6 questions à Karl Fogel, auteur de Produire du logiciel libre

À l'occasion de la sortie du framabook **Produire du logiciel libre** (dont notre secret espoir est qu'il suscite des vocations chez les jeunes et les moins jeunes), nous avons posé quelques questions à son auteur Karl Fogel.

Est-ce que la situation a évolué depuis la première version du livre, en particulier avec les nouvelles forges comme GitHub (qui repose entre autres la question du fork) ? Est-ce un problème d'héberger des logiciels libres sur des plateformes propriétaires ? Est-ce que l'informatique devrait être enseignée en tant que telle aujourd'hui à l'école ?

Autant de questions auxquelles il apporte de très intéressantes réponses.

Entretien avec Karl Fogel

L'interview en version originale anglaise sur le blog de Karl (intéressants commentaires inside)

(Traduction Framalang : Don Rico pour les questions et Olivier Rosseler pour les réponses)

La version française de POSS vient tout juste d'être publié et votre livre a été traduit, ou est en cours de traduction, dans d'autres langues. Que pensez-vous de ces adaptations de votre œuvre, rendues possibles par le choix de le placer sous licence libre ?

Je suis absolument ravi. Je n'y vois vraiment aucun inconvénient. Les traductions permettent une diffusion plus large du livre, et c'est exactement ce que je souhaite.

Je suis extrêmement reconnaissant envers les traducteurs.

Si vous deviez écrire une deuxième version de POSS aujourd'hui, qu'est-ce que vous changeriez ou ajouteriez ? Et d'ailleurs, est-ce qu'une deuxième version est prévue ?

Et bien, en fait, j'y apporte toujours des petites modifications, à mesure que les pratiques de l'open source évoluent. La version en ligne change constamment. On pourra peut-être la nommer officiellement « Version 2.0 » à un moment donné, mais au fond, c'est vraiment un processus continu.

Par exemple, il y a cinq ou six ans, presque tous les projets avaient leur propre infrastructure de développement. Chacun avait son serveur, son système de contrôle de versions, son système de suivi de bogues, un responsable de la liste de diffusion, un wiki peut-être, c'étaient les outils de développement.

Mais depuis, on a assisté à des regroupements. De nos jours, seuls les très gros et les très petits projets possèdent leur propre infrastructure. La majorité des projets choisissent des sites pré-conçus, comme GitHub, Google Code Hosting, SourceForce, Launchpad, etc. La plupart des développeurs open source se sont familiarisés avec ces environnements.

Et par conséquent, j'ai mis à jour la partie du livre traitant des infrastructures d'hébergement, pour enrichir la section « Les sites Web » et parler des sites comme ceux mentionnés ci-dessus, plutôt que de ré-inventer la roue à chaque projet. Les gens se rendent bien compte qu'administrer son propre hébergement requiert énormément de ressources, malgré les avantages que l'on peut en tirer, et que donc, externaliser cette tache est devenu presque une obligation si on veut avoir un peu de temps pour effectivement travailler sur le projet.

J'ai également mis le livre a jour pour parler des nouvelles versions des licences open source (comme la GNU General Public License 3, qui est sortie après que le livre ait été publié), et j'ai également revu mes recommandations vis à vis de certains logiciels, car les temps changent. Par exemple, Git est de bien meilleure qualité aujourd'hui qu'à l'époque où j'ai rédigé la toute première édition.

La manière de produire des logiciels libres n'a pas tellement changée en

cinq ans. Mais de nouvelles forges sont apparues, sur un modèle un peu différent de SourceForge. Je pense à Google Code mais surtout à GitHub. GitHub serait un peu le « Facebook des forges open source », avec ses fonctions de réseau social, son édition à même le navigateur… Son slogan est « Fork me on GitHub ». La notion de fork semble ne plus être tout à fait la même qu'avant. Que pensez-vous de tout cela ?

En fait, je pense que la notion de fork n'a pas changé. La terminologie, peut-être, mais pas le concept.

Si je me penche sur les dynamiques des rouages des projets open source, je ne vois pas de differences fondamentales selon que le projet utilise une forge ou l'autre. GitHub propose un produit fantastique, mais ils ont aussi un marketing fantastique. Ils encouragent les projets à inviter leurs utilisateurs à « créer une fork sur GitHub », c'est à dire « créer une copie pour jouer un peu avec ».

Et même si en un sens la copie d'un projet hébergé sur Git peut techniquement s'appeler un « fork », en pratique ça n'en est pas un. Le concept de fork est avant tout politique, pas technique.

À l'origine, initier un fork signifiait élever la voix pour dire : « nous pensons que le projet ne prend pas la bonne direction, nous avons pris la décision d'en faire une copie pour le poursuivre dans la bonne direction, que tout ceux qui partagent ce point de vue se joignent à nous ». Et les deux projets se retrouvaient alors publiquement en concurrence, à l'attention des développeurs et des utilisateurs, parfois aussi pour des questions d'argent. Parfois l'un des deux l'emporte, parfois ils fusionnent pour ne former à nouveau qu'un seul projet. Mais quelle qu'en soit l'issue, c'est avant tout un processus politique : susciter des adhésions pour continuer ensemble le projet.

Cette dynamique est toujours d'actualité, elle se poursuit tous les jours. Qu'on parle de « fork » pour designer quelque chose de différent, pourquoi pas, mais ça ne change pas la réalité, on utilise juste un terme différent pour décrire la réalité.

GitHub a commencé à parler de « fork » pour dire « créer une copie à bidouiller ». Maintenant, c'est vrai qu'avec ce genre de copie il est facile de s'éloigner du projet originel pour re-fusionner plus tard, c'est l'une des caractéristiques de Git et de tous les systèmes de contrôle de

version décentralisé. Et c'est vrai que s'éloigner pour re-fusionner est plus compliqué avec les systèmes de contrôle de version centralisé comme Subversion et CVS. Mais tous ces « forks » créés sur Git ne sont pas des forks au sens premier du terme. En général, lorsqu'un développeur se fait une copie sur Git et la modifie, c'est en espérant que ses changements seront fusionnés dans la copie « maîtresse ». Et quand je dis « maîtresse », ce n'est pas au sens technique, mais bien au sens politique : la copie maîtresse est celle que la plupart des utilisateurs suivent.

Je trouve que ces fonctionnalités de Git et de GitHub sont géniales, et j'aime bien les utiliser, mais il n'y a rien de révolutionnaire ici. Il y a peut-être une évolution de la terminologie, mais la vraie dynamique des projets open source ne varie pas : les développeurs fournissent de gros efforts pour que leurs modifications soient intégrées a la distribution principale, car ils ne veulent pas s'embarrasser avec une copie privée qu'ils auraient a entretenir. Git réduit la pénibilité liée à la maintenance de modifications indépendantes, mais pas encore suffisamment pour que cet effort soit négligeable. Les développeurs intelligents forment des communautés et tentent de conserver un code de base unifié, car c'est la meilleure chose à faire. Ça n'est pas près de changer.

En juin 2010, Benjamin Mako Hill remarque dans son article Free Software Needs Free Tools (traduit ici sur le Framablog) qu'héberger un projet libre sur une plateforme propriétaire pose problème. À votre avis, quelle est l'importance de ce problème ?

Et bien, je connais Mako Hill, je l'apprécie et j'éprouve beaucoup de respect pour lui. Mais je dois dire que je ne partage pas son avis sur ce point, et ce, pour plusieurs raisons.

D'abord, il faut être réaliste. On ne peut pas être un développeur logiciel sans outils propriétaires de nos jours. Réduire arbitrairement la notion de « plateforme » n'est qu'un artifice pour croire qu'on travaille dans un milieu entièrement libre. Par exemple, je peux héberger mon projet chez Launchpad, qui est un logiciel libre, mais est-ce que je peux vraiment écrire du code sans utiliser le moteur de recherche de Google, qui n'est pas libre ? Bien sur que non. Tous les bons programmeurs utilisent en continu Google, ou un autre moteur de recherche propriétaire. Il faut inclure ces recherches Google dans la

« plateforme », impossible de se voiler la face.

Mais on peut pousser la réflexion plus loin :

Qu'attendez-vous de l'hébergeur de votre projet, quelles sont les libertés importantes ? Vous utilisez une plateforme et vous demandez aux autres de l'utiliser aussi pour collaborer avec vous, donc, idéalement, la plateforme devrait être libre.

Ainsi, si vous souhaitez y apporter des modifications, vous pouvez : si quelqu'un veut créer un fork de votre projet (au sens ancien, politique, du terme), ils peuvent reproduire l'infrastructure d'hébergement ailleurs, où ils la contrôleront, si nécessaire. Alors, en théorie tout cela est très bien et très joli, mais honnêtement, même si le code source de Google Code, par exemple, était libre, vous ne pourriez pas reproduire Google Code Hosting. Il vous manquerait encore le personnel, le service, les data center de Google… toute l'infrastructure qui n'a rien à voir avec le code source. Ça n'est pas réalistiquement faisable.

Vous pouvez *forker* le projet, mais en général vous ne pouvez pas reproduire son hébergement, cela demande trop de ressources. Et puisque ça n'est pas votre propre service, vous ne pouvez pas l'adapter a votre convenance ; ce sont les gens qui font tourner les serveurs matériels qui décident de quels ajustements sont acceptables ou pas. Donc dans la pratique, vous ne disposez pas de ces libertés.

(Certains services d'hébergement tentent d'octroyer autant de libertés que possible a leurs utilisateurs. Par exemple, le code de Launchpad est open source, et ils intègrent les correctifs de leurs membres. Mais l'entreprise qui héberge Launchpad doit quand même approuver chaque modification puisque ce sont eux qui font tourner les serveurs. Je crois que SourceForge veut tenter la même expérience, si l'on en croit l'annonce faite récemment à propos d'Allura.)

Alors, en fonction de tout cela, quelles sont les libertés possibles ?

Il vous reste la liberté de faire entrer et sortir vos données. En d'autres termes, le noeud du problème se situe au niveau de la possibilité qu'on les interface de programmations (API pour Application Programming Interfaces) de déplacer les données d'un service à l'autre, de manière fiable et automatique. Si je peux écrire un programme qui peut récupérer toutes les données de mon

projet depuis une forge pour les transférer à une autre, c'est une liberté utile. Je ne suis pas pieds et poings liés. Ça n'est pas la seule liberté qui compte, on est même loin d'une liberté idéale. Mais c'est une liberté utile dont on dispose dans un monde où utiliser ses propres serveurs est devenu inabordable.

Ce n'est pas que cette conclusion m'enchante. Mais les choses sont ainsi. La période de « chasseur/cueilleur » dans l'open source est terminée, nous sommes entrés dans l'ère agricole et urbaine. Vous ne pouvez plus creuser vos propres sillons d'irrigation ou votre propre système d'évacuation des eaux usées. C'est trop compliqué. Mais, au moins, si vous n'êtes pas satisfait du service rendu par un hébergeur, vous pouvez déménager chez un autre plus efficace grâce a la portabilité des données.

Donc ça m'importe assez peu de savoir que la plateforme GitHub est propriétaire, par exemple. Evidemment, ça serait mieux si elle était entièrement open source, mais le fait qu'elle ne le soit pas n'est pas vraiment un énorme problème. Le premier critère auquel je fais attention lorsque j'évalue un service d'hébergement est la richesse de leurs APIs. Est-ce que je peux récupérer toutes mes données si besoin ? Si leurs APIs sont riches, c'est bon signe, ils feront leur travail pour maintenir un service de qualité, car c'est le critère qui leur permettra de conserver leurs utilisateurs.

En France, les élevés de collège et de lycée ne suivent pas de cours d'informatique. Pensez-vous que l'informatique devrait être une matière a part entière, et pas seulement un outil pour les autres matières ?

Evidemment. La compréhension des données et du calcul formel est très importante désormais. C'est une forme d'alphabétisme. Sans aller jusqu'à maîtriser la programmation, il faut savoir comment les données fonctionnent. Cela fait écho à une discussion récente où je me suis rendu compte du gouffre qui peut exister.

J'étais chez le docteur, pour faire quelques tests. L'un d'eux consistait à filmer les battements de mon cœur grâce aux ultra-sons et toute la séquence était enregistrée. C'était incroyable a voir ! Et donc, une fois terminé, je demande à l'accueil si je pouvais avoir les données. Pour être précis, j'ai demandé : « Est-ce que je pourrai avoir les données de l'echocardiogramme ? » L'assistante m'a répondu qu'ils pouvaient m'imprimer

des images basse-résolution. J'ai alors répondu : « Merci, mais ce sont les données que je veux ». Elle m'a répondu que c'est bien ce qu'elle me proposait. Pour elle, le mot « données » n'avait pas la même signification précise que pour ceux qui ont appris ce que sont les données. Ma question impliquait évidemment que je voulais toutes les données qu'ils avaient enregistres. C'est bien ce que signifie « Toutes les données », non ? Il ne devrait pas y avoir de perte d'information : c'est une copie bit par bit. Mais cela ne lui parlait pas. Pour elle, les données, c'est « quelque chose qui ressemble a ce que j'ai demandé ». Je parlais d'information, d'informatique, elle me parlait de perception.

Je suis bien conscient que mon point de vue est radical, mais je trouve que c'est une forme d'illettrisme de nos jours. Vous devez savoir faire la différence entre les vraies informations et les fausses informations et vous devez comprendre l'énorme différence d'application qui existe entre les deux. Si je me rends chez un autre médecin, vous imaginez bien la différence que ça fait si je lui présente la vidéo complète sur clé USB par rapport à des copies basse résolution d'images fixes. L'une est utile, l'autre ne sert strictement à rien.

Les entreprises qui comprennent le mieux la valeur des données, de données nous concernant, ont de plus en plus de moyens d'utiliser ces données à leur avantage, mais pas nécessairement dans le vôtre. Les cours d'informatique sont une forme de défense contre ceci, une réponse immunitaire à un monde dans lequel la possession et la manipulation des données se transforme de plus en plus en pouvoir. Vous êtes mieux à même de comprendre comment les données peuvent être utilisées si vous les avez déjà manipulées vous-même.

Donc oui, je suis pour les cours d'informatique… mais pas seulement comme moyen de défense :-). C'est aussi une formidable occasion pour les écoles de réaliser quelque chose de collaboratif. L'enseignement se focalise trop souvent sur des apprentissages « individuels ». D'ailleurs, la coopération à l'école est souvent prohibée et on appelle cela de la triche. Or en cours d'informatique, la chose la plus naturelle est d'initier des projets open source ou de participer à des projets open source.

Bien sûr, tous les étudiants ne seront pas forcément doués ou hyper motivés pour cela, mais c'est la même choses dans toutes les autres matières. Je pense donc que les cours d'informatique sont une bonne opportunité d'exposer les élèves aux

plaisirs du développement collaboratif. Ces cours devraient avoir un impact incroyable sur certains élèves, comme, par exemple, les cours de musique.

Une toute dernière question : quel conseil donneriez-vous au programmeur en herbe qui souhaite découvrir la communauté des logiciels libres et open source ? Essayez de répondre en une phrase, pas avec un livre entier

Trouvez un projet ouvert que vous appréciez (et, idéalement, que vous utilisez) et commencez à y participer ; vous ne le regretterez pas !