D’un projet à l’autre, franchissez les frontières (Libres conseils 11/42)

Chaque jeudi à 21h, rendez-vous sur le framapad de traduction, le travail collaboratif sera ensuite publié ici même.

Traduction Framalang : ga3lig, lenod, peupleLa, LAuX, billouche, Goofy, jcr83, purplepsycho, Jej, KoS, Julius22, kalupa, tuki, lamessen, okram + Sinma

La collaboration entre projets

Henri Bergius

Henri Bergius est le fondateur de Midgard(1), un dépôt de contenu pour les logiciels libres. Il a aussi été longtemps impliqué dans la géolocalisation d’ordinateurs de bureaux sous Linux ainsi que dans les communautés Maemo et Meego. Il dirige un petit cabinet de conseil nommé Nemein, bidouille CoffeeScript et PHP et passe beaucoup de son temps à faire de la moto dans des régions reculées du continent Eurasien. Il vit dans la froide ville nordique d’Helsinki, en Finlande.

> *Il se peut qu’il existe un système complètement nouveau dans lequel vous pouvez être défini davantage par qui vous êtes plutôt que par ce que vous possédez, par ce que vous avez créé et partagé, parce que d’autres personnes ont ensuite construit sur cette base*
>
> *– John Seely Brown, ancien directeur de Xerox PARC dans An Optimist’s Tour of the Future (Mark Stevenson, 2010)*

Le monde du logiciel libre est pour l’essentiel divisé en tribus rassemblées autour de choses appelées projets. Il existe des projets majeurs tels que GNOME(2), KDE(3) ou Drupal(4) et il existe bien d’autres projets plus modestes tournant autour d’une seule application ou bibliothèque logicielle.

En fait, les qualifier de « projets » est un peu ridicule.

Selon moi, un projet est l'organisation d'un effort visant un but que l'on puisse atteindre et comprend un calendrier avec dates de début et de fin. Ainsi, GNOME 3.1 serait par exemple un projet tandis que GNOME, pris dans son ensemble, n'en est pas un. Il s'agit d'une communauté d'individus qui entretiennent et créent le corps d'un logiciel par de petits efforts variés ou des projets.

Assez de pédantisme. Le souci avec le concept de projet c'est qu'il finit par maintenir une séparation entre les personnes. Cela crée des communautés isolées souvent réticentes voire incapables de collaborer avec « la concurrence ». Mais en fin de compte, toutes ces communautés sont composées de personnes écrivant des logiciels libres. Et ce sont elles qui décident de l'utilisation possible ou non d'un logiciel dans différents environnements.

En fin de compte, nous voulons tous que le logiciel que nous avons créé soit utilisé par d'autres. Mieux encore : nous voulons que les autres joignent leurs efforts aux nôtres et créent des choses sympa à partir de ce que nous avons créé. Après tout, ceci constitue le cœur même des logiciels libres.

Alors pourquoi érigeons-nous ces murs autour de nous ? Garder une communauté isolée ne fait que favoriser une mentalité de type « nous contre eux ». Les incompatibilités des différents langages de programmation contribuent déjà fortement à notre division. Pourquoi en rajouter ?

La leçon de Midgard

Il est une chose que j'aurais voulu savoir quand j'ai démarré, dans cette période optimiste des « .com » de la fin des années 90 : c'est qu'en réalité le développement de logiciels ne gagne rien à s'isoler. Avec un peu d'efforts nous pouvons partager nos logiciels et nos idées par le biais de communautés, ce qui renforce et améliore à la fois les logiciels et les communautés.

Quand j'ai démarré ma carrière dans le logiciel libre, c'était l'époque des grands projets. Netscape ouvrait son code source, la fondation Apache prenait forme et des sociétés de capital-risque venaient de partout. Tenter de bâtir sa communauté devenait la norme. C'était le chemin assuré vers la gloire, la fortune et la réalisation de choses extraordinaires.

Alors nous avons construit nos propres infrastructures web. À ce moment-là il n'y en avait pas tant que cela, en particulier pour le tout jeune langage PHP. Le PHP n'était même pas notre premier choix. Nous l'avions seulement choisi au terme d'un long débat sur l'utilisation de Scheme(5) que notre développeur principal préférait. Mais le PHP gagnait alors en popularité, devenant le langage de programmation de la Toile. Et nous voulions construire la Toile.

Au début, les choses semblaient prometteuses. Beaucoup de développeurs rejoignaient notre communauté et commençaient à y contribuer. Il y a même eu des entreprises fondées autour de Midgard. Notre infrastructure gagnait en fonctionnalités et devenait de plus en plus liée à Midgard.

Avec le recul, c'est là que nous avons fait une erreur. Nous avons positionné Midgard pour être distinct du PHP lui-même. Quelque chose que vous installeriez séparément, et utiliseriez comme base pour y construire des sites entiers. Il fallait soit suivre notre voie, soit suivre celle de tout le monde.

Avec Midgard, vous deviez utiliser notre interface de dépôt de contenus pour tout, aussi bien pour notre gestion des utilisateurs que pour le modèle de permissions. Vous deviez utiliser notre système de modèles et stocker beaucoup de votre code dans le dépôt au lieu d'utiliser un système de fichiers.

Ceci ne passait évidemment pas très bien auprès de l'ensemble de la communauté PHP. Nos idées leur semblaient étranges, et Midgard, à ce moment-là, était même distribué en tant que gigantesque correctif à la base de code puisqu'on ne pouvait pas charger de modules avec PHP3.

Les années ont passé et la popularité de PHP a connu des hauts et des bas. Pendant ce temps, la communauté Midgard est restée relativement constante : un petit groupe soudé faisant des progrès sur le long terme mais séparé du monde plus large de PHP.

Nous nous sommes toujours demandé pourquoi il était si difficile d'interagir avec le monde PHP. Même d'autres communautés faisant des choses complètement différentes, comme l'environnement de bureau GNOME, semblaient plus faciles à approcher. Ce n'est que récemment, après des années d'isolement, que nous avons pris conscience du problème. En résumé : les infrastructures nous séparent alors que les bibliothèques nous permettent de partager notre code et nos expériences.

À propos des bibliothèques et des infrastructures

En définitive, les logiciels ont pour objectif l'automatisation, la construction d'outils que les autres peuvent utiliser pour résoudre des problèmes ou se connecter entre eux. Avec les logiciels, ces outils comportent plusieurs couches. Il existe des services de bas niveau comme les systèmes d'exploitation, puis les bibliothèques, les infrastructures, les boîtes à outils et enfin les applications elles-mêmes.

Les applications sont toujours écrites pour des usages spécifiques, donc entre elles il existe peu de possibilités de partage de code.

Les possibilités les plus séduisantes se situent au niveau des bibliothèques et infrastructures. Une infrastructure, si elle est suffisamment générique, peut généralement être utilisée pour construire différentes sortes de logiciels. Une bibliothèque, quant à elle, peut être utilisée pour apporter un élément particulier de logique ou de connectivité là où le besoin s'en fait sentir. De mon point de vue, c'est dans cette couche que le plus gros de la programmation devrait être fait, avec des applications spécifiques qui ne sont que des connexions entre diverses bibliothèques au sein d'une infrastructure qui s'occupe alors de faire tourner l'application en question.

Qu'est-ce qu'une bibliothèque et qu'est-ce qu'une infrastructure ? Les gens utilisent souvent ces termes indifféremment bien qu'il existe une règle grossière qui permet de les différencier : une bibliothèque est une ressource à laquelle votre code fait appel, alors qu'une infrastructure est une ressource qui fait appel à votre code.

Si vous voulez que votre code soit utilisé et amélioré, le meilleur moyen est de maximiser le nombre de ses utilisateurs et contributeurs potentiels. Avec le logiciel libre, cela fonctionne en s'assurant que votre code peut être adapté à de multiples situations et environnements.

En définitive, ce que vous voulez développer c'est une bibliothèque. Les bibliothèques c'est cool.

Comment faire en sorte que la collaboration fonctionne

Le plus difficile est de franchir la barrière du « eux-contre-nous ». Les développeurs de l'autre communauté sont des bidouilleurs concevant du logiciel libre, tout comme vous. Il suffit donc de franchir le pas et de commencer à leur parler.

Une fois le débat engagé, voici quelques points que j'ai trouvés importants quant à l'application effective des idées communes ou des bibliothèques au-delà des frontières du projet

- Utilisez des licences permissives et essayez d'éviter les cessions de droits d'auteurs et autres exigences que les utilisateurs potentiels trouveraient onéreuses. Hébergez le projet en terrain neutre. Pour les projets web, Apache est un assez bon havre. Pour les projets bureautiques, Freedesktop est probablement le meilleur choix. Utilisez des technologies qui n'imposent pas trop de contraintes. Les bibliothèques doivent être de

bas niveau, ou fournir des API (interfaces de programmation) D-Bus utilisables avec n'importe quel système.

- Évitez les dépendances spécifiques à une infrastructure. KDE a, par exemple, trouvé GeoClue difficile à adopter parce qu'il utilise des paramètres spécifiques à l'interface GNOME. Rencontrez les autres. Si vous venez du projet GNOME, allez à l'aKademy et donnez-y une conférence. Si vous êtes développeur KDE, allez parler au GUADEC. Après avoir partagé une bière ou deux, la collaboration par IRC vient beaucoup plus naturellement.
- Enfin, vous devez accepter que votre implémentation ne soit pas utilisée par tout le monde. Mais si, au moins, d'autres mettent en œuvre les mêmes idées, alors une collaboration reste possible.

Bonne chance pour abattre les frontières du projet ! Dans la plupart des cas, cela fonctionne si vos idées sont bonnes et présentées avec un esprit ouvert. Mais même si vous ne trouvez pas de terrain d'entente, tant que votre application remplit sa fonction pour vous, ça n'a pas été fait en vain. Après tout, ce qui compte c'est de proposer des logiciels et d'offrir la meilleure expérience utilisateur possible.

(1) http://midgard-project.org/

(2) gnomefr.org

(3) fr.kde.org

(4) drupalfr.org

(5) http://fr.wikipedia.org/wiki/Scheme