

3TC37: Contribution à un logiciel libre

Modèles de gouvernance, communautés

Marc Jeanmougin
Théo Zimmermann

Télécom Paris, Institut Polytechnique de Paris

CC BY-SA 4.0



Communautés

Communautés

Communautés

En général, un projet se structure autour d'une *communauté*.

Communautés

Communautés

En général, un projet se structure autour d'une *communauté*.

- **Utilisateurs/ices (et clients)**
- **Contributeurs/ices**

Communautés

Communautés

En général, un projet se structure autour d'une *communauté*.

- **Utilisateurs/ices (et clients)**
- **Contributeurs/ices**

On parle d'**organisations** pour les entités capables de gérer les aspects légaux : dons, contrats, etc.

- **Associations, entreprises, fondations, institutions, etc.**

Communautés

En général, un projet se structure autour d'une *communauté*.

- **Utilisateurs/ices (et clients)**
- **Contributeurs/ices**

On parle d'**organisations** pour les entités capables de gérer les aspects légaux : dons, contrats, etc.

- **Associations, entreprises, fondations, institutions, etc.**

On observe une grande variété de situations : plusieurs projets peuvent partager une communauté, etc.

Communautés

Interactions

Les **individus** qui forment une communauté interagissent selon des normes sociales implicites ou explicites. Lorsqu'elles sont explicites, on parle souvent de **code of conduct** (code de bonne conduite).

Exemples : Ubuntu, Django

Ces normes définissent aussi les interactions pouvant arriver entre différentes catégories de personnes : salariés, bénévoles, etc.

Types de contributions

Lister les manières de *contribuer* à un projet

Identification des acteurs dans un projet

Collaboration $\neq$ Égalité

Qui contrôle le projet ? Qui prend les décisions ? Quels sont les rôles de chacun ?
Quelles sont les organisations impliquées, et quels sont ses intérêts ?

Le droit au fork

Un *fork* est une copie du code qui diverge du projet original.

Un fork peut avoir plusieurs usages :

- **Contribuer** des changements dans le cadre d'un développement basé sur les pull requests (*development fork*).
- **Maintenir** un projet abandonné par son auteur initial.
- Créer un **projet concurrent** dirigé par une équipe indépendante (*hard fork*).

Les licences libres **garantissent le droit** de distribuer des copies modifiées, donc de créer un fork.

Le droit au fork

Ce droit a un impact sur :

- La **durabilité** :

Le droit au fork

Ce droit a un impact sur :

- La **durabilité** :

Même si une entreprise décide unilatéralement de cesser de développer un logiciel libre, un fork peut permettre de **préserver** ce logiciel et de **continuer son développement**.

Le droit au fork

Ce droit a un impact sur :

- La **durabilité** :

Même si une entreprise décide unilatéralement de cesser de développer un logiciel libre, un fork peut permettre de **préserver** ce logiciel et de **continuer son développement**.

- La **confiance** :

Le droit au fork

Ce droit a un impact sur :

- La **durabilité** :

Même si une entreprise décide unilatéralement de cesser de développer un logiciel libre, un fork peut permettre de **préserver** ce logiciel et de **continuer son développement**.

- La **confiance** :

Les utilisateurs et les contributeurs peuvent être rassurés que si le projet s'arrête ou prend une mauvaise direction, il sera toujours **possible d'y remédier**.

Le droit au fork

Ce droit a un impact sur :

- La **durabilité** :

Même si une entreprise décide unilatéralement de cesser de développer un logiciel libre, un fork peut permettre de **préserver** ce logiciel et de **continuer son développement**.

- La **confiance** :

Les utilisateurs et les contributeurs peuvent être rassurés que si le projet s'arrête ou prend une mauvaise direction, il sera toujours **possible d'y remédier**.

- La **gouvernance** :

Le droit au fork

Ce droit a un impact sur :

- La **durabilité** :

Même si une entreprise décide unilatéralement de cesser de développer un logiciel libre, un fork peut permettre de **préserver** ce logiciel et de **continuer son développement**.

- La **confiance** :

Les utilisateurs et les contributeurs peuvent être rassurés que si le projet s'arrête ou prend une mauvaise direction, il sera toujours **possible d'y remédier**.

- La **gouvernance** :

Même lorsqu'un projet libre est officiellement dirigé par un "dictateur", sa gouvernance est toujours **partiellement démocratique / méritocratique**.

Apache : un fork communautaire

By The Apache Software Foundation and Vulphere, Apache License 2.0



- NCSA HTTPd était l'un des tout premier serveur web et le plus utilisé dans le monde de 1993 à 1995. (Développé à l'Université d'Urbana-Champaign par Robert McCool.)
- Le projet ayant cessé d'être maintenu, ses utilisateurs décident d'en produire un **fork** en agrégeant des patchs qui circulaient.
- Le nom est un jeu de mot accidentel avec "a patchy server".
- **Serveur web le plus utilisé** dans le monde de 1995 à 2016.
- Donne lieu à la création de l'**Apache Foundation** et de la **licence Apache 2.0**.

Exemple de hard fork : ECGS

- GCC était maintenu par la FSF, de manière très **conservatrice** sur l'ajout de nouvelles fonctionnalités.
- EGCS (Experimental/Enhanced GNU Compiler System) démarra comme un fork rassemblant divers changements qui avaient été proposés par des contributeurs et continua sur un mode de **développement plus ouvert** (mais tout en restant synchronisé avec les modifications faites dans GCC).
- Des distributions Linux commencèrent à adopter EGCS à la place de GCC comme compilateur C.
- La FSF reconnut la validité du modèle de développement d'EGCS et donna le contrôle du projet GCC aux mainteneurs d'EGCS, **mettant fin à la divergence**.

Modèles de gouvernance

Modèle Linux : dictateur bienveillant

- Traditionnellement, de nombreux projets open source fonctionnent avec un “**dictateur bienveillant**”. Dans le projet Python, le titre était “BDFL” qui veut dire “Benevolent Dictator For Life”.
- Le dictateur tire sa légitimité de ses compétences techniques et humaines et de son **implication de long terme** dans le projet. C’est très souvent l’**auteur initial** du projet.
- Le dictateur est **celui/celle qui décide** quelles modifications sont acceptées dans le projet. Mais la plupart du temps, ce pouvoir est **délégué** :
 - Par exemple, dans le modèle de gouvernance de Linux, Linus Torvalds est le dictateur et ses “lieutenants” sont les mainteneurs de divers composants du noyau.

Modèle Apache : méritocratie hiérarchisée

Le modèle de gouvernance de la Fondation Apache définit une hiérarchie de rôles au sein d'un projet :

- **Utilisateurs** : participent aux discussions/à rapporter des bugs.
- **Contributeurs** : participent à l'évolution du projet par des contributions de code ou de documentation.
- **Committers** : ont un accès en écriture au projet.
- Membres du **PMC** (Project Management Committee) : dirigent le projet et décident de son évolution.

Le PMC peut rejeter une contribution d'un committer, mais ne peut forcer personne à faire une contribution, c'est pourquoi ce modèle (comme dans tout projet open source) est aussi une "do-ocracy" (**le pouvoir appartient à ceux qui agissent = do**).

Cooptation

- Acquérir des **droits en écriture** (commit) sur le projet est un signe de confiance de la part des autres mainteneurs.
- En général, on ne les accorde qu'à des **contributeurs réguliers**.
- Les mainteneurs (ou le dictateur, ou le PMC) **décident quand proposer** à des contributeurs d'acquérir ces droits. Cela peut faire suite à une discussion dont il est naturel qu'elle ait lieu **en privé**.
- Il peut y avoir une distinction entre des **droits limités** (maintenance d'un composant) et des droits globaux (sur tout le projet).
- Le PMC (ou équivalent) peut être renouvelé également par cooptation ou être élu (par un électorat composé des committers ou plus vaste).

Exemple : projet Python

- Guido van Rossum (créateur du langage) était le BDFL jusqu'à ce qu'il **démissionne** en 2018.
- La gouvernance actuelle (inspirée de Django) est composée :
 - D'une **équipe cœur** (106 membres actifs).
 - D'un **comité de direction** (5 membres).
- Les nouveaux membres de l'équipe cœur doivent être validés par un vote de 2/3 des membres actifs.
- Le comité de direction est **élu par l'équipe cœur** et renouvelé régulièrement (tous les ans). Il a un fort **pouvoir décisionnaire**, à n'utiliser qu'en dernier recours.
- Pas plus de deux membres du comité de direction ne peuvent avoir le même employeur.

Le consensus paresseux

- Quel que soit le modèle exact de gouvernance d'un projet open source, il est standard que la plupart des décisions soient prises par **consensus paresseux** :
 - Les oppositions doivent être **explicites**.
 - Les silences sont considérés comme des approbations.
 - Les oppositions sont prises au sérieux et quasiment équivalentes à des **vetos** (selon par qui elles sont émises).
- Selon l'importance de la décision : **processus + ou – formel**.
 - Le – formel : agir (pousser un changement) et annuler l'action en cas de protestation (*revert*).
 - Le + formel : déclarer la décision qui va être prise si personne ne s'y oppose pendant une période définie (de plusieurs jours), *"Final Comment Period"*.

Les PEP / RFC

Certains projets (dont des langages de programmation comme Python, Go et Rust) ont adopté un modèle de décision pour les **modifications importantes** au code (ou à la gouvernance) reposant sur des **discussions préalables sur des documents**, PEP (Python Enhancement Proposals) ou RFC (Request For Comments) :

- **Un projet de modification est présenté** avec des motivations, des explications sur les choix effectués, les alternatives et les conséquences qui ont été considérées.
- Une **discussion** a lieu sur les forums / mailing lists qui peut conduire à la **mise à jour du document**.
- Un processus de décision (ex. recherche d'un consensus paresseux, validation par l'équipe cœur, etc.) permet d'**entériner la proposition**, qui pourra ensuite être implémentée par des volontaires.

Méritocratie/Do-ocracy : un système parfait ?

Quels sont les problèmes posés par les systèmes de do-ocracy ? de BDFL ?

Rôles et parcours de contributeurs

Pourquoi contribuer à des logiciels libres ?

Pourquoi contribuer à des logiciels libres ?



Motivations des contributeurs open source (le bleu/vert représente l'accord, le jaune le désaccord). Extrait de Gerosa et al. (2021).

Contributeurs occasionnels

Beaucoup de contributions se font par des contributeurs **occasionnels** (drive-by)

- Utiliser un outil
- Constater un bug
- Récupérer le source
- Corriger le bug
- Faire une MR/PR
- La faire accepter

Contributeurs réguliers et mainteneurs

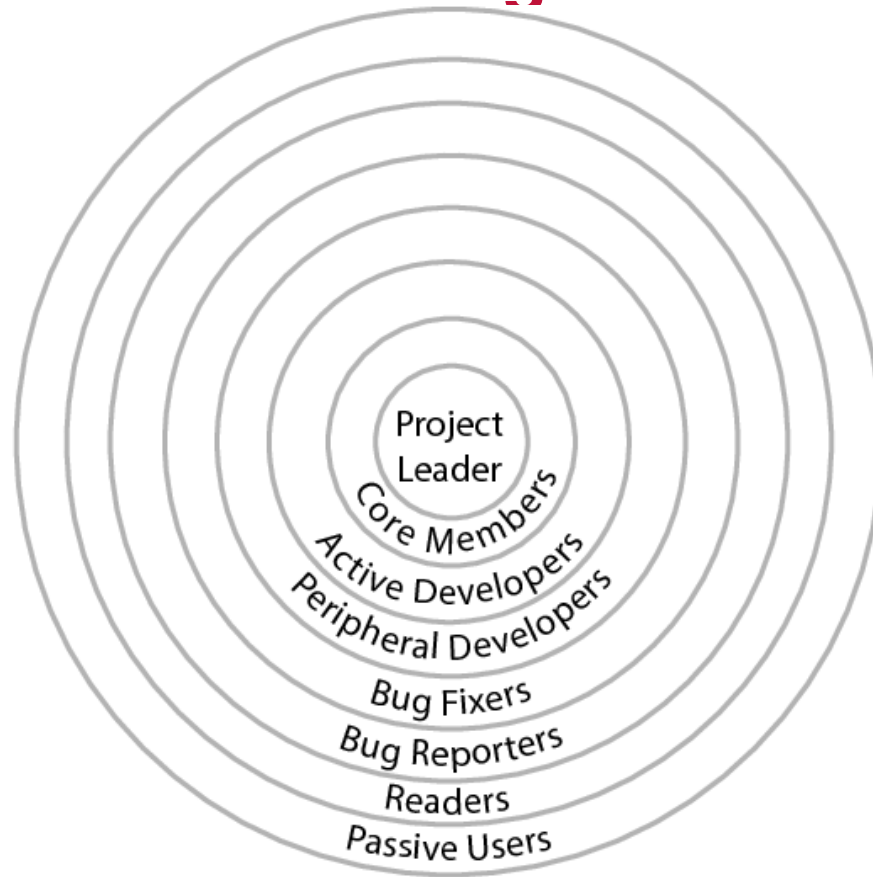
Les contributeurs *réguliers* sur un projet ont en général plus d'impact long-terme sur

- L'évolution future du projet (*roadmap*)
- La maintenance
- La gestion de la communauté et l'accueil des contributeurs

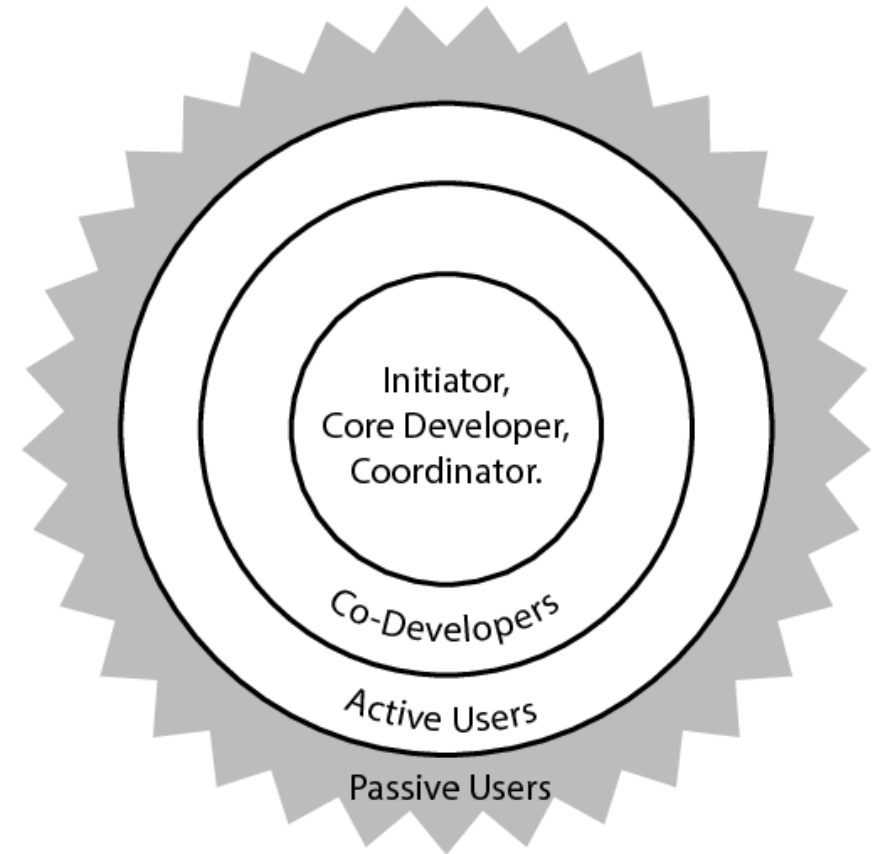
Un *mainteneur* est un contributeur régulier qui partage la charge de la gestion du projet et prend des décisions.

Rôles et parcours de contributeurs

Structure en oignon



Seven layer community Structure.
From Nakakoji et al (2002).



Four layer team structure.
From Crowston and Howison (2005)