Introduction aux Logiciels Libres – TD n° 9

Écosystèmes de logiciels libres

Exercice 1 – Licences et dépendances

1. Si ma bibliothèque *contient* du code sous licence GNU GPL, ai-je le droit de la publier sous une licence permissive (type MIT) ?
2. Si ma bibliothèque *dépend* d'une bibliothèque sous licence GNU GPL, ai-je le droit de la publier sous une licence permissive ?
3. Si je dépends de manière transitive d'une bibliothèque sous licence GNU GPL, sa licence s'applique-t-elle à mon application ?
4. Quels sont les risques à publier une bibliothèque sous licence permissive si elle dépend d'une bibliothèque sous licence GPL ? Dans quels cas cela peut être considéré comme utile ?

Exercice 2 – Dependency hell Le fichier MAINTAINERS du dépôt `https://github.com/commercialhaskell/stackage` présente les motivations du projet Stackage :

> This project is built around the concept of maintainers taking responsibility for making their packages work with the rest of the stable ecosystem, usually meaning the newest version of all dependencies. This is a social contract, and is not reflected in the codebase in any way.
>
> The idea behind Stackage is that, if all packages work with the newest versions of dependencies, we avoid dependency hell.

1. Que veulent dire les auteurs par "dependency hell" ?
2. Sur qui le projet Stackage compte-t-il pour éviter cette situation ?
3. Comment le problème est-il évité ?

Exercice 3 – Organisations communautaires de maintenance de paquets Le README du dépôt `https://github.com/dlang-community/discussions` présente les motivations de l'organisation Dlang-community :

> DLang-community is a GitHub organization which maintains D packages that are important to the D ecosystem.
>
> This organization was formed by annoyance of needing to fork popular repositories to get fixes merged. [...] projects are still driven by their original authors if they have the time. However, small bug fixes don't need to wait in the queue for months, and in case the author is completely gone, the DLang community has one upstream repository instead of ten different forks containing the same fix. Moreover, thanks to being a larger organization, the overhead of a project can be distributed and more easily automated (e.g. documentation builds, binary releases etc.).

1. Quel est le problème que les fondateurs de cette organisation ont rencontré et qui les a conduit à la créer ?
2. Quelle solution proposent-ils ?
3. Quels sont les autres avantages qu'ils entrevoient ?

Exercice 4 – Changements cassants Voici l'abstract de l'article "Breakbot : Analyzing the Impact of Breaking Changes to Assist Library Evolution" par Lina Ochoa *et al.* :

> "If we make this change to our code, how will it impact our clients ?" It is difficult for library maintainers to answer this simple—yet essential !—question when evolving their libraries. Library maintainers are constantly balancing between two opposing positions : make changes at the risk of breaking some of their clients, or avoid changes and maintain compatibility at the cost of immobility and growing technical debt. We argue that the lack of objective usage data and tool support leaves maintainers with their own subjective perception of their community to make these decisions. We introduce BreakBot, a bot that analyses the pull requests of Java libraries on GitHub to identify the breaking changes they introduce and their impact on client projects. Through static analysis of libraries and clients, it extracts and summarizes objective data that enrich the code review process by providing maintainers with the appropriate information to decide whether—and how—changes should be accepted, directly in the pull requests.

1. Quel dilemme doivent résoudre les mainteneurs de bibliothèques ? Quels sont les inconvénients de chacune des deux options qui s'offrent à eux ?
2. De quoi ont besoin les mainteneurs pour prendre une meilleure décision, selon les auteurs ? Quelle est la solution qu'ils proposent pour les aider dans cette tâche ?