governance)	Community	Collaborative	University
Product (IP Ownership)	Community	University	University
	Not differentiating	Reason-to-use	Reason-to-buy

Commercial relevance of features

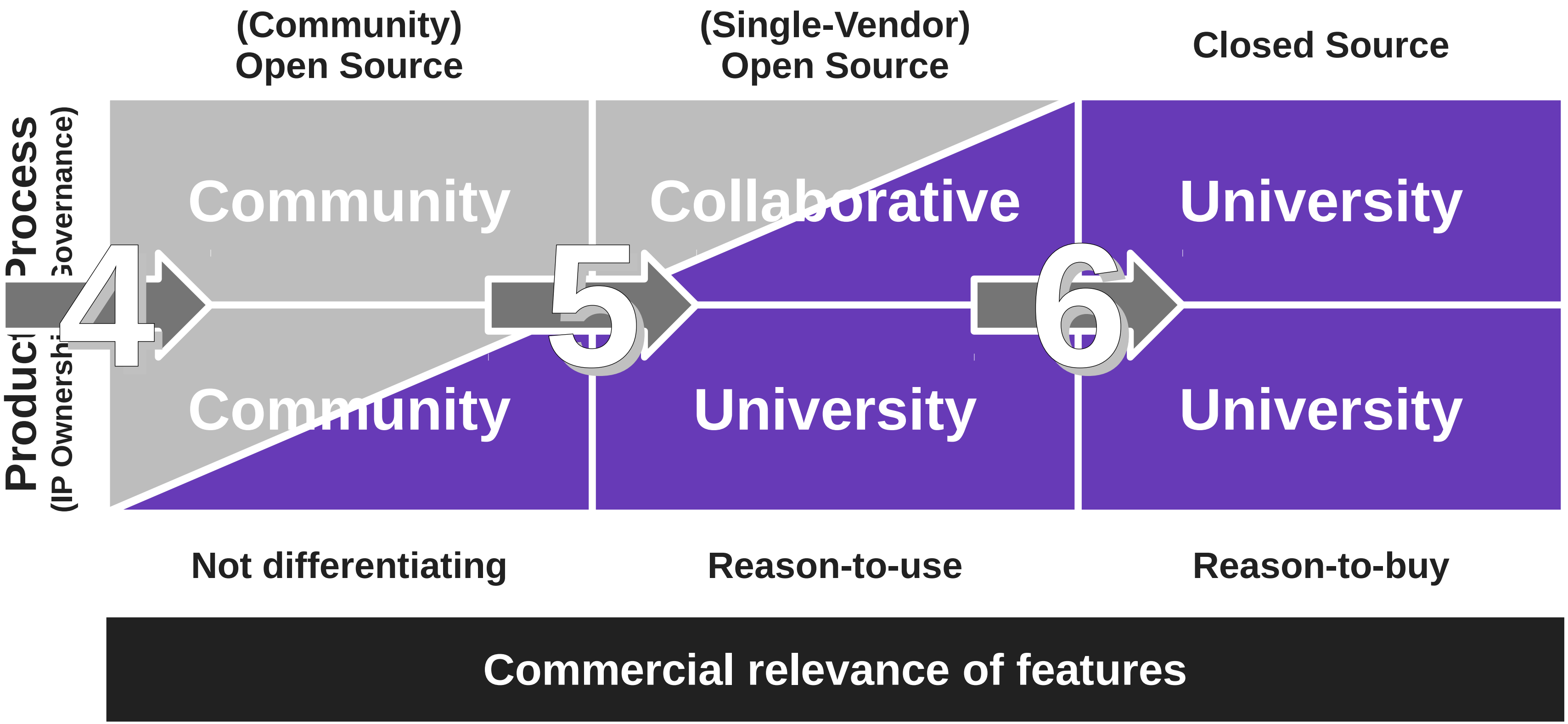
Pushing out Features



	(Community) Open Source	(Single-Vendor) Open Source	Closed Source
Process (Governance)	Community	Collaborative	University
Product (IP Ownership)	Community	University	University
	Not differentiating	Reason-to-use	Reason-to-buy

Commercial relevance of features

Pulling in Features



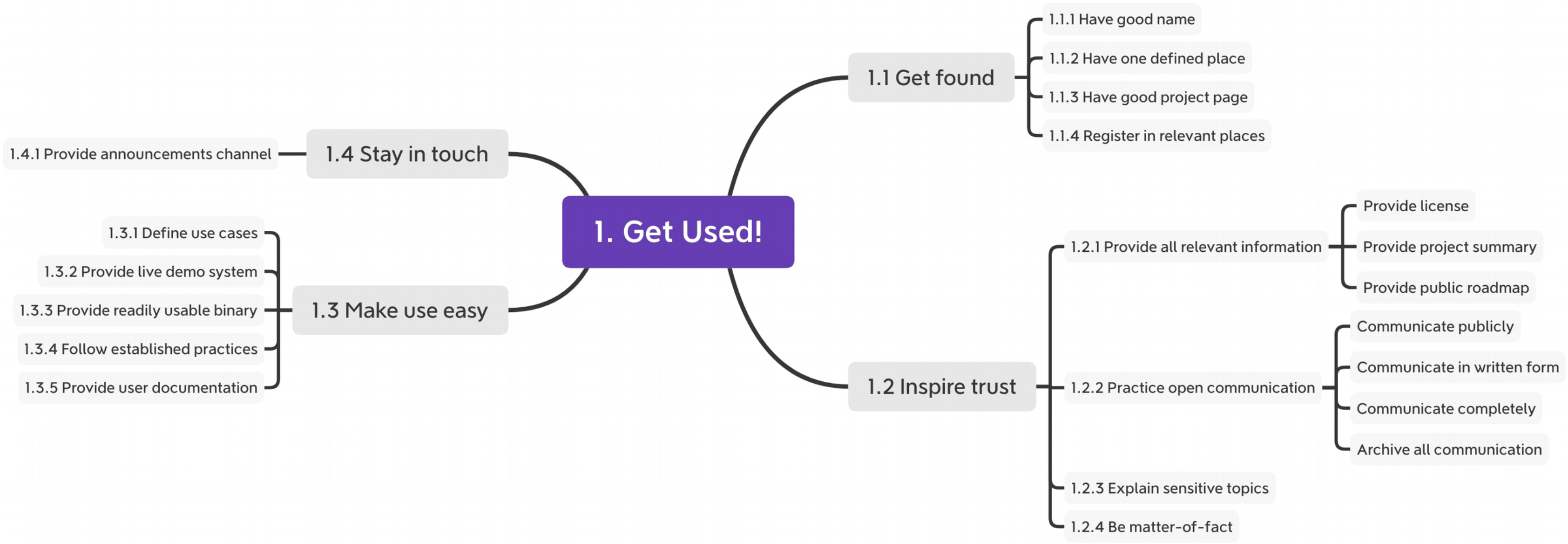
Public Money, Public Code?

- Universities have both a non-profit and commercial part
  - Universities are tasked with research and teaching [1]
  - Universities are also tasked with exploiting their innovations
- It is the university's choice what to give away for free

[1] And spending inordinate amounts on useless administrative matters

7. Setting-up Shop

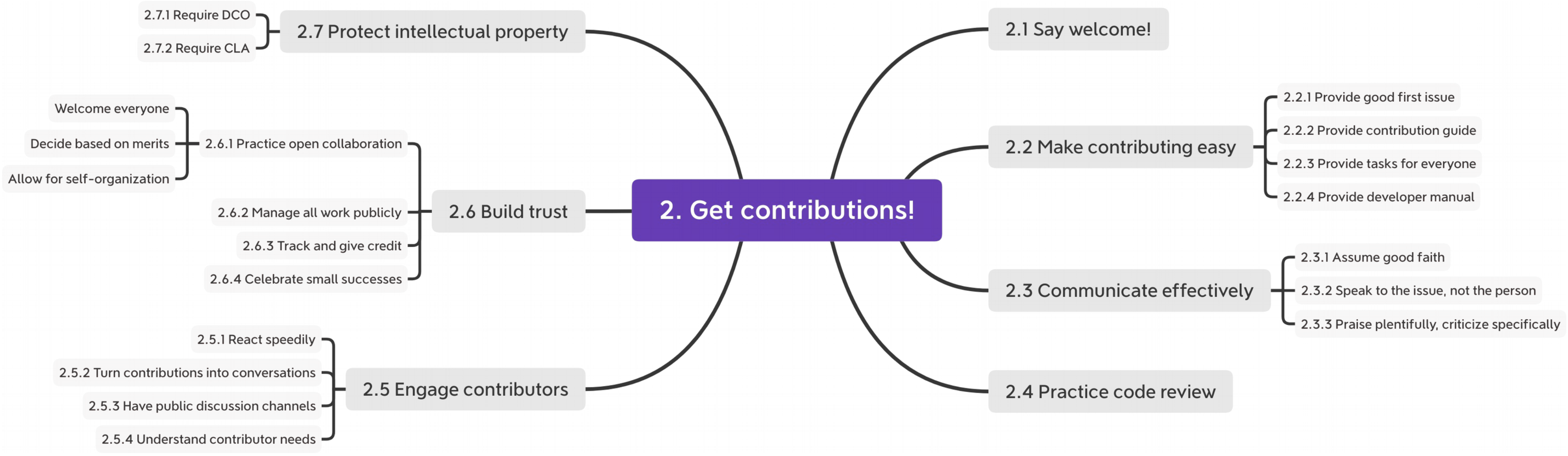
Best Practices for Setting-up Shop and Getting Users


1. Get Used!
1.1 Get found
1.1.1 Have good name
1.1.2 Have one defined place
1.1.3 Have good project page
1.1.4 Register in relevant places
1.2 Inspire trust
1.2.1 Provide all relevant information
Provide license
Provide project summary
Provide public roadmap
1.2.2 Practice open communication
Communicate publicly
Communicate in written form
Communicate completely
Archive all communication
1.2.3 Explain sensitive topics
1.2.4 Be matter-of-fact
1.3 Make use easy
1.3.1 Define use cases
1.3.2 Provide live demo system
1.3.3 Provide readily usable binary
1.3.4 Follow established practices
1.3.5 Provide user documentation
1.4 Stay in touch
1.4.1 Provide announcements channel

© 2020 Dirk Riehle - Some Rights Reserved

8. Getting Contributions

Best Practices for Getting Contributions



2. Get contributions!
2.1 Say welcome!
2.2 Make contributing easy
2.2.1 Provide good first issue
2.2.2 Provide contribution guide
2.2.3 Provide tasks for everyone
2.2.4 Provide developer manual
2.3 Communicate effectively
2.3.1 Assume good faith
2.3.2 Speak to the issue, not the person
2.3.3 Praise plentifully, criticize specifically
2.4 Practice code review
2.5 Engage contributors
2.5.1 React speedily
2.5.2 Turn contributions into conversations
2.5.3 Have public discussion channels
2.5.4 Understand contributor needs
2.6 Build trust
2.6.1 Practice open collaboration
Welcome everyone
Decide based on merits
Allow for self-organization
2.6.2 Manage all work publicly
2.6.3 Track and give credit
2.6.4 Celebrate small successes
2.7 Protect intellectual property
2.7.1 Require DCO
2.7.2 Require CLA

Summary

1. Research vs. startup
2. Project management
3. Product management
4. Lead programming
5. Open source governance
6. Open source strategy
7. Setting-up shop
8. Getting contributions

© 2020 Dirk Riehle - Some Rights Reserved

Thank you! Questions?

dirk.riehle@fau.de – http://osr.cs.fau.de

dirk@riehle.org – http://dirkriehle.com – @dirkriehle

Credits and License

- Original version
  - © 2020 Dirk Riehle, some rights reserved
  - Licensed under Creative Commons Attribution 4.0 International License

- Contributions
  - None yet