Produire de l'open source
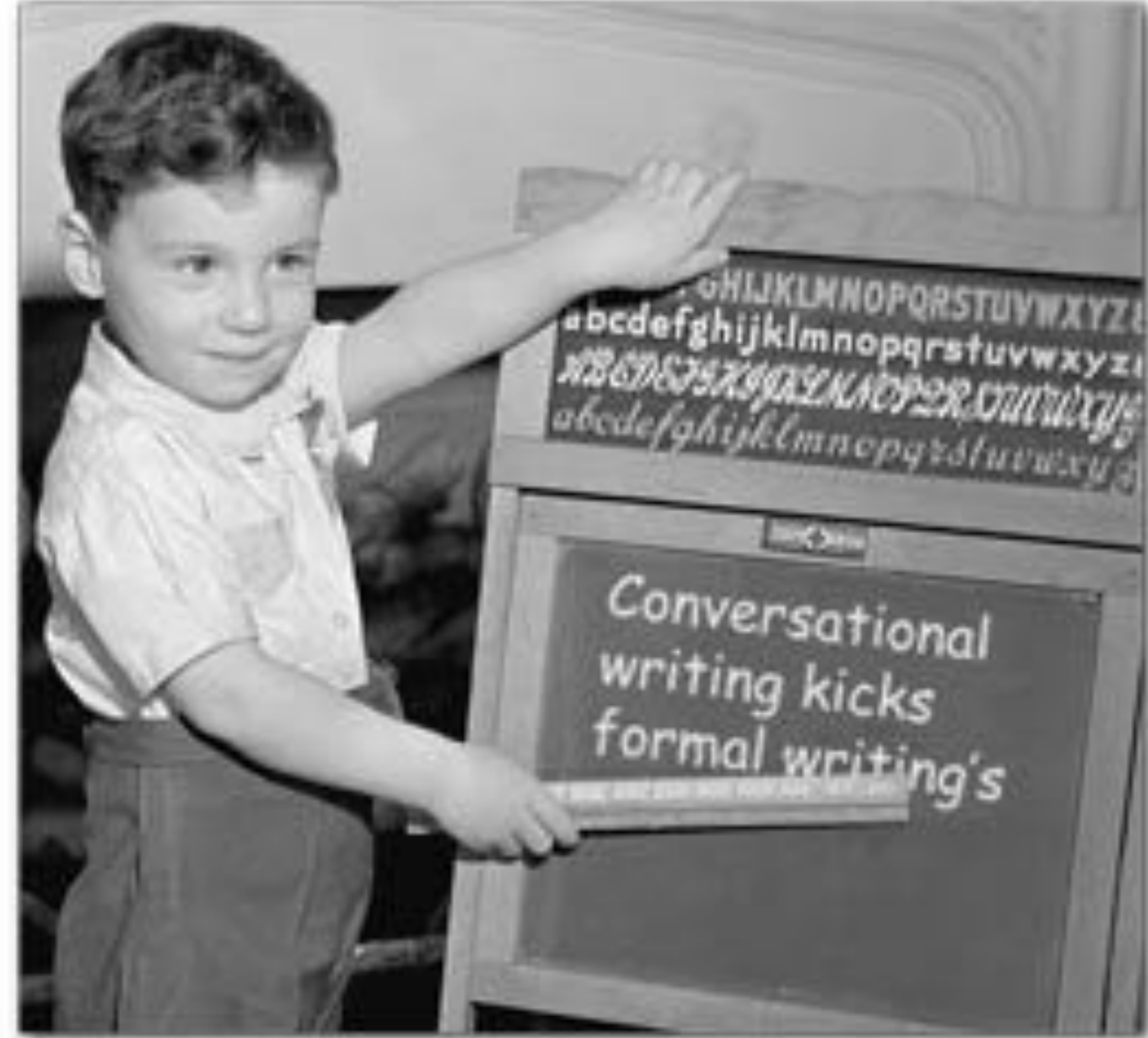
Pourquoi et comment créer une communauté ?

Stéfane Fermigier

Cours à l'EPITA - 18 oct. 2022

Une communauté: pourquoi?

Out-spend or Out-teach

Conversational writing kicks formal writing's

Source: Kathy Sierra

Source: Kathy Sierra
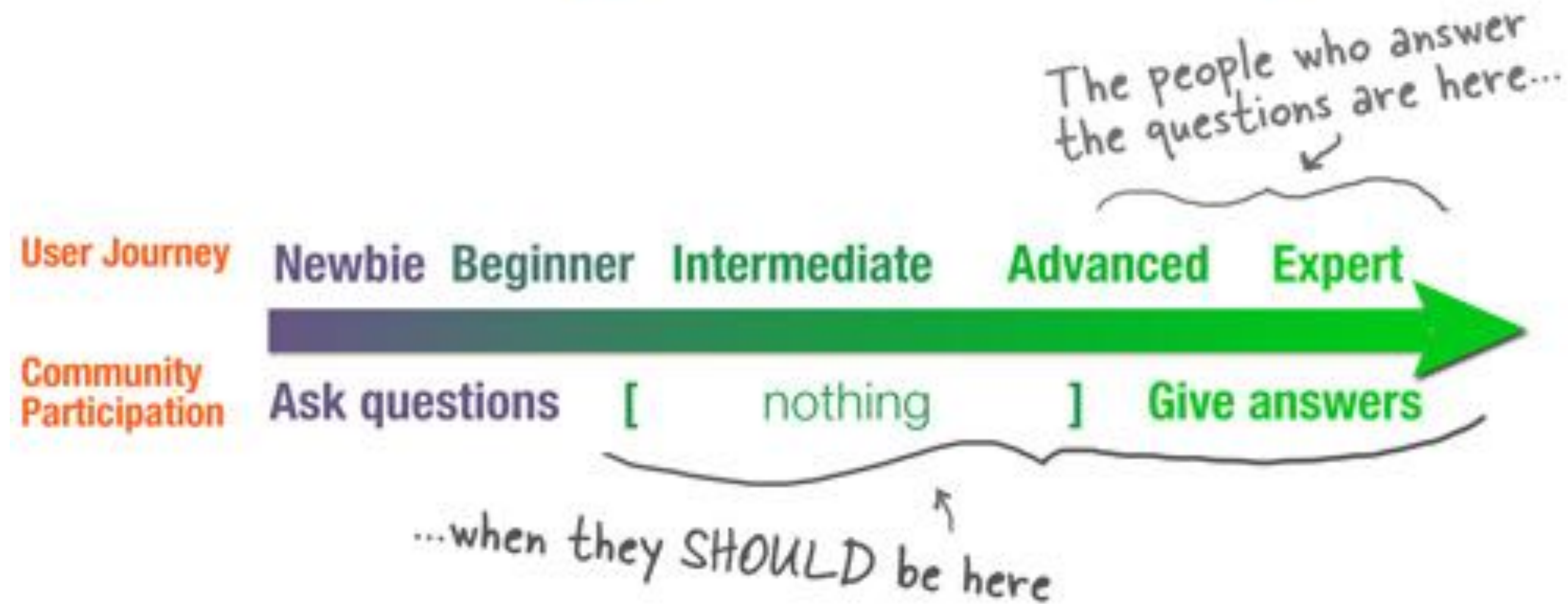
Community ROI — the more your users participate, the less money you need. Source: Kathy Sierra

Building a User Community

Source: Kathy Sierra

→ Common goals

- Get **feedback**

- Get **contributors**

- Improve our software **quality**

- Generate **buzz** and **evangelists**

- **Show** that we do have a community

Stratégie open source

Evolution classique

- *Software developed by communities of individuals*

- *Vendors begin to engage with existing open source communities*

- *Vendor-dominated open source development and distribution projects*

- *Corporate-dominated open source development communities*

Source: Matt Aslett, 451 Group

Modèle d'adoption
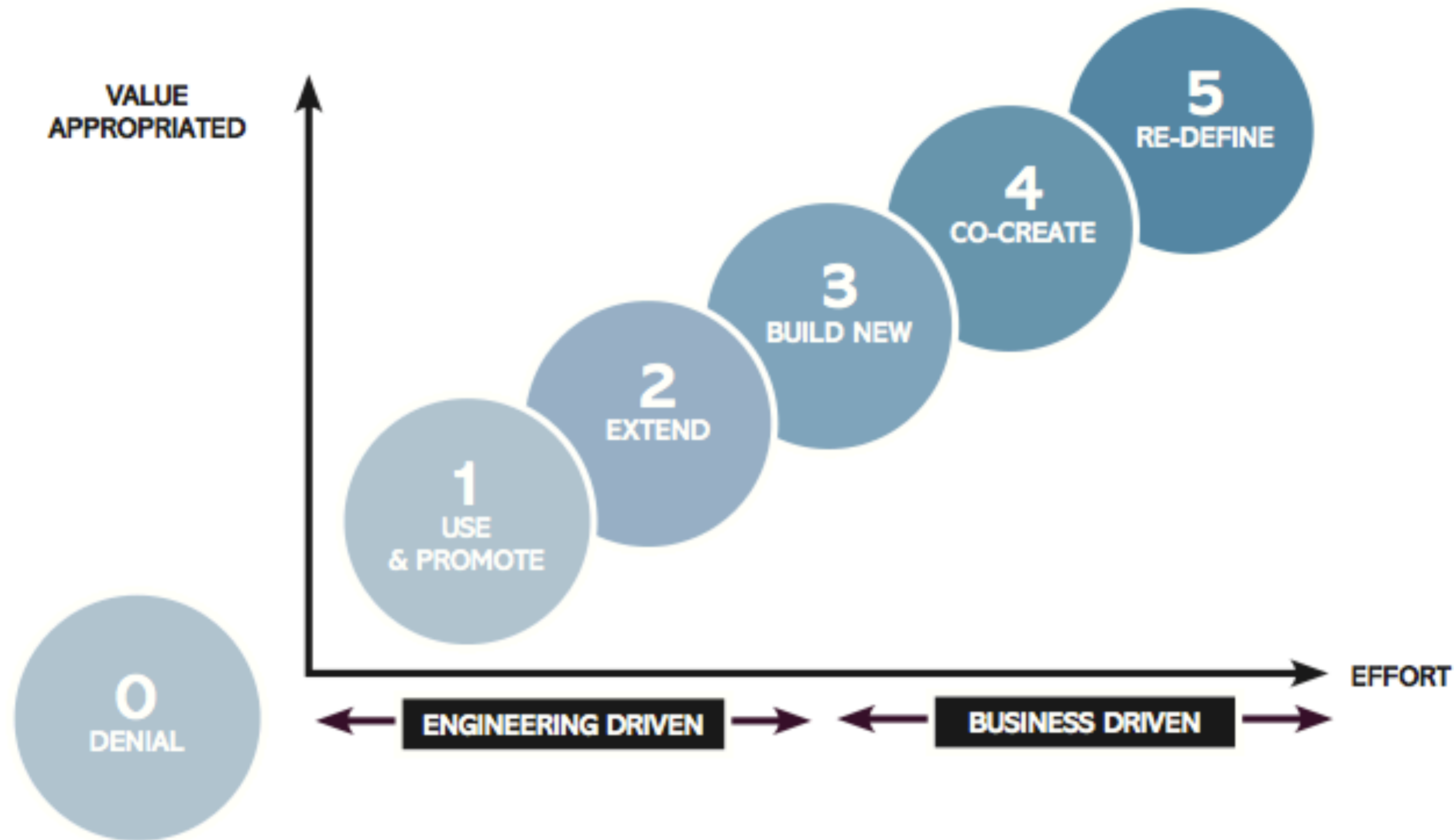
Source: Eclipse Foundation/Carleton University & Nortel

Eléments de stratégie - Pour un éditeur open source

- *Software License*

- *Copyright Ownership*

- *Development Model / Community*

- *Revenue Generator*

Marketing et "évangélisation"

Site Web

- Design
  - Utiliser / acheter un template "pro"
  - Tendance récente: Twitter Bootstrap
- Pitch (5 lignes)
  - Doit parler à des non-spécialistes
- *Features / benefits*

Site Web

- Définir l'audience cible

  - Segmenter si nécessaire

- *Progressive disclosure*

  - 1 minute / 5 minutes / 1 heure

- News et roadmap

  - Montrer qu'il y a de l'activité

Site Web

- Liens vers les outils communautaires (*cf. infra*)

- Liens vers les resources documentaires

  - Doc (architecture, utilisateurs)

  - Slides (SlideShare ou SpeakerDeck)

  - Screencasts

Le code

- Doit être facile à trouver, à builder ("configure ; make ; make install")

  - Comment gérer les dépendances ?

- README, INSTALL, etc.

  - Note: le fichier README est devenu crucial avec des outils comme GitHub

- Packaging (distribs Linux, Mac, Win...)

L'évangélisation selon Guy Kawasaki

- "Evangelism" became a business buzzword during the internet boom of the late 1990s. In fact, as Apple's second software evangelist, I helped popularize the term. The idea is simple: Derived from a Greek word that means, roughly, "to proclaim good news," evangelism is explaining to the world how your product or service can improve people's lives.

- My job at Apple was to proclaim the good news that Macintosh would make everyone more creative and productive. I wasn't just marketing a computer; I believed in it so much that I wanted others to experience it too. Now, as the chief evangelist of Canva, my job is to share a platform that democratizes design. Evangelists truly have the best interests of others at heart.

Emergence des "DevRel" (developer relations)

"DevRel is the marketing technique used to ensure that one's company, products, and developers establish a good, continuous relationship with external developers through mutual communication."
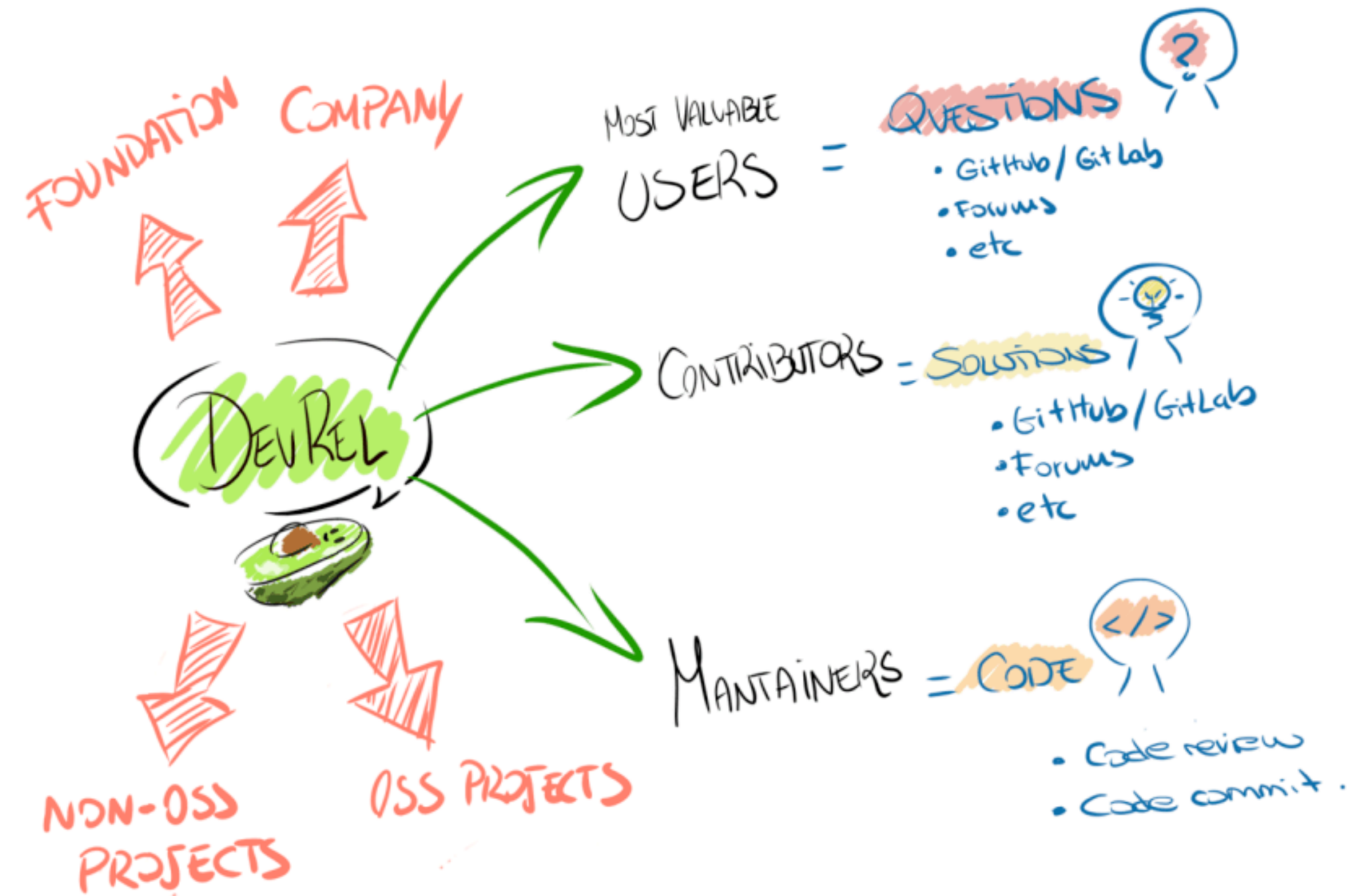
Activités en lien avec le DevRel

- A DevRel program may comprise a framework built around some or all of the following aspects:

  - **Developer Marketing**: Outreach and engagement activities to create awareness and convert developers to use a product.

  - **Developer Education**: Product documentation and education resources to aid learning and build affinity with a product and community.

  - **Developer Experience (DX)**: Resources like a developer portal, product, and documentation, to activate the developer with the least friction.

  - **Developer Success**: Activities to nurture and retain developers as they build and scale with a product.

  - **Community**: Nourishes a community to maintain a sustainable program.

Exemples d'animation

- Participation à / organisation de conférences

- Workshops

- Sprints

- Hackathons

- Club utilisateurs

DevRel vs. Community management

- La dénomination "Community manager" est trompeuse, le plus souvent il s'agit de "correspondants sur les réseaux sociaux"

Gouvernance
et modèle de développement

abilian

Modèles de gouvernance

- *Vendor-led*

  - Concessions possibles: club utilisateur, *board* plus ou moins indépendant et influent

- *Community led*

  - Formel ou informel

  - Communauté établie ("Fondation": FSF, ASF, Eclipse, OW2...) ou ad-hoc

Good Governance Initiative - OW2 & OSPO.zone

Abraham Maslow's Hierarchy of Behavioral Motivation

Strategy	Embracing the full potential of OSS. Proactively using OSS for innovation and competitiveness.
Engagement	Engaging with the OSS ecosystem. Contributing back. Developing visibility, event participation.
Culture	Implementing best practices. Developing OSS culture. Sharing experience.
Trust	Securely and responsibly using OSS. Compliance and dependency management policies.
Usage	Technically using OSS. Technical ability and experience with OSS. Some OSS awareness.

OW2 Goals to OSS Good Governance

Modèles de décision dans les gouvernance communautaires

- Hiérarchie des membres

  - Contributeur, *committer*, *core committer...*

- Unanimité, consensus ou *BDFL* ?

  - Qui porte la vision ? Comment est-elle partagée ?

  - Enjeux? Vitesse d'exécution, masse critique ?

Considérations juridiques

Propriété du code

- Centralisée?

    - Chez l'éditeur

    - Au sein d'une communauté

- Ou partagée?

- Notion de *contributor's agreement* (CLA / CCA) + DCO

Choix des licences

- Contrat moral avec la communauté

  - Tout changement risque d'être vécu de manière traumatique

- Contraintes business

  - Ex: *open core, double licensing*

  - Copyleft / weak copyleft / pas de copyleft

Choix des licences

- ~100 licences reconnues par l'OSI, 8 "*popular and widely used or with strong communities*"

  - BSD, MIT, (L)GPL, APL, MPL, EPL, CDDL

- Critères importants:

  - Compatibilité GPL (en général désirable)

  - Compatibilité intégration avec du propriétaire (choix)

Évolution des pratiques

FOSS 0.1
1983-1990

Richard Stallman, Founder of the Free Software Movement

- The free software movement was started in 1983 by Richard Stallman

- Most of the open source software produced at the time was developed by **very small teams** (2-3 persons), using local development tools

- Software were distributed using **tapes**, then **FTP**

- Marketing was mostly through word-of-mouth

- Community = people with internet access (not many people)

Early successes

- The GNU "operating system" (minus the kernel) was already displacing proprietary tools in the early 90s

- The moral and legal frameworks upon which the free software (and later, the open source) movement is built

  - Didn't mandate / prescribe any production model for free software, though

Challenges

- Economic and moral questioning:

  - **Is it ok to make money with free software?**

  - How to make the system **sustainable**?

- How to **scale development** efforts to larger teams?

FOSS 0.9
1991-1998

Successes

- **Larger scale projects** start to appear, attracting tens, then hundreds of developers (and later, thousands)

- **Tools** and **practices** are developed, most often on top of existing internet protocols to address the needs of distributed development at this scale :

  - Centralized source code management

  - Mailing lists or usenet forums

Successes

- Linux (1991)

- The Debian (1993) and Red Hat (1994) distributions

- The Apache Web Server (1995)

FOSS 1.0
1998-2007

- **Open source** becomes the preferred term for most free software based businesses

- The **Web** becomes pervasive

- Several organizations created to foster **governance** of open source projects (Apache Foundation, Eclipse Foundation, OW2...)

- Several successful IPOs on top of the Web 1.0 bubble (Red Hat, VA Linux), Netscape open sources the Mozilla browser...

The 4 engines of collaboration

- Real-time **shared vision**

- Real-time **status updates**

- Real-time **help requests**

- Self-service **archives**

Source: Bertrand Delacretaz, 2009

OS Secret Sauce

"Every successful open source project I know uses **PRIM**. Every closed source project I know, doesn't. People wonder how open source projects manage to create high-quality products without managers or accountability. The answer: **we're accountable to our infrastructure**. PRIM is the **open source secret sauce**."

Ted Husted http://jroller.com/TedHusted/entry/prim

P = Portal (often, a Wiki)

R = Repository

Repositories list   **Mercurial**

Name	Description	Contact	Last change		
addons	Nuxeo Addons	Florent Guillaume <fg@nuxeo.com>	2 weeks ago	zip	RSS Atom
addons/nuxeo-core-folder-versioning	Nuxeo Core Folder Versioning	Florent Guillaume <fg@nuxeo.com>	12 months ago	zip	RSS Atom
addons/nuxeo-correspondence-marianne	Nuxeo Correspondance Mariane	Florent Guillaume <fg@nuxeo.com>	11 minutes ago	zip	RSS Atom
addons/nuxeo-features-page-layout	Nuxeo Features Page Layout	Florent Guillaume <fg@nuxeo.com>	9 months ago	zip	RSS Atom
addons/nuxeo-generic-wss-handler	Nuxeo Generic WSS Handler	Florent Guillaume <fg@nuxeo.com>	4 months ago	zip	RSS Atom
addons/nuxeo-http-client	Nuxeo HTTP Client	Florent Guillaume <fg@nuxeo.com>	4 weeks ago	zip	RSS Atom
addons/nuxeo-platform-annotations	Nuxeo Platform Annotation service addon	Florent Guillaume <fg@nuxeo.com>	3 months ago	zip	RSS Atom
addons/nuxeo-platform-archive	Nuxeo Platform Archive service addon	Florent Guillaume <fg@nuxeo.com>	10 months ago	zip	RSS Atom
addons/nuxeo-platform-archive-importer	Nuxeo Platform Archive Importer plugin	Florent Guillaume <fg@nuxeo.com>	18 months ago	zip	RSS Atom
addons/nuxeo-platform-audio	Nuxeo Platform Audio Addon	Florent Guillaume <fg@nuxeo.com>	3 weeks ago	zip	RSS Atom
addons/nuxeo-platform-audit-web-access	Nuxeo Platform Audit Web Access	Florent Guillaume <fg@nuxeo.com>	3 months ago	zip	RSS Atom
addons/nuxeo-platform-categorization	Nuxeo Automated Document Categorization: framework and service	Florent Guillaume <fg@nuxeo.com>	2 weeks ago	zip	RSS Atom
addons/nuxeo-platform-categorization/nuxeo-platform-categorization-coverage	Nuxeo Automated Document Categorization: dc:coverage model	Florent Guillaume <fg@nuxeo.com>	2 weeks ago	zip	RSS Atom
addons/nuxeo-platform-categorization/nuxeo-platform-categorization-language	Nuxeo Automated Document Categorization: dc:language model	Florent Guillaume <fg@nuxeo.com>	2 weeks ago	zip	RSS Atom
addons/nuxeo-platform-categorization/nuxeo-platform-categorization-subjects	Nuxeo Automated Document Categorization: dc:subjects model	Florent Guillaume <fg@nuxeo.com>	2 weeks ago	zip	RSS Atom
addons/nuxeo-platform-document-categorization	Nuxeo Automated Document Categorization	Florent Guillaume <fg@nuxeo.com>	2 weeks ago	zip	RSS Atom
addons/nuxeo-platform-documentLink	Application Level document proxies for Nuxeo5	Florent Guillaume <fg@nuxeo.com>	4 months ago	zip	RSS Atom

I = Issue (or Bug)Tracker

M = Mailing List (+ foruM)

Software Forges, a more integrated approach

- **Sourceforge**, launched in 1999 by VA Linux, integrates all these tools in a consistent Web (1.0) portal

- Makes it super easy for anyone (3.4 million users currently) to start a new open source project (324 000 as of today)

- Several similar products launched afterwards (Collabnet, Trac, Redmine)

Works for non open source software too...

Open Collaboration within Corporations Using Software Forges

Dirk Riehle, John Ellenberger, Tamir Menahem, Boris Mikhailovski, Yuri Natchetoi, Barak Naveh, and Thomas Odenwald, *SAP*

IEEE Software

SAP
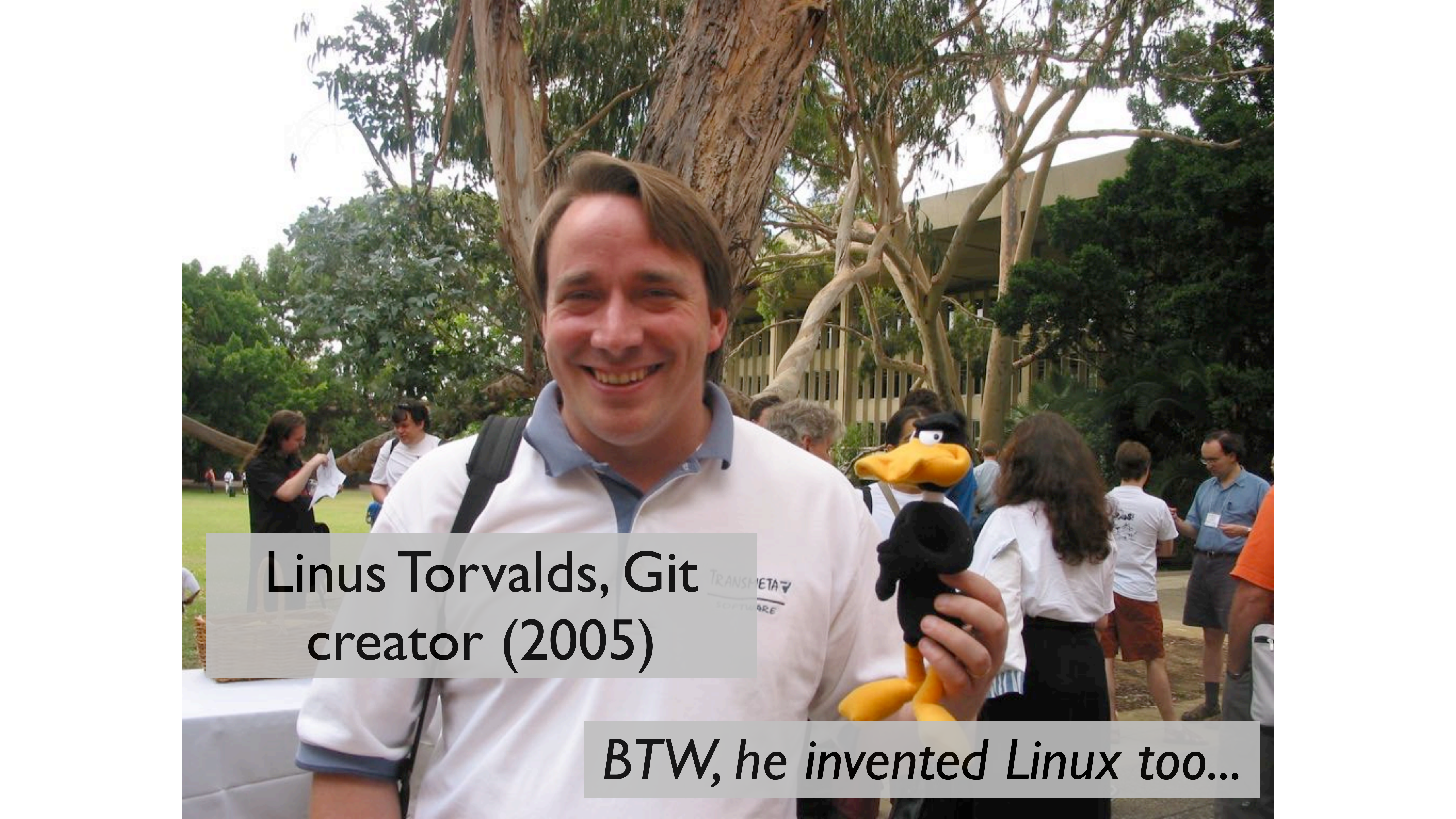
FOSS 2.0
2008-2018

Web 2.0

- Wikipedia (2001)

- Tim O'Reilly's *Architecture of Participation* (2004) and *Web 2.0* (also 2004)

- Consumer Web 2.0, then Enterprise 2.0 replace older applications

- **Git**, and a bunch of other Distributed Source Control Management Systems (DSCM), appear circa 2005 to address the need of very large distributed development teams (1000s of developers for Linux)

- They allow for completely **decentralized development**, and make it much easier for developers to try out new ideas on their own, then "merge" the changes with the main development lines

Linus Torvalds, Git creator (2005)

BTW, *he invented Linux too...*

- A new breed of SaaS offerings for developpers, such as GitHub (2008) or StackOverflow (2008), appear, leveraging many of the characteristic features of W2.0 or E2.0 applications:

  - Activity streams

  - Social networking

  - Tagging / folksonomies

  - Votes, reputation

GitHub, like SourceForge, but more social

StackOverflow, a knowledge base based on a reputation system

Additional tools with a social impact

- **Continuous integration** (with a strong testing culture) allows distributed development to happen with confidence that developers don't "break the build"

- **Code review** applications

Continuous integration

Code review on GitHub

FOSS "2.0+(?)" (?)
2018-now

- GitHub est devenu quasiment incontournable, par application de la loi de Metcalfe

- GitHub, qui avait développé un aspect "plateforme" (i.e. permettant l'intégration de services tiers: CI/CD, audit de code, *SDLC management...*) a fini par intégrer certains de ces services, entrant ainsi en concurrence avec son écosystème

- Le rachat de GitHub par Microsoft fait par ailleurs tiquer de nombreux acteurs, et on voit l'émergence de nouvelles plateformes ou la popularité croissante de plateformes existantes

Conseils pratiques et activités
dans le contexte d'un éditeur open source

People first

- Give warm welcomes to new members

- Thank contributors

- Give positive feedback

- Act quickly on new contributions (thank you, feedback, commit)

- Never forget to give credit (CONTRIBUTORS.txt, release notes)

Make it easy to become a contributor

- It should be easy to add or fix a translation, a particular bit of documentation, a FAQ entry, etc.

- It should also be easy to contribute new modules (add-ons)

- This is the whole idea of "**The architecture of participation**" (O'Reilly, 2004)

When to give away the "commit bit"

- New contributors have to go though a learning process and build trust before being allowed to commit directly on the code repository

- Ask them first to submit patches on the issue tracker

- Some legal paperwork can be required (CLA / CCA / DCO)

- Practices vary widely between different projects

Engage with people

- Be generic:

  - Solicit feedback ("what do you think of...?")

  - Ask for beta testers, bug reports

- Be specific:

  - Link to the right places (relevant space on issue tracker, forum, FAQ entry, etc.)

  - Engage with specific people

The Roadmap

- Make the roadmap clear and visible

  - Publish plan for at least next minor and major releases

  - Include *tentative* dates and scope (make it clear it *is* tentative, though)

- Make it consistent with the Issue Tracker (and the reality)

- Ask for feedback and contributions

Get good at Email

- (Assuming you're using a mailing list; this is probably similar w/ a forum like Discourse)

- Reformulate until everything's 100% clear

- Make your emails easy to read (short paragraphs, skip one line btw paragraphs...)

- Don't over quote previous messages, but keep some context

- Use URLs to quote previous conversations or online documents

Blog

- Some email messages (new features, etc.) should be written as blog posts, *then* sent to the mailing list (either copied or as links)

- Put pictures or diagrams on your blog posts

- Weekly / monthly technical reports

- Reinforce with tweets and other status updates

FAQ and READMEs

- There should be one README in each project module (even if it's only one link to a particular web page)

- Read "README-driven development" (by Tom Preston-Werner):

  - *"Write your Readme first. First. As in, before you write any code or tests or behaviors or stories or ANYTHING. I know, I know, we're programmers, dammit, not tech writers! But that's where you're wrong. Writing a Readme is absolutely essential to writing good software."*

- Constantly update the FAQ with questions asked on the mailing list or feedback from the community

Community vs. Support

- If someone's obviously using the community as a substitute for support, let others deal with him

- Don't support people that never give anything in return

- Aggressive people should be dealt with with care, and certainly not by being aggressive in return

Community vs. Sales

- When you identify interesting people in the community, pass useful information to sales

- Sometimes hint that we are doing interesting projects for real customers (without giving away confidential information)

- Give information to help people make their case for using the product in their organization

Make it easy to build our software

- Indispensable pour avoir des contributeurs (et des utilisateurs)

- Old way:

  - "./configure ; make ; make install"

- New way:

  - "pip install .", "mvn install", "npm install", "go build", etc.

  - Or docker / docker-compose

Provide good documentation

- C'est un métier ("Technical writer"), mais aussi une activité à ne pas négliger pour les développeurs

- Intégration avec le code ("API doc", etc.)

- Outils populaires: Sphinx, Mkdocs, etc.

- Sources:

  - https://www.writethedocs.org/

  - Books

Practice open decision making

- Processus de décisions transparents, concernant notamment:

  - Les corrections et améliorations mineures (bug trackers publics)

  - Les améliorations substantielles (ex: "PEPs = Python Enhancement Proposals")

  - La nomination des personnes décisionnaires

Dealing with poisonous people

- "Code of conduct" / "Charters"

- Enforcement

- Cf. Cathy Sierra évoqué en début de présentation

Relationship with other communities

- Upstream: languages, libraries

- Sidestream: plugins creators

- Downstream: packagers (e.g. for Linux distributions)

- Customers: OSPOs

Recap

- It's about **people, first**: getting to know each other, making sense of the crowd, creating a sense of belonging

- Always be **respectful**, **transparent**, **authentic** and **helpful**

- Contribute to the **architecture of participation**

References / Credits

- https://producingoss.com/ (Karl Fogel)

- http://www.artofcommunityonline.org/ (Jono Bacon)

- http://www.slideshare.net/bdelacretaz/life-in-open-source-communities-apachecon-us-2009 (Bertrand Delacretaz)

- http://www.slideshare.net/jaaronfarr/making-open-source-work-presentation (J. Aarron Farr)

- http://lwn.net/Articles/370157/ (Josh Berkus)

- http://headrush.typepad.com/ (Kathy Sierra)