Cours Logiciels Libres

Processus de contribution (suite)

Théo Zimmermann

Université de Paris & Inria

Vendredi 11 février 2022

Pourquoi contribuer à des logiciels libres ?

Pourquoi contribuer à des logiciels libres ?

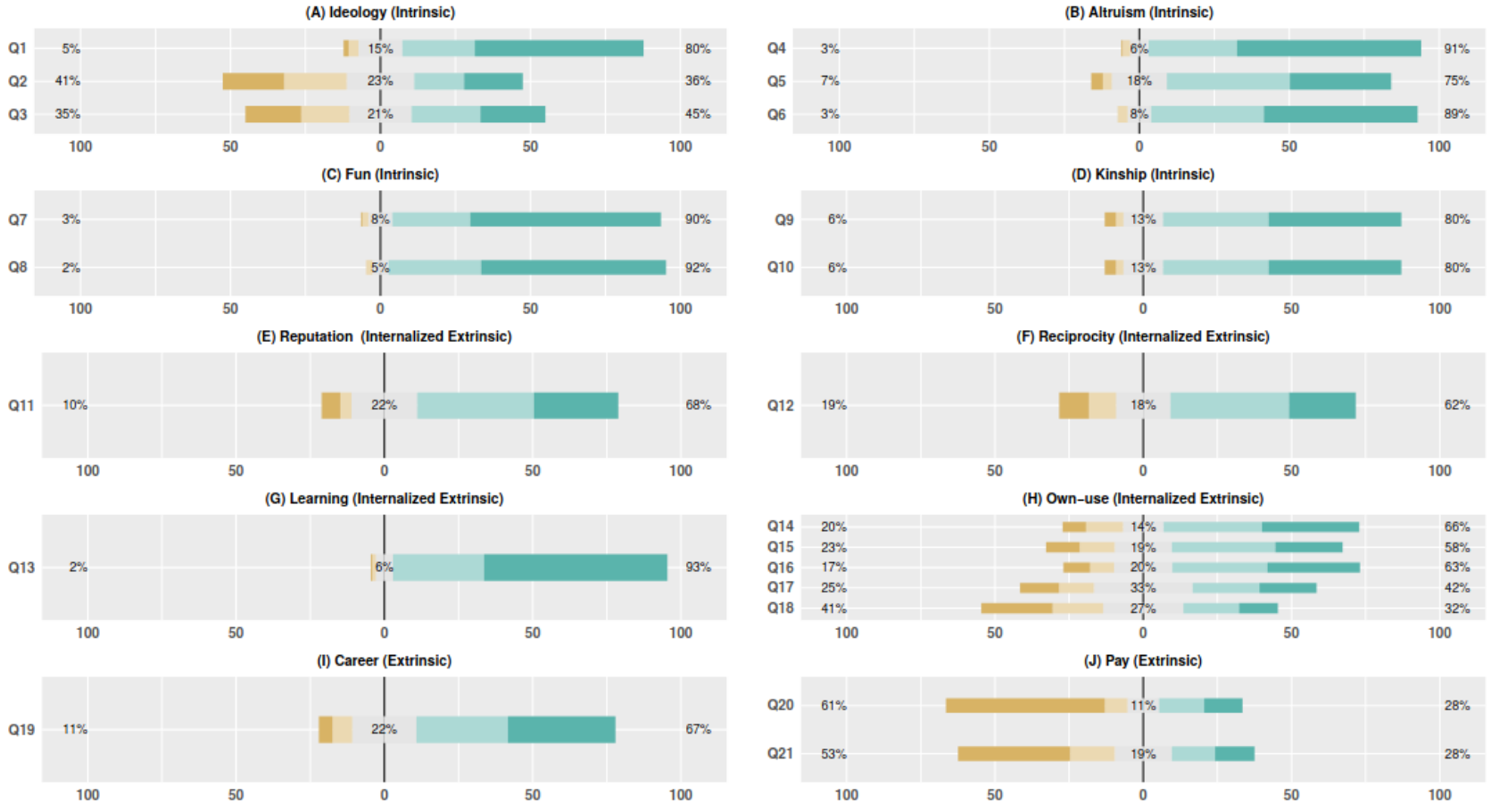

Figure 1: Motivations des contributeurs open source (le bleu/vert représente l'accord, le jaune le désaccord). Extrait de Gerosa et al. (2021).

Avant de contribuer

Se poser quelques questions :

- Le projet est-il **libre** ? (Sous quelle licence ?)
- Accepte-t-il les contributions ?
- Est-il **actif** ? (De quand date le dernier commit, le dernier ticket, les tickets sont-ils traités ?)

Comment contribuer ?

Commencer par **comprendre et respecter** les règles (écrites et non écrites).

- Lire la documentation : licence (`LICENSE`), `README`, “guide de contribution” (`CONTRIBUTING`), “code de conduite” (`CODE_OF_CONDUCT`), etc.
- Se familiariser avec le projet (naviguer dans le système de tickets, sur les forums, les chats, s’inscrire à l’activité).

- Règles génériques : être poli, **ne pas s’impatienter** (les mainteneurs du projet ne vous doivent rien, pas même de l’attention), répondre aux questions calmement, vérifier la documentation avant de poser des questions. . .

Comment contribuer: un rapport de bug ?

Si vous découvrez un bug en utilisant un logiciel libre, vous pouvez le rapporter aux mainteneurs du logiciel.

Comment contribuer: un rapport de bug ?

Si vous découvrez un bug en utilisant un logiciel libre, vous pouvez le rapporter aux mainteneurs du logiciel.

1. Chercher le système de tickets / *bug tracker*.

Comment contribuer: un rapport de bug ?

Si vous découvrez un bug en utilisant un logiciel libre, vous pouvez le rapporter aux mainteneurs du logiciel.

1. Chercher le système de tickets / *bug tracker*.
2. Chercher avec différents mots-clés si le bug a déjà été rapporté. Si c'est le cas, vous pouvez éventuellement y ajouter des informations nouvelles qui aident à le reproduire ou débugger.

Comment contribuer: un rapport de bug ?

Si vous découvrez un bug en utilisant un logiciel libre, vous pouvez le rapporter aux mainteneurs du logiciel.

1. Chercher le système de tickets / *bug tracker*.
2. Chercher avec différents mots-clés si le bug a déjà été rapporté. Si c'est le cas, vous pouvez éventuellement y ajouter des informations nouvelles qui aident à le reproduire ou débugger.
3. Si vous n'avez pas trouvé de ticket similaire, vous pouvez en créer un nouveau. Pensez à bien indiquer toute information utile pour reproduire le bug :
   - **version** du logiciel,
   - comment vous avez obtenu / compilé le paquet,
   - quel est votre **environnement** (système d'exploitation, navigateur. . .),
   - séquence d'étapes précise / fichier permettant de **reproduire le problème** (le plus court possible).

Comment contribuer: un rapport de bug ?

Si vous découvrez un bug en utilisant un logiciel libre, vous pouvez le rapporter aux mainteneurs du logiciel.

1. Chercher le système de tickets / *bug tracker*.
2. Chercher avec différents mots-clés si le bug a déjà été rapporté. Si c'est le cas, vous pouvez éventuellement y ajouter des informations nouvelles qui aident à le reproduire ou débugger.
3. Si vous n'avez pas trouvé de ticket similaire, vous pouvez en créer un nouveau. Pensez à bien indiquer toute information utile pour reproduire le bug :
   - **version** du logiciel,
   - comment vous avez obtenu / compilé le paquet,
   - quel est votre **environnement** (système d'exploitation, navigateur...),
   - séquence d'étapes précise / fichier permettant de **reproduire le problème** (le plus court possible).
4. Pensez à répondre aux questions des mainteneurs.

Comment contribuer : du code ?

Important : penser à **communiquer** avant de se lancer dans le code.

Pour soumettre un patch, cela dépend du projet :

- Linux : `git format-patch` envoyé par e-mail.
- Patch en pièce attachée sur un bug tracker.
- **Pull request** sur GitHub / GitLab (le plus commun pour les projets récents, mais aussi certains projets anciens).

Pour savoir : trouver et **lire la documentation** (“contributing guide”).

Les patchs

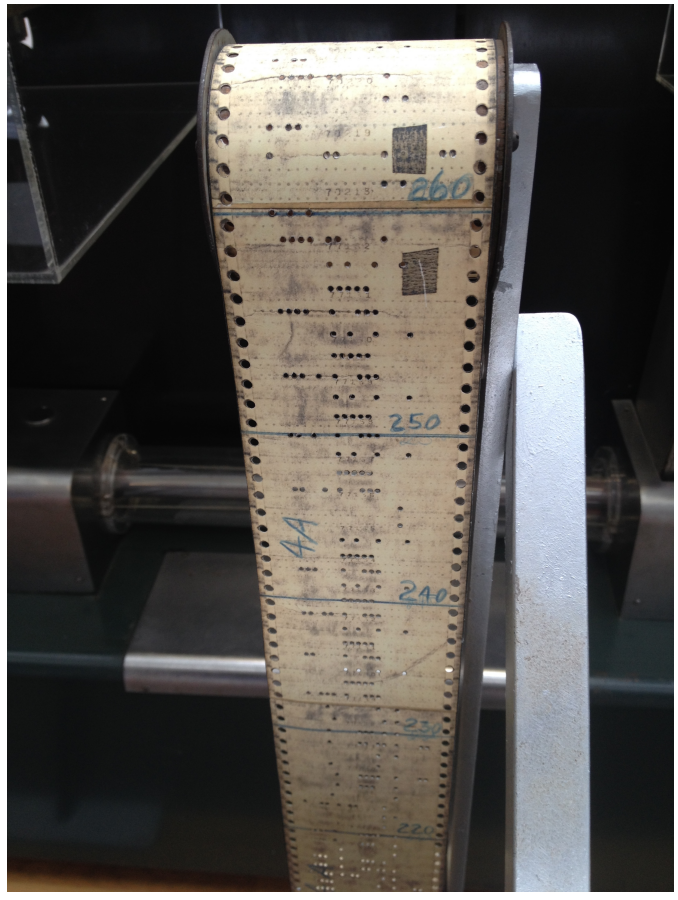

Figure 2: Le mot patch trouverait son origine dans la manière dont on corrigeait les bugs dans les programmes sur ruban perforé.

Les patchs

- Qu'est-ce qu'un patch ? C'est un fichier indiquant **comment modifier** un programme existant. Il est donc bien **plus léger** de distribuer un patch qu'une copie modifiée du programme.
- Lorsque le code source du programme est disponible, le patch est généralement distribué sous forme de **diff** (lignes ajoutées, lignes supprimées).
- Unix comportait déjà les commandes `diff` (pour créer un patch) que les éditeurs de l'époque (ou la commande `patch`) pouvaient ensuite **appliquer** sur un fichier source.
- Les **gestionnaires de versions** (des tout premiers dans les années 70 à git aujourd'hui) enregistrent l'histoire d'un fichier sous forme de diffs.

Partage de patchs

- Avant l'invention des "logiciels propriétaires" et des "logiciels libres", les utilisateurs avaient le plus souvent accès au code source des programmes, et pouvaient donc les modifier et **partager leurs modifications sous forme de patchs** avec leurs collègues.
- Dans le cas des logiciels libres, la redistribution de versions modifiées est autorisée, mais pour contribuer à un logiciel ou distribuer un correctif, il est **plus simple** de partager des patchs (ils sont plus légers et permettent de **voir ce qui a changé**).
- Le mainteneur d'un logiciel libre peut donc **recevoir des patchs et choisir de les appliquer** (tels quels ou modifiés) dans les nouvelles versions du logiciel.
- D'après Eric Raymond, c'est la **principale activité d'un mainteneur open source** (en particulier de Linus Torvalds).

Gestionnaires de versions et dépôts publics

- Tout bon développeur sait qu'il vaut mieux utiliser un gestionnaire de versions pour permettre de retracer l'historique du code (cela vaut pour les logiciels libres et les logiciels propriétaires).
- Les gestionnaires de versions existent depuis **longtemps** : SCCS (1972), RCS (1982), CVS (1990), SVN (2000), Git (2005).
- Internet et les gestionnaires de versions permettent à partir des années 90 de partager un logiciel libre dans un **dépôt public**. Cela veut dire que les utilisateurs ont accès à la toute dernière **version de développement** en permanence. Ils peuvent utiliser le gestionnaire de versions pour préparer leurs patchs.
- Avec les gestionnaires centralisés, les patchs restent le **seul moyen de contribuer** pour ceux qui n'ont pas les droits de commit sur le dépôt.

Les gestionnaires de versions décentralisés

- Symmétrie entre les différents dépôts : on peut **tirer** des changements de n'importe quel dépôt public.
- Donc on peut pousser des changements sur une **branche** publique, et proposer au mainteneur de les tirer (*pull request*).
- Cependant, le projet **Linux n'utilise pas les pull requests** pour les contributions. L'aspect distribué est utilisé pour la maintenance (**maintenance hiérarchique** avec un dépôt par mainteneur).
- C'est GitHub qui **popularise le modèle des pull requests** pour les contributions aux logiciels libres.

Le modèle de Linux

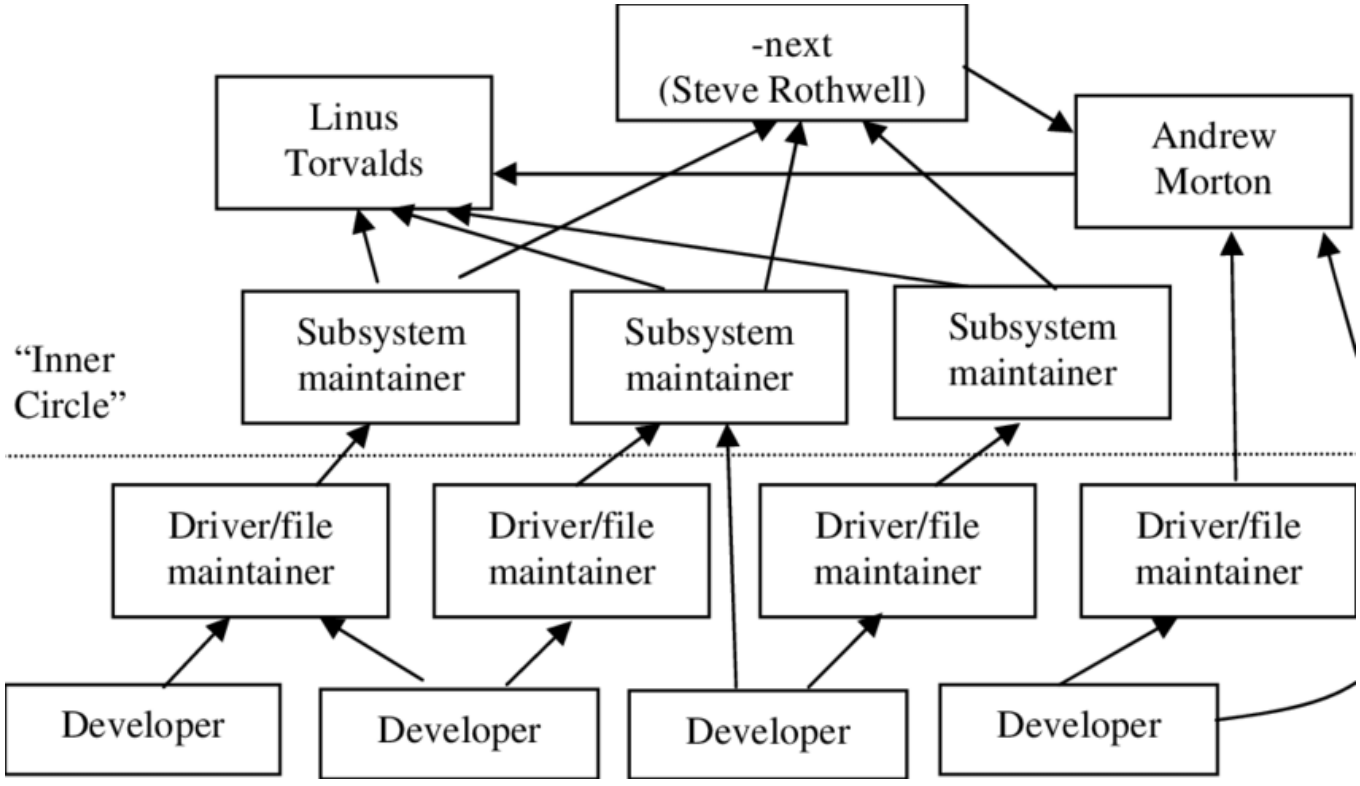

Figure 3: Extrait de Xia et al. (2010) "Exploring the knowledge creating communities: an analysis of the Linux Kernel developer community"

Les pull requests

1. Cliquer sur le bouton “Fork” sur GitHub / GitLab.

2. Cloner le dépôt (votre fork).

```
$ git clone git@github.com:mon-compte/le-projet
```

3. Créer une nouvelle branche (**important !**) :

```
$ git checkout -b ma-branche
```

Les pull requests

4. Coder... (il peut être utile de **lire la documentation**, de poser des questions sur le ticket si besoin, etc.)
5. Lancer les tests en local si possible.
6. Commit (voir la doc sur le format de message) et push :

```
$ git add mes fichiers modifies
$ git commit -m "Mon message de commit.

Des informations supplémentaires sur
plusieurs lignes."
$ git push -u origin ma-branche
```

Les pull requests

7. Ouvrir la pull request (par exemple en cliquant sur le lien que GitHub donne au moment de pousser, ou en naviguant sur le dépôt du projet).
8. Écrire un message expliquant les modifications et les tickets qu'elle ferme.

Les pull requests

9. Prendre en compte les retours sur la pull request :
   - Retours automatiques de l'intégration continue et de bots éventuels.
   - Commentaires des mainteneurs.

   (Si besoin) modifier le code dans la branche, commit, push.

   Selon les attentes du projet, ça peut devenir plus compliqué et apprendre à bien maitriser git peut être utile.

Quelques écueils à éviter

- Ne pas **être disponible pour répondre** aux questions, prendre en compte les suggestions.
- Les **trop gros** changements : plus votre contribution est grosse, plus la revue de code est difficile, et donc plus elle prendra du temps à être intégrée.
- Mélanger des changements **indépendants** : les mainteneurs préfèrent éviter cela en général. De plus, plusieurs contributions plus petites sont plus rapides à relire qu'une seule trop grosse.
- Les fonctionnalités dont la **conception** n'est pas arrêtée. Il vaut mieux mélanger le moins possible revue du design et revue du code. Pour discuter du design, **un ticket suffit**.

Aider à la revue de code

Les projets open source peuvent apprécier que les contributeurs participent à la revue de code même s'ils ne sont pas mainteneurs.

Avant de le faire :

- Vérifier que c'est quelque chose que les mainteneurs du projet apprécient (ou même suggèrent).
- Déterminer quelles sont les attentes pour les *reviewers*.

À quoi sert la revue de code ?

Aider à la revue de code

Les projets open source peuvent apprécier que les contributeurs participent à la revue de code même s'ils ne sont pas mainteneurs.

Avant de le faire :

- Vérifier que c'est quelque chose que les mainteneurs du projet apprécient (ou même suggèrent).
- Déterminer quelles sont les attentes pour les *reviewers*.

À quoi sert la revue de code ?

- Garantir un certain niveau de **qualité** du code **nouvellement introduit** ou modifié. Est-il facile à comprendre ? Si oui, il y a moins de risques qu'il introduise des bugs.
- S'assurer que certains **standards** sont respectés (si cela n'est pas possible automatiquement) : par exemple, ajout (et clarté) de la documentation.

Devenir mainteneur

- La plupart des projets n'ont **pas de processus établi** pour devenir *committer* / mainteneur.
- Les nouveaux mainteneurs sont généralement **co-optés parmi les contributeurs réguliers** les plus actifs.
- Dans les plus gros projets, les mainteneurs peuvent n'être responsables que d'une partie du code (*code owners*).

Système de code owners

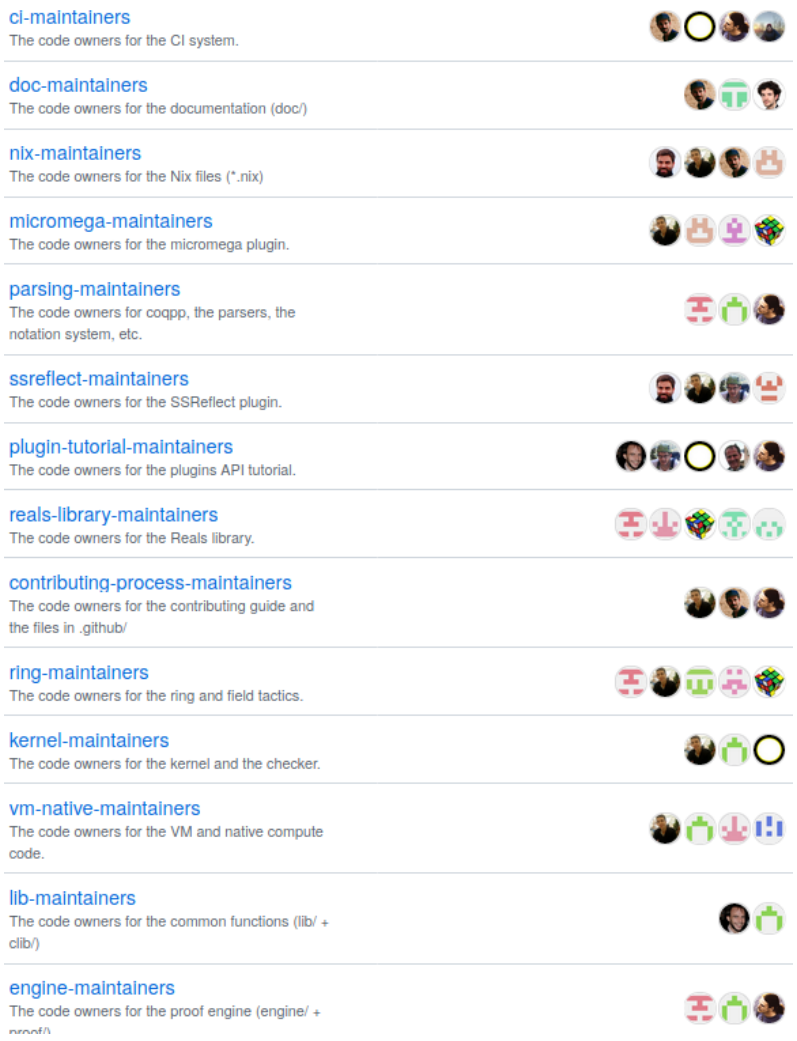

Figure 4: Exemple dans le projet Coq