10 règles d’or pour rejoindre les développeurs d’un logiciel libre

Je ne sais si c’est un regret mais je ne suis pas développeur. Du coup ma connaissance du logiciel libre est exogène et non endogène et par là-même inévitablement partielle. C’est pourquoi je suis souvent à l’affût d’informations sur les modes opératoire des projets communautaires libres qui me permettent de combler certaines lacunes et parfaire ma culture en la matière. J’y trouve également des sources d’inspiration pour notre propre projet qui est celui d’animer collectivement le réseau Framasoft avec quelque part le même état d’esprit qu’une communauté de développeurs open source.

Tout ça pour dire que cette nouvelle traduction concerne avant tout ceux qui voudraient rejoindre pour la première fois la communauté d’un logiciel libre mais également tous ceux qui en fins observateurs souhaitent un peu mieux comprendre *comment ça marche* de l’intérieur. Parce qu’effectivement, et Framasoft est là pour en témoigner en aval, force est de constater que le logiciel libre ça marche et même plutôt bien

Nous avons conservé tout le long l’expression *open source* utilisée par l’auteur. Même si parfois sujet à précision voire controverse, elle est ici pour nous pleinement synonyme de *logiciel libre* (*free software*).

Je signale au passage que sur le même thème notre dynamique petite équipe de traduction (baptisée *Framalang*) a entrepris une autre chantier autrement plus ambitieux : traduire le

livre de Karl Fogel Producing Open Source Software dont nous espérons la matérialisation en un joli Framabook dans le courant de l'été[1].

10 règles d'or pour démarrer avec l'open source

10 golden rules for starting with open source

Tobias Schlitt – 19 avril 2007
(Traduction Framalang : Daria, Olivier et Yostral)

Êtes-vous nouveau en open source ? Si ce n'est pas le cas, vous rappelez-vous encore ce que c'était, quand vous avez commencé pour la première fois avec l'open source? J'ai récemment essayé de me rappeler ces jours… c'était en 2001, quand j'ai découvert PEAR et que, peu de temps après, j'ai commencé à travailler sur mes propres paquets pour PEAR…

Je suis presque sûr d'avoir violé au moins 9 des 10 règles que je vais essayer d'écrire ici, règles que vous devriez connaître et prendre à coeur si vous voulez faire partie de la communauté open source. En tout cas, vous devrez garder ces règles en tête si vous voulez entrer dans la communauté PHP, mais je suis presque sûr que cela s'applique aussi à d'autres communautés.

Si vous voulez vous lancer tout de suite dans le développement open source, soyez sûr de lire les règles suivantes avant d'aller plus loin. Je suis sûr que vous avez beaucoup, beaucoup de grandes idées à l'esprit et que vous ne pouvez pas attendre pour en parler et les réaliser. Mais, prenez d'abord une grande respiration et lisez les règles suivantes, pensez-y, prenez-les à coeur, puis recommencez et repensez-y encore…

1. Collez à votre niveau de Karma

L'open source n'est pas une démocratie. Vous avez entendu

autre chose ? C’est faux. Gardez toujours à l’esprit : l’open source n’est pas une démocratie. Chaque développeur a un certain niveau de Karma (inexprimé et inexprimable), ce qui lui permet de décider (ou même de dicter) des choses et d’utiliser certaines infrastructures de la communauté (comme leur système de versionnage , leurs serveurs…). Pensez au Karma comme à votre niveau de points dans un jeu de rôle. Plus le nombre de niveau auxquels vous avez joué augmente, plus vous résolvez des énigmes, plus votre niveau de Karma s’élève. Mais prenez garde, il peut aussi baisser si vous n’agissez pas correctement.

Cela étant, si vous êtes nouveau dans cette communauté, votre niveau de Karma est de 0, par défaut. Vous devrez toujours garder cela en tête. Donc, qu’est-ce que c’est tous ces trucs autour du Karma ? Le Karma représente essentiellement la confiance que la communauté a en vous. Il n’est pas possible de mesurer le Karma avec un nombre ou même de le deviner, parce qu’il y a tellement de facteurs influençant votre Karma, et votre niveau de Karma est différent pour chaque individu de la communauté. Par exemple votre niveau d’expérience technique influence habituellement grandement votre Karma : plus vous en savez à propos du sujet de votre projet et au sujet de l’environnement technique, plus votre Karma sera élevé. Un autre facteur est la quantité de travail que vous investissez dans le projet : si vous êtes membre de ce projet depuis longtemps et que vous avez déjà passé des milliers d’heures à travailler dessus, votre Karma sera probablement élevé aussi.

Il y a beaucoup d’autres facteurs qui influencent votre Karma de développeur open source, comme vous allez l’expérimenter dans les prochaines semaines et prochains mois. Ce que vous devez avant tout vous rappeler est ceci : si vous êtes nouveau dans la communauté votre niveau de Karma est de 0 (ou proche). Pour augmenter votre Karma vous avez besoin de montrer à la communauté que vous êtes digne de confiance. Vous y parviendrez en respectant simplement les 9 règles suivantes.

2. L’information est le Karma

La première chose que vous voudriez probablement faire c’est poser une question ou proposer une idée cool à la communauté. Il y a de bonnes chances pour que vous le fassiez sur une liste de diffusion, qui est le moyen de communication le plus commun dans le monde de l’open source. Evitez de faire cela tout d’abord ! Les gens se fatiguent vraiment très vite si vous leur demandez quelque chose qui est évident à leurs yeux ou si vous proposez une idée/posez une question qui a été proposée/posée par d’autres avant (surtout si cela est déjà arrivé plusieurs fois).

Donc, que devrez-vous effectivement faire avant de poster une question ? Cherchez l’information ! Essayez Google, les sites des projets, les archives des listes de diffusion, la sphère des blogs du projet et toutes les ressources que vous pourrez imaginer. N’effectuez pas une seule recherche, mais affinez votre recherche pour voir s’il n’y a vraiment rien de relatif à votre question. Il n’y a rien? Essayez encore de chercher ! Il n’y a vraiment rien ? Ok, alors allez-y et écrivez votre billet. Mais soyez sûr de lire les 8 règles suivantes avant de le faire !

Et que faire si vous avez une idée ? Faites la même chose que ce que vous devez faire pour les questions ! Regardez si quelqu’un d’autre n’a pas eu la même idée ou une idée similaire. Vous ne trouvez rien ? Vraiment sûr ? Faites alors des recherches sur le sujet. N’écrivez pas simplement quelque chose comme « Ne serait-il pas cool de… » ou « J’ai eu l’idée de… ». Effectuez des recherches sur le sujet dont vous voulez parler avec pédantisme. Comment d’autres projets (peut-être dans d’autres langages ou sur d’autres systèmes d’exploitations) résolvent-ils la question ? N’y a-t-il rien de similaire qui existe ? Pensez aux autres besoins des utilisateurs. Est-ce que c’est quelque chose spécialement pour vous ? Alors commencez à écrire une prétendue proposition. Choisissez un sujet explicite pour votre billet (pas seulement

« J’ai une idée » ou quelque chose comme ça). Commencez à écrire une spécification : quel est votre problème ? Quelle est votre idée pour le résoudre ? Comment cela s’intégre-t-il dans le projet ? Qu’est-ce qui est nécessaire pour le faire ? Soyez prolixe sur tout cela. Dans la plupart des cas, les autres personnes ont une perspective complètement différente de la situation globale.

3. S’y habituer

Un point très important avant que vous ne commenciez à devenir un « contributeur » est de vous habituer à ce avec quoi vous travaillez et au sujet dont vous parlez. Vous n’aurez probablement jamais utilisé certains des outils qui sont employés couramment pour le projet, ou bien les outils utilisés sont différents de ceux dont vous avez l’habitude. Habituez-vous à ces outils et, plus important encore, habituez-vous au projet lui-même. Connaître tous les processus qui sont mis en oeuvre dans votre communauté est un point important. Devenir un habitué des outils qu’ils utilisent l’est encore plus.

L’un des ces outils importants est GNU patch, un outil qui applique des modifications (communément appelé un *diff* pour difference) à une version existante du code. Générer un patch est en général réalisé avec l’outil GNU diff. Si vous voulez produire un patch pour du code, vous aurez probablement besoin de cet outil. Beaucoup de projets utilisent un système de versionnage pour archiver leur code source. Les programmes les plus courants sont CVS et son petit frère plus récent SVN. Habituez-vous à ces systèmes et utilisez les activement ! Ces deux systèmes vous autorisent à *extraire* (*check out* en anglais) une certaine version de la source, à la manipuler et automatiquement à génerer un *diff* pour vos changements.

Si vous avez déjà fait ça, continuez et lisez les 7 prochaines règles !

4. Ne vous surestimez pas

Vous êtes un geek cool. Vous faites du développement depuis des années et, dans votre entourage, vous êtes un des gars les plus intelligents. C'est super. Mais ne présumez pas que cela vaut aussi pour le projet que vous voulez rejoindre. L'open source est habituellement réalisée par des gens qui sont extrêmement intéressés par le sujet, qui sont parfois beaucoup plus intelligents que vous et qui ont probablement une centaine de fois plus d'expérience que vous dans ce sujet spécifique. Restez calme et soyez plutôt timide qu'arrogant. Réfléchissez aux réponses données par les membres du projet, même si vous les trouvez stupides de prime abord ou si elles vous semblent grossières. Etudiez les sujets dont les gens parlent avant d'écrire une réponse. Prenez les réponses à vos demandes sérieusement, même si elles peuvent vous sembler illogiques.

5. Agir signifie Karma

Comme dit auparavant, le Karma est une valeur appréciable, qui dépend d'une centaine de facteurs. Un de ces facteurs est la comparaison entre ce que vous dites et ce que vous faites. Si vous avez une idée pour un projet et que vous êtes capable de la réaliser : allez-y et mettez la en œuvre. Si vous avez besoin d'une fonctionnalité, vous la ferez probablement de toute façon et utiliserez votre patch pour votre usage personnel. Si cela fonctionne, allez à la règle 2 et attachez votre patch à la proposition que vous avez écrite ! Cependant, avant d'agir, finissez de lire les 5 règles suivantes.

6. Soyez amical et ouvert

La plupart des développeurs open source avec qui vous traiterez auront probablement fait de l'open source depuis des années. Ils sont déjà stressés par des utilisateurs qui attendent que quelque chose se passe mal et par les nouveaux développeurs qui ne collent pas aux règles que vous êtes en train de lire. Souvenez-vous de ceci quand vous lisez leurs

mails. Ces gars ne sont pas inamicaux ou grossiers, ils sont seulement occupés et ennuyés. Vous arriverez dans un état, où vous aurez à lire 20 listes de diffusion avec des milliers de posts, garder un oeil sur le grand nombre de lignes de code, intervenir dans beaucoup de discussions et interagir avec beaucoup de personnes différentes. Quand vous lirez les mails, prenez juste l'essence objective des mots que vous lisez et ignorez la tonalité que vous pourriez ressentir.

7. S'énerver est mauvais !

Cette règle[2] est presque la même que la règle 6, mais c'est toujours très important. Lisez chaque conversation avec attention. Je connais ce sentiment assez bien, lorsque vous pensez que votre interlocuteur « est un idiot ». Il ne l'est pas ! Il n'y a pas d'idiot ici. Il a juste un point de vue différent du votre, ou a une base technique différente. Restez calme, réfléchissez à votre réponse pendant quelques minutes/heures/jours, puis écrivez-la quand vous ne serez plus en colère. Essayez d'énoncer vos arguments avec des mots polis et expliquez en détails pourquoi vous avez une opinion différente. Si vous sentez votre colère revenir pendant que vous écrivez, gardez votre réponse et revoyez-la encore plus tard. Les discussions enflammées sont vraiment une mauvaise chose et polluent les canaux de communication de votre projet. Elles ne cesseront jamais d'être mais c'est ainsi. La seule chose que vous pouvez faire contre cela, c'est de ne pas y prendre part.

8. Respecter le code étranger

Chaque partie du code que vous verrez a été écrite dans un but précis. Si vous parcourez le code d'autres personnes, vous penserez souvent « Quoi !!! ». Ne changez pas immédiatement le code pour qu'il fonctionne comme vous l'attendez personnellement. Prenez contact avec le développeur qui a écrit le code (par exemple en utilisant « svn blame », voir la règle 3). Discutez de votre point de vue avec lui. Ne faites

pas cela en public tant que cela n'affecte pas une grande partie du projet. Si vous le faîtes, référez-vous au système de communication générale de votre projet et annoncez le problème à cet endroit. Souvenez-vous à tout prix des règles 2, 3 et 4. Si vous ne pouvez pas décider si un problème y a sa place, prenez contact avec des gens du projet que vous connaissez déjà et demandez-leur de l'aide. Si vous êtes sympa et que vous leur décrivez ce que vous voulez, ils vous aideront sûrement.

9. N'attendez rien

L'open source signifie habituellement travailler sur la base du volontariat. Ces gens fournissent (pour la plupart) des logiciels gratuits, donc ils ne font pas d'argent avec ce qu'ils réalisent de leurs mains. Ils le font pour différentes raisons. Certains sont juste idéalistes, d'autres veulent simplement partager quelque chose, d'autres veulent se faire connaître, d'autres veulent faire de l'argent avec les services qu'ils fournissent en plus et d'autres encore ont tout en même temps à l'esprit ou encore d'autres raisons… Quelle que soit leur raison pour faire de l'open source, ils le font gratuitement. Gardez toujours cela à l'esprit quand vous leur soumettez une requête.

Par exemple « les demandes de fonctionnalités » sont une bonne chose. Elles donnent aux développeurs une idée de ce dont ils pourraient aussi avoir besoin. Cependant, la majorité des développeurs open source implémenteront seulement les fonctionnalités dont ils ont aussi besoin, ou dans lequel ils voient un défi technique intéressant. Ne soyez pas ennuyé s'ils refusent d'implémenter une fonctionnalité dont vous auriez besoin. C'est leur strict droit de le refuser, jusqu'à ce que vous les payez pour le faire. Si une demande de fonctionnalité est refusée ou n'est pas implémentée, allez-y, implémentez-la vous-même ou envisagez de payer un développeur pour son implémentation. Les développeurs open source ont aussi besoin d'argent pour vivre et ils ne refuseront

probablement pas de vous fournir un patch en échange d'un salaire. Quoi qu'il en soit, ils pourraient encore ne pas inclure votre idée dans leur projet ou l'implémenter d'une façon différente. Ne soyez pas fâché ! Ils ont le droit de le faire ! Essayez de faire avec leur conclusion ou essayez de les convaincre de le faire différemment, mais souvenez-vous toujours de respecter les 8 règles précédentes quand vous le faites.

Peu importe ce qui a été décrit avant, reconsidérez toujours votre idée plusieurs fois. Attendre quelque chose d'un développeur open source n'est pas la manière dont les choses marchent habituellement. Le faire vous-même est ce qui habituel.

10. Apprendre est tout

L'idée la plus importante derrière l'open source est l'apprentissage. Les gens fournissent leurs sources de manière ouverte pour permettre à d'autres personnes d'apprendre de leurs sources et d'apprendre des contributions des autres. C'est la même chose pour n'importe quel savoir ou connaissance fourni à propos du projet. Regardez simplement cet article et réfléchissez aux raisons pour lesquelles j'ai pu l'écrire ? C'est exact, c'est parce que je veux partager mon expérience de la communauté open source avec toute personne la découvrant et parce que je veux avoir des retours des autres, pour voir où je pourrais encore avoir des faiblesses et ce à quoi je n'ai pas prêté attention, pour le moment.

Soyez sûr d'apprendre quelque chose de chaque ligne que vous lisez, que ce soit du code ou une conversation. Vous pouvez apprendre de tout le monde, même si cela vous permet seulement d'apprendre comment une chose ne doit pas être faite…

Pendant que j'écrivais ces 10 règles, que je considère très importantes à lire pour chaque nouveau développeur open source, j'avais déjà d'autres règles en tête. Mais restons en

là avec ces 10 règles, pour donner un point de départ. Si je pense à d'autres choses qui s'avèreraient être un ajout réellement intéressant, je les ajouterai plus tard. Nous verrons. Merci pour tous vos retours et j'espère que ce billet sera utile à chaque nouveau…

Kore m'a aussi conseillé de me référer à De la bonne manière de poser les questions, qui lui a rappelé ce billet, lorsqu'il le relisait. Je n'ai pas lu cet article avant (mais peut-être un autre semblable), mais cela a vraiment l'air approprié. Si vous avez d'autres ressources, n'hésitez pas à laisser un commentaire !

Notes

[1] Crédit photo : TheAlieness GiselaGiardino (Creative Commons By-Sa)

[2] NdT : Le titre original était *Flaming is bad*. Vous trouverez une explication du flaming sur Wikipédia.