Université Paris-Cité—M1 Informatique Année 2021–2022

Introduction aux Logiciels Libres – TD n° 4

Gouvernance de projets open source

Exercice 1 – Droit au fork Dans le chapitre 4 du livre “Producing open source software”, Karl Fogel écrit :

> The indispensable ingredient that binds developers together on a free software project, and makes them willing to compromise when necessary, is the code's forkability : the ability of anyone to take a copy of the source code and use it to start a competing project, known as a fork.
>
> The paradoxical thing is that the possibility of forks is usually a much greater force in free software projects than actual forks are. Actual forks are very rare. Because a fork is usually bad for everyone, the more serious the threat of a fork becomes, the more willing people are to compromise to avoid it.
>
> The potential for forks is the reason there are no true dictators in free software projects. This may seem like a surprising claim, considering how common it is to hear someone called the “dictator” (sometimes softened to “benevolent dictator”) in a given open source project. But this kind of dictatorship is special, quite different from our conventional understanding of the word. Imagine a ruler whose subjects could copy her entire territory at any time and move to the copy to rule as they see fit. Would not such a ruler govern very differently from one whose subjects were bound to stay under her rule no matter what she did ?

1. Quel élément fait que les projets open source peuvent être forkés à tout moment ?
2. Pourquoi Karl Fogel dit-il qu'il n'y a pas de vrais dictateurs dans les projets open source ?
3. Pourquoi les forks sont-ils en fait assez rares ?

Exercice 2 – Consensus paresseux Dans le modèle de gouvernance méritocratique de l'OSS Watch de l'université d'Oxford, le consensus paresseux est expliqué de la manière suivante :

> This is a meritocratic, consensus-based community project. [...]
>
> The PMC makes decisions when community consensus cannot be reached. [...]
>
> Since most people in the project community have a shared vision, there is often little need for discussion in order to reach consensus. In general, as long as nobody explicitly opposes a proposal or patch, it is recognised as having the support of the community. This is called lazy consensus—that is, those who have not stated their opinion explicitly have implicitly agreed to the implementation of the proposal.
>
> Lazy consensus is a very important concept within the project. It is this process that allows a large group of people to efficiently reach consensus, as someone with no objections to a proposal need not spend time stating their position, and others need not spend time reading such mails.

1. Quel est le mode préféré de prise de décision dans les projets open source ?

2. Pourquoi le consensus paresseux est-il considéré comme un mode de décision efficace ? Quels seraient les problèmes associés à un autre mode de décision (consensus explicite, vote systématique, etc.) ?
3. Comment les projets open source peuvent-ils gérer les cas où le consensus ne parvient pas à être atteint ?

Exercice 3 – Dictateurs bienveillants Dans son chapitre 4, Karl Fogel donne la description suivante du rôle du dictateur bienveillant :

> Only when it is clear that no consensus can be reached, and that most of the group wants someone to make a decision so that development can move on, does she put her foot down and say "This is the way it's going to be." Reluctance to make decisions by fiat is a trait shared by almost all successful benevolent dictators ; it is one of the reasons they manage to keep the role.

Par ailleurs, le document décrivant la gouvernance du projet Ubuntu contient la justification suivante du rôle du dictateur :

> The community functions best when it can reach broad consensus about a way forward. However, it is not uncommon in the open-source world for there to be multiple good arguments, no clear consensus, and for open questions to divide communities rather than enrich them. The debate absorbs the energy that might otherwise have gone towards the creation of a solution. In many cases, there is no one 'right' answer, and what is needed is a decision more than a debate. The SABDFL (self-appointed benevolent dictator for life) acts to provide clear leadership on difficult issues, and set the pace for the project.

Enfin, le document sur la gouvernance dictatoriale de l'OSS Watch contient le texte suivant :

> The project lead's role is a difficult one : they set the strategic objectives of the project and communicate these clearly to the community. They also have to understand the community as a whole and strive to satisfy as many conflicting needs as possible, while ensuring that the project survives in the long term.

1. Le dictateur est-il celui ou celle qui prend la plupart des décisions ?
2. En quoi le dictateur est-il utile au projet open source ?
3. En quoi le rôle du dictateur est-il difficile ?
4. À votre avis, pourquoi avec le temps beaucoup de projets open source abandonnent le modèle de gouvernance dictatoriale ? Quelles alternatives choisissent-ils alors ?