Le projet, c'est d'abord des personnes (Libres conseils 34/42)

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

Traduction Framalang : Ouve, Julius22, Sphinx, fubik, peupleLà, goofy, KoS, merlin8282, Munrek, Asta, Jej, Alpha, lamessen

L'important, c'est les gens

Nóirín Plunkett

Nóirín Plunkett est une touche-à-tout qui maîtrise plusieurs domaines. Rédactrice technique le jour, son travail open source *illustre l'expression « Si vous voulez que quelque chose soit fait, demandez à une personne occupée ». Nóirín a commencé dans l'*open source *avec Apache, donnant un coup de main sur la documentation du projet httpd. En moins d'un an, elle a été recrutée dans l'équipe de planification des conférences, qu'elle dirige désormais. Elle a participé à la mise en place du projet de développement communautaire chez Apache et a déjà agi en tant qu'administratrice d'organisation pour le Summer of Code. Elle siège aux conseils d'administration de la fondation du logiciel Apache et de l'Initiative Open Cloud. Quand elle n'est pas en ligne, elle est dans son élément naturel sur une piste de danse. Mais c'est également une harpiste et chanteuse talentueuse et une excellente* sous-chef *(NdT : en français dans le texte).*

Rien ne vaut une voie classique, bien que la mienne le soit peut-être moins que la plupart des autres. J'ai fait ma

première contribution quand j'avais la vingtaine. À cette époque, j'avais déjà travaillé plus d'un an chez Microsoft. Mais après Microsoft, j'ai déménagé à l'étranger afin de poursuivre mes études. C'était sympathique d'avoir un divertissement, j'ai donc commencé à travailler sur différentes documentations et traductions et j'ai contribué au projet httpd d'Apache.

Comme par hasard, bien sûr, la conférence européenne sur Apache allait avoir lieu à Dublin, alors que, cet été-là, j'étudiais à Munich. Mais la chance sourit aux Irlandais et, avec un peu d'astuce, j'ai convaincu Sun Microsystems de financer ma participation à la conférence.

J'ai une photo du moment où j'ai pris conscience que cette chose appelée *open source* était bien réelle, et que ça allait changer le monde. C'était pendant la soirée avant la conférence. Nous n'avions toujours pas trouvé où la fibre se terminait, elle était censée constituer la colonne vertébrale de notre réseau. Nous avions vérifié chaque coin, chaque armoire et chaque plinthe, en vain. Nous avions laissé tomber pour cette nuit, et nous étions occupés à nous assurer que les salles qui accueilleraient les sessions de formation auraient au moins suffisamment de connectivité pour que les formateurs puissent utiliser leurs supports de présentation (1).

Et à mesure que la nuit tombait, que les routeurs révélaient lentement les secrets de leurs configurations par défaut, la demi-douzaine de volontaires, des gens que je n'avais rencontrés que dans l'après-midi même, devenaient des amis.

Je ne pourrais pas vous dire où sont les six filles avec lesquelles j'ai vécu pendant cet été-là à Munich. Mais je suis toujours en contact avec chacune des personnes que vous voyez sur cette photo. L'une d'elles a déménagé dans un autre pays, une autre est partie sur un autre continent. La plupart ont changé de travail entre-temps, j'ai eu mon diplôme et je me suis conformée à la grande tradition irlandaise de

l’émigration pour trouver du travail.

Vous voyez, l’*open source*, c’est d’abord des gens. Vraiment, sur presque n’importe quel projet dont vous voudriez faire partie, le code ne vient qu’après.

Ce qui fait que travailler sur un projet est un bonheur et non une plaie, ce sont les gens. Ce qui fait qu’un projet prospère plutôt qu’il ne stagne, ce sont les gens. Bien entendu, vous serez capable de coder toute la nuit pour un projet si ça permet de résoudre un problème que vous pensez être important ; mais, à moins d’avoir des gens avec lesquels vous pouvez collaborer, discuter, concevoir et développer, vous allez probablement finir par perdre la motivation ou vous retrouver bloqué pour un bout de temps.

Les conférences, les sprints, les hackathons, les « retraites » (NdT : une ou plusieurs journées qui se concentrent sur la création de code de très bonne qualité plutôt qu’écrit dans l’urgence) ou tout ce que votre communauté appelle ses « moments de face à face », voilà leur vraie valeur : permettre de se retrouver face à face avec les gens avec lesquels vous avez travaillé. Les êtres humains sont des animaux sociaux ; les bébés reconnaissent des visages avant même de commencer à gazouiller, et peu importe à quel point les gens sont polis ou amicaux dans leurs courriels, il y a toujours quelque chose qui manque dans ces communications-là.

Rencontrer des gens en face à face nous donne une occasion de voir l’humanité de ceux avec qui on a pu avoir du mal à s’entendre, de partager la joie du travail bien fait avec ceux avec qui on aime travailler. Ainsi, si j’avais un conseil à donner à ceux qui commencent, et j’aurais aimé qu’on me le donne, ça serait de sortir, de rencontrer des gens, de coller des noms aux visages dès que l’opportunité se présente (2).

Et si vous trouvez que les occasions sont rares et trop

espacées, n’hésitez pas à demander. Cherchez des gens qui voyagent près de chez vous ou qui vivent là où vous voyagez, dénichez un parrainage pour assister aux grands événements de la communauté, organisez votre propre événement !

C’est la richesse de nos communautés qui donne toute sa valeur à l’*open source*, ainsi que les efforts partagés vers des objectifs communs. Et, bien sûr, les sessions musique, les repas, les pintes et les soirées ! Ce sont les choses qui nous rassemblent, et vous allez découvrir qu’une fois que vous avez rencontré les gens en personne, même vos interactions par courriel seront plus riches, plus gratifiantes et plus fructueuses qu’elles ne l’étaient auparavant.

_____

Notes de l’auteur :

(1) Le lendemain matin, nous sommes allés dans les combles pour essayer de trouver la fibre, toujours rien. Pour finir, nous l’avons trouvée dans le local technique de la boîte de nuit, située dans le sous-sol à côté.

(2) Malheureusement, je dis ça comme une mise en garde : comme dans tout rassemblement important, assister à une conférence *open source* présente des risques. Certains pires que d’autres, mais d’après mon expérience, les agressions, particulièrement, semblent plus fréquentes dans les communautés techniques que dans les communautés non-techniques. Dénichez les événements qui publient un code de conduite ou une politique anti-harcèlement et demandez de l’aide si vous ne vous sentez pas en sécurité. La grande majorité des gens que vous trouverez dans un événement *open source* sont des êtres humains formidables et attentionnés. J’espère qu’avec le temps, changer les attitudes empêchera la minorité de penser qu’elle peut se permettre des comportements déraisonnables dans ce genre de lieux…

S’intégrer au projet par l’action, sans attendre (Libres Conseils 31/42)

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

Traduction Framalang : merlin8282, goofy, Corentin, lerouge, Asta, peupleLà, Alpha, lamessen, Julius22

Trouver ses marques dans une équipe de promotion

Stuart Jarvis

Stuart Jarvis a commencé à travailler avec l’équipe de promotion de KDE en 2008 en écrivant des articles pour le site web d’actualités de KDE. Il a appris à la dure comment faire bouger les choses dans une communauté du logiciel libre et participe davantage aux activités de l’équipe de promotion en écrivant les annonces des nouvelles versions de KDE et en rédigeant des articles sur les logiciels KDE dans la presse Linux. Il siège maintenant dans le groupe de travail marketing de KDE, contribue à définir la ligne de conduite pour la promotion de KDE et les activités marketing et aide les nouveaux contributeurs à trouver leurs marques. Il fait maintenant aussi partie de l’équipe éditoriale de KDE.News, là où il a commencé à participer.

« C’est celui qui code qui décide » est le mantra du développement dans le logiciel libre. Mais que faire quand il

n’y a pas de code ?

Rejoindre l’équipe de promotion et de marketing de votre projet de logiciel libre préféré représente un défi particulier. Pour les nouveaux codeurs, la plupart des projets ont des systèmes de révision du code, des mainteneurs et des pré-versions du logiciel qui les aident à mettre en évidence les erreurs dans le code, ce qui rend moins effrayante la contribution à leur premier correctif.

La promotion peut nécessiter que votre travail soit visible par le public, après une relecture minimale, parfois immédiatement. La nature non-hiérarchisée des communautés de logiciel libre implique qu’il y a rarement une unique personne vers qui vous pouvez vous tourner et qui pourra vous dire si vos idées sont bonnes et prendre des responsabilités à votre place.

Obtenir un consensus versus obtenir des résultats

J’ai d’abord commencé à contribuer à KDE en écrivant des articles pour le site d’actus officiel, KDE.News. J’avais déjà écrit pour des organes de presse, mais j’avais toujours affaire à une personne bien identifiée à qui j’envoyais un brouillon pour avoir un retour et faire les corrections demandées. Dans l’équipe de promotion de KDE, il n’y avait pas une seule personne ou un seul groupe de personnes pour assumer cette tâche. Je devais essayer, juger aux réponses que j’avais sur les brouillons d’articles et décider si j’avais tous les retours dont j’avais besoin et si l’article était prêt pour une publication.

Avec les conseils de contributeurs plus expérimentés, j’ai finalement appris à proposer quelque chose et à le publier en quelques jours s’il n’y avait pas d’objection majeure. Cette approche peut être utilisée par n’importe quel contributeur

d’une équipe de promotion de logiciel libre, qu’il soit nouveau ou ancien.

Tout d’abord, travaillez sur la façon dont vous feriez quelque chose, que ce soit écrire un article, changer le texte d’un site web ou donner une conférence dans votre école locale. Planifiez, écrivez l’article ou le nouveau texte. Envoyez vos idées, pour relecture, sur la liste de diffusion de l’équipe de promotion de votre organisation. Surtout, ne demandez pas aux gens ce qu’ils en pensent – vous pourriez attendre des jours ou des semaines sans obtenir de réponse définitive. Signalez plutôt que vous allez publier ou soumettre votre texte, ou mettre en œuvre votre programme à telle date précise, en attendant les objections d’ici là.

Lorsque vous soumettez une date limite, pensez au temps nécessaire à chaque membre actif de l’équipe pour lire ses messages et évaluer votre proposition. Vingt-quatre heures est un minimum absolu pour un simple oui ou non en réponse à une question fermée. Lorsqu’il s’agit de quelque chose nécessitant une lecture ou une recherche, vous devriez envisager un délai de réponse de plusieurs jours.

Si la date limite que vous fixez ne rencontre pas une forte opposition, vous pouvez avancer. S’il existe de gros problèmes par rapport à votre projet, quelqu’un vous le dira. Les choses se font, en réalité. Vous ne serez pas frustré par un manque de progrès et vous aurez la réputation de mener à bien les tâches.

Finalement, c’est vous qui décidez

Les communautés du logiciel libre peuvent facilement devenir des groupes de discussion. Tout le monde a son opinion. Si vous n’êtes pas prudent, les discussions peuvent s’éterniser, s’évanouir au fur et à mesure que les personnes s’en désintéressent et finir sans conclusion convaincante. Cela peut paraître assez difficile à gérer lorsque vous faites

partie de la communauté depuis quelque temps : vous avez l'habitude de prendre vos propres décisions et d'avoir votre propre idée sur ceux dont les avis vous importent. Quand vous débutez, cela peut être très déroutant.

Si vous voulez que votre propre travail aboutisse, vous allez probablement devoir faire des choix entre des points de vue opposés. Vous pouvez mettre un terme au débat en donnant un résumé des points principaux et en donnant votre opinion sur ces points. Essayez de ne pas laisser de questions ouvertes en suspens, à moins que vous ne souhaitiez un débat plus long – donnez simplement vos conclusions et dites ce que vous allez faire. Dès lors que vous êtes correct, les autres personnes respecteront probablement votre avis, même si elles ne sont pas d'accord.

Soyez proactif – n'attendez pas qu'on vous demande

Le premier contact avec l'équipe de promotion que vous voulez rejoindre peut très bien être l'envoi d'un courriel sur leur liste de diffusion leur offrant vos compétences. Je pensais pouvoir énumérer mes points forts et espérer que les gens me suggéreraient des choses à faire. En pratique, ça ne fonctionne pas tout à fait comme ça.

La plupart des communautés manquent de volontaires et ont vraiment besoin de vos compétences. Mais comme elles manquent de volontaires, elles peuvent aussi manquer de temps pour donner de bons conseils et encadrer. Si vous voulez travailler sur une partie spécifique du projet, dites-le. Il est beaucoup plus facile pour quelqu'un du projet de dire simplement « Vas-y ! » plutôt que d'essayer d'arriver avec un projet qui correspond à vos compétences.

Même quand vous avez travaillé sur quelques projets et prouvé vos compétences, il y a peu de chances que vous soyez contacté

directement pour une tâche. Ceux qui coordonnent l'équipe marketing ne connaîtront pas votre situation personnelle et peuvent donc être mal à l'aise à l'idée de vous demander quelque chose de particulier sur votre temps libre, gratuitement. Une communauté idéale va poster régulièrement – que ce soit sur une liste de diffusion ou une page web – les tâches que des volontaires peuvent prendre en charge. Si ce n'est pas le cas, trouvez vous-même des choses à faire et prévenez la liste de diffusion que vous êtes en train de les faire. Les gens vont le remarquer et cela augmente les chances que vous soyez directement contacté dans le futur.

Si vous êtes proactif, vous pouvez rapidement vous rendre compte que vous êtes l'une des personnes expérimentées de la communauté vers qui les nouveaux venus se tourneront pour avoir des conseils ou du travail à réaliser. Essayez de vous souvenir comment c'était quand vous avez commencé et faites en sorte de faciliter au maximum leur vie de nouveau contributeur.

Coordonner les flux de contributions (Libres Conseils 30/42)

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

Traduction Framasoft : merlin8282, Sphinx, Julius22, goofy, Corentin, lerouge, Asta, peupleLà, okram, Alpha, lamessen

Au confluent de l’amont et de l’aval

Vincent Untz

Vincent Untz est un activiste passionné du logiciel libre, un amoureux partisan de GNOME, ainsi qu’un élément moteur d’openSUSE. Il a été responsable des versions de GNOME de 2008 à 2011, jusqu’à la sortie de GNOME 3.0 ; il a été directeur exécutif de la fondation GNOME (2006-2010) et il dirige l’équipe GNOME chez openSUSE. Quoi qu’il en soit, il trouve plus simple de se décrire comme un « touche-à-tout *» (NdT : en français dans le texte) et il travaille dans divers services (certains diraient au petit bonheur la chance) du bureau pour aider openSUSE à rester extraordinaire. Vincent continue à faire du forcing pour que le français soit la langue officielle de GNOME et espère bien réussir bientôt. Sinon, il aime la crème glacée.*

Il y a bien longtemps, dans une chambre, la nuit…

J’étais en train de jeter un dernier coup d’œil sur une liste de bogues pour voir si je n’avais pas oublié de fusionner un correctif. Je m’étais bien assuré d’écrire ce que je pensais être une entrée NOUVEAUTÉS au sujet de la nouvelle version. J’ai entré make distcheck (NdT : commande GNU permettant de créer un paquet et de le tester automatiquement dans un répertoire différent de celui de développement pour démarrer le processus de diffusion) et je regardais la console afficher des centaines de lignes. Une archive avait été créée, et j’ai à nouveau vérifié que l’archive se construisait correctement. J’ai vérifié, encore et encore : j’étais inquiet. D’une certaine manière, je ne faisais pas totalement confiance à la commande make distcheck. Après avoir tout vérifié plusieurs

fois, j’ai envoyé l’archive sur le serveur et expédié un courriel d’annonce.

J’avais réussi à le faire : j’avais sorti ma première archive d’un logiciel sur le développement dont j’étais récemment devenu co-responsable. Et j’ai certainement pensé : « Ah, maintenant les utilisateurs vont pouvoir apprécier un bon truc ! ». Mais à peine quelques secondes après le chargement de mon archive, quelques personnes l’ont téléchargée et ont rendu ma version réellement accessible aux utilisateurs.

C’est une chose que je tenais pour acquise, car je pensais que c’était une tâche banale. J’avais tort.

Amont et aval

De nombreuses personnes participent au processus d’acheminement du logiciel. Et cet effort se répartit généralement entre deux groupes de personnes d’égale importance dans la manière dont fonctionne le logiciel libre aujourd’hui.

En amont : c’est le groupe qui crée le logiciel. Il inclut évidemment les programmeurs mais, en fonction du projet, d’autres catégories de contributeurs sont également essentielles : designers, traducteurs, rédacteurs de documentation, testeurs, trieurs de bogues, etc. En général, l’amont se charge seulement de livrer le code source sous forme d’archive.

En aval : c’est le groupe responsable de la distribution du logiciel aux utilisateurs. Tout comme en amont, les contributeurs ont une gamme de profils très variée et travaillent à la traduction, la documentation, les tests, le triage de bogues, etc. Il y a cependant un profil qui, jusqu’à présent, est spécifique au travail en aval : le packager, qui prépare le logiciel afin de le rendre disponible sous forme de paquet, un format mieux adapté à un usage facile que le seul

code source.

Chose intéressante, les utilisateurs ont plutôt bien l'intuition de cette séparation également, bien que nous n'en soyons pas conscients : nous supposons souvent que les développeurs de logiciels sont inaccessibles et nous envoyons plutôt nos retours et demandes d'aide aux distributeurs.

Pour éclairer cette séparation entre amont et aval, une analogie parlante et classique est celle du circuit des biens de consommation, avec les magasins de détail (« l'aval ») qui distribuent des produits manufacturés (« l'amont ») et jouent un rôle important pour les clients (« les utilisateurs »).

Un regard plus attentif sur l'aval

Si je devais résumer le rôle de l'aval en une phrase, voici comment je le décrirais :

L'aval est le pont entre les utilisateurs et l'amont.

Lorsque j'ai sorti ma première archive en amont, je supposais que, pour l'aval, le travail consisterait principalement à compiler la source pour construire un paquet avec, et rien d'autre. Construire un paquet est effectivement la première étape. Mais c'est seulement le début du voyage vers l'aval : différentes tâches viennent ensuite. Certaines sont purement techniques tandis que d'autres sont sociales. Je vais me contenter de décrire très brièvement ce voyage ici, de manière non-exhaustive, puisque ça pourrait faire l'objet d'un chapitre entier de ce livre (1).

La construction du paquet proprement dit peut se révéler moins triviale que prévu. Il n'est pas rare qu'un *packager* rencontre des problèmes qui étaient inconnus de l'amont. Comme lorsqu'une nouvelle version du compilateur est utilisée (avec de nouvelles erreurs), qu'une bibliothèque spécifique a d'abord besoin d'être mise à jour (parce que l'archive utilise

de nouvelles interfaces de programmation) ou bien que le système de compilation de l'archive est conçu pour une certaine façon de fonctionner (qui ne suit pas les directives de la distribution cible). Ce qui est encore plus méconnu par beaucoup est le fait que tous ces problèmes peuvent également se produire après que l'archive a déjà été *empaquetée*, comme lors de la migration d'une distribution entière vers un nouveau compilateur ou bien une nouvelle chaîne de compilation. Aucun de ces problèmes techniques n'est vraiment compliqué à résoudre en lui-même, et l'amont est souvent content de contribuer à la solution. Mais sans l'aval, ces problèmes pourraient ne pas être remarqués par l'amont avant un long moment.

Ce qui selon moi est plus important que ces défis techniques, c'est que l'aval est généralement en contact avec davantage d'utilisateurs que l'amont. Cela se traduit par des rapports de bogue, des demandes de support, des requêtes de changement de la configuration par défaut ou bien d'autres choses encore. C'est là que la foule en aval donne la mesure éclatante de sa force : au lieu de simplement transférer ça en amont, l'aval va travailler sur les retours des utilisateurs afin de ne renvoyer que des synthèses qui seront utilisables en amont. Bien souvent, les rapports de bogue sont soumis avec trop peu d'informations sur le problème (auquel cas l'aval demandera plus de détails). Souvent, les demandes de support sont issues d'une incompréhension du côté de l'utilisateur (ce que l'aval peut, parfois, traduire par une suggestion visant à modifier le programme afin d'éviter cette incompréhension). Souvent, de nouveaux paramètres par défaut sont suggérés sans réflexion suffisante (l'aval travaillant alors avec les utilisateurs pour voir si le raisonnement est valide). À partir de cette énorme quantité de données, l'aval produira un ensemble d'informations plus compact que l'amont sera en mesure de digérer facilement. Ce qui amènera à des améliorations sur le logiciel.

Il existe généralement deux récompenses pour les contributeurs en aval : les contributions directes et indirectes vers le projet en amont grâce aux efforts effectués par l'aval sont suffisantes pour beaucoup. Mais plus important encore, le contact direct avec davantage d'utilisateurs amène à recueillir la satisfaction qu'ils expriment. Et une situation aussi gratifiante rend facilement heureux beaucoup de gens.

Une petite note au passage : lorsqu'on considère la quantité de travail fournie en aval, je ne serais pas surpris qu'au bout du compte, beaucoup de contributeurs en amont soient bien contents d'avoir des gens agissant comme intermédiaires : cela diminue significativement la quantité de retours tout en améliorant leur qualité (en évitant les commentaires en double, les problèmes non documentés, etc.). Cela permet à ceux qui travaillent en amont de rester focalisés sur le développement lui-même, au lieu de les obliger à soit trier les retours, soit les ignorer.

Rien qu'en regardant mon expérience en amont, je ne compte plus le nombre de correctifs que j'ai reçus de l'aval pour résoudre des problèmes de compilation. Je me rappelle aussi d'innombrables discussions que j'ai eues à propos des bogues qui affectaient le plus les utilisateurs et qui m'ont permis de prioriser mon travail. De fait, depuis que j'ai rejoint les équipes en aval, j'ai commencé à faire remonter des correctifs proches de ceux liés à des problèmes de compilation à l'amont et à discuter avec ma base en aval pour transmettre des retours d'expérience d'utilisateurs. Une telle collaboration amont-aval participe à l'amélioration de la qualité générale de notre écosystème du logiciel libre et je la considère comme essentielle à notre bonne santé.

Remonter de l'aval vers l'amont !

Je crois fermement que, pour qu'un projet réussisse, il faut qu'il y ait une forte collaboration amont-aval. Je doute que

beaucoup désapprouvent. Cependant, par « aval », les gens pensent généralement au travail fait dans les distributions. Mais, particulièrement pour les applications, il devient de plus en plus viable de pousser ce travail fait en aval en dehors des distributions et de tirer parti d'un tel mouvement vers l'amont.

Des outils comme l'Open Build Service (NdT : distribution *open source* dédiée à la compilation de paquets pour diverses distributions GNU/Linux) permettent plus facilement d'avoir des personnes qui compilent et distribuent des paquets d'une application pour plusieurs distributions. Cela présente des avantages à la fois pour les utilisateurs (qui peuvent profiter plus rapidement et plus facilement des mises à jour de leurs applications préférées) et pour l'amont (qui peut aider à construire une relation plus forte avec sa base d'utilisateurs). Le seul défi qu'un tel mouvement représente est le besoin perpétuel d'avoir quelqu'un qui s'occupe de l'empaquetage, mais aussi qui gère des retours plus nombreux des utilisateurs. Dans les faits, il y a toujours besoin de quelqu'un pour faire le travail de l'aval, sauf qu'il serait fait au sein de la branche amont.

Pour moi, cela semble une perspective excitante et j'irais même plus loin en suggérant que nous, la communauté du logiciel libre, devrions migrer lentement le travail d'aval fait sur les distributions vers un travail d'aval fait directement, aussi souvent que possible, en amont. C'est souvent possible, au moins pour les applications. Cela requiert évidemment de penser différemment. Mais ça permettrait de partager un travail qui, actuellement, est le plus souvent dupliqué sur toutes les différentes branches en aval.

Pour ceux qui souhaitent actuellement commencer à contribuer aux applications qu'ils apprécient, ce travail sur les paquets en amont est une toute nouvelle approche qui pourrait vraiment être une réussite !

J’ai essayé, je suis resté. Pourquoi pas vous ?

L’aval a toujours été essentiel à ma vie en tant qu’utilisateur de logiciels libres – après tout, seules quelques personnes installent manuellement leur système à partir de zéro et je n’en fais pas partie. Cependant, c’est aussi devenu un atout pour moi en tant que développeur en amont, étant donné que j’ai commencé à prendre plus de temps pour discuter avec des personnes en aval afin d’obtenir plus de retours sur les bogues, les fonctionnalités, la qualité générale et même les directions futures du logiciel sur lequel je travaillais.

C’est seulement lorsque j’ai commencé à être moi-même en aval que j’ai compris que cette position est en effet privilégiée afin de conseiller en amont, grâce au contact direct avec les utilisateurs et la perspective différente que l’on retient de cette position différente.

Sans l’aval, nous ne serions pas là où nous sommes aujourd’hui. Si vous souhaitez avoir un impact significatif, soyez persuadé qu’en participant en aval et en discutant avec l’amont, vous réussirez.

Et vous y prendrez du plaisir.

(1) Note de l’auteur : Il est bon de mentionner que je ne crois pas que l’aval devrait modifier significativement le logiciel mis à disposition par l’amont. Certains, en aval, le font tout de même et cela s’ajoute à leur charge de travail.

Passer de l’exercice scolaire à la maintenance des paquets (Libres conseils 28/42)

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

Traduction Framalang : satanas_g, Sphinx, Sky, Julius22, peupleLà, lamessen, goofy

Du débutant au professionnel

Jonathan Riddell

Jonathan Riddell est développeur KDE et Kubuntu, actuellement employé par Canonical. Quand il n’est pas devant un ordinateur, il fait du canoë sur les rivières d’Écosse.

Il y avait un bogue dans le code. Un bien méchant en plus : un plantage sans enregistrement des données. C’est bien là le problème dès qu’on regarde le code, on trouve des trucs à réparer. C’est facile de s’impliquer dans le logiciel libre ; le plus dur est d’en sortir. Après le premier bogue réparé, il y en a d’autres, et de plus en plus, tous à portée de main. Les corrections de bogues mènent à l’ajout de fonctionnalités, ce qui mène à la maintenance de projet, ce qui mène à faire fonctionner une communauté.

Tout a commencé en lisant Slashdot, cette masse d’actualité geek et technique peu filtrée avec des commentaires de quiconque peut recharger assez vite pour être en haut de liste. Chaque actualité était intéressante et excitante, apportait un éclairage nouveau sur le monde de la technologie qui finissait par me fasciner. Je n’avais plus à accepter ce qui m’était donné par de grandes entreprises de logiciels, je

pouvais voir là, dans la communauté du logiciel libre, le code se développer devant moi.

En tant qu'étudiant, il était possible de finir les exercices donnés par les professeurs très rapidement. Mais les exercices ne sont pas des programmes terminés. Je voulais savoir comment appliquer les compétences basiques qu'ils m'avaient données dans le monde réel en écrivant des programmes résolvant des problèmes réels pour les gens. J'ai donc recherché du code, qui n'était pas difficile à trouver, il se trouvait là, sur Internet, en fait. En regardant le code des programmes que j'utilisais de plus près, j'y ai décelé de la beauté. Non pas parce que le code était parfaitement soigné ou bien structuré, mais parce que je pouvais le comprendre avec les concepts que j'avais déjà appris. Ces classes, méthodes et variables étaient bien en place, me permettant de résoudre les problèmes pertinents. Le logiciel libre est le meilleur moyen de franchir le pas entre savoir comment finir ses exercices de cours et comprendre comment de vrais programmes sont écrits.

Tous les étudiants en informatique devraient travailler sur du logiciel libre comme sujet de leur mémoire. Sinon, vous avez de grandes chances d'y passer six mois à un an pour qu'il finisse au sous-sol d'une bibliothèque sans être jamais plus consulté. Seul le logiciel libre permet d'exceller en faisant ce qui va de soi : vouloir apprendre comment résoudre des problèmes intéressants. À la fin de mon projet, des programmeurs de la NASA utilisaient mon outil de création de diagrammes en **UML** (NdT : langage de modélisation unifié) et il reçut des prix au cours de réceptions somptueuses. Avec le logiciel libre, on peut résoudre de vrais problèmes pour de vrais utilisateurs.

La communauté des développeurs est remplie de personnes formidables, passionnées et dévouées à leur travail, sans espoir autre de récompense qu'un programme d'ordinateur couronné de succès. La communauté des utilisateurs est également incroyable. Il est satisfaisant de savoir qu'on a

aidé quelqu'un à résoudre un problème. Et j'apprécie les messages de remerciement que je reçois.

Après avoir écrit un logiciel utile, il faut le mettre à la disposition du plus grand nombre. Le code source ne va pas fonctionner pour la plupart des gens, il doit être compilé. Avant d'être impliqué, je trouvais que le fait de compiler était une manière un peu paresseuse de contribuer au logiciel libre. Vous vous attirez la plus grande partie de la reconnaissance sans rien avoir à coder. C'est, quelque part, quelque chose d'injuste. De même, la gestion de la communauté nécessaire pour porter un projet de logiciel libre peut aussi être vue comme une façon de s'attirer la reconnaissance sans faire de code.

Les utilisateurs dépendent beaucoup des *packagers* (NdT : les « empaqueteurs » qui préparent et maintiennent les paquets logiciels). Il est nécessaire que leur travail soit à la fois rapide, pour satisfaire ceux qui veulent la dernière version, et fiable, pour ceux qui veulent la stabilité (autant dire tout le monde). La partie la plus délicate, c'est que cela implique de travailler avec les logiciels des autres, qui sont toujours « cassés ». Une fois que le logiciel est lâché dans la nature, commencent à emerger des problèmes qui n'étaient pas repérables sur l'ordinateur de l'auteur. Il est possible que le code ne puisse pas être compilé avec une version de compilateur différente, peut-être que la licence n'est pas claire et ne permet pas de le copier, peut-être que la gestion des versions est incohérente et qu'une mise à jour mineure est incompatible, ou encore que la taille de l'écran est différente, les environnements de bureau peuvent aussi l'affecter, quelquefois, des bibliothèques tierces nécessaires ne sont pas encore à jour. De nos jours, le logiciel doit pouvoir tourner sur différentes architectures. Les processeurs 64 bits ont occasionné pas mal de problèmes quand ils sont devenus courants. Aujourd'hui, ce sont les processeurs ARM qui déjouent les calculs des codeurs. Les *packagers* doivent régler

tous ces problèmes pour donner aux utilisateurs quelque chose qui fonctionne de façon fiable.

Nous avons une règle chez Ubuntu selon laquelle les paquets avec des tests unitaires doivent inclure ces mêmes tests dans le processus de la création des paquets. Souvent, ils échouent et l'auteur du logiciel nous dit que les tests sont uniquement à son usage. Malheureusement, quand il s'agit de logiciel, il n'est jamais assez fiable de le tester soi-même, il doit aussi être testé par d'autres. Un test unique est rarement suffisant, il faut une approche à plusieurs niveaux. Les tests unitaires du programme original devraient être le point de départ, ensuite, le *packager* les teste sur son propre ordinateur, il faut ensuite que d'autres personnes les testent aussi. L'installation automatique et les tests de mise à jour peuvent être scriptés assez correctement sur les services d'informatique dans le nuage. L'envoyer dans la branche de développement d'une distribution permet d'effectuer plus de tests avant de le voir distribué en masse quelques mois après. À chaque étape, des problèmes peuvent être et seront découverts, ils devront être corrigés, puis ces correctifs eux-mêmes devront être testés. Il n'y a donc pas forcément à écrire beaucoup de code, mais il y a pas mal de travail pour passer le logiciel de 95 % à 100 % prêt. Ces 5 % sont la partie la plus difficile, un lent et délicat processus qui demande une grande attention pendant tout son cours.

Vous ne pouvez pas faire de paquets sans une bonne communication avec les développeurs en amont. Quand des bogues se produisent, il est vital de pouvoir trouver la bonne personne à laquelle parler rapidement. Il est important d'apprendre à bien les connaître comme des amis et des collègues. Les conférences sont vitales pour cela, car rencontrer quelqu'un apporte beaucoup plus de contexte à un message sur une liste de diffusion qu'une année entière de messages.

Une des faces cachées du monde du logiciel libre réside dans

la communication par les canaux IRC privés utilisés par les principaux membres d'un projet. Tous les grands projets en ont. Quelque part, Linus Torvalds a un moyen de discuter avec Andrew Morton et les autres sur ce qui est bon et sur ce qui est mauvais dans Linux. Ils sont plus sociaux que techniques et, quand on en abuse, ils peuvent être très antisociaux pour la communauté en général. Mais pour les moments où on a besoin d'un canal de communication rapide sans bruit parasite, ils fonctionnent bien.

Tenir un blog est un autre moyen de communication important dans la communauté du logiciel libre. C'est notre principale méthode pour promouvoir à la fois le logiciel que nous produisons et nous-mêmes. Non pas que ce soit utilisé éhontément pour de l'auto-promotion (il est inutile de prétendre que vous sauverez des vies avec votre blog…), mais parler de votre travail sur le logiciel libre aide à construire une communauté. Cela peut même vous valoir de trouver un travail ou d'être reconnu dans la rue.

Ces histoires venant de Slashdot, à propos de développements de nouvelles technologies, ne concernent pas des personnalités éloignées que vous ne rencontrerez jamais comme dans la presse *people*. Elles concernent des personnes qui ont trouvé un problème et qui l'ont résolu en utilisant l'ordinateur qu'elles avaient en face d'elles. Pendant quelques années, j'ai édité le site d'informations de KDE, trouvant les personnes qui résolvaient des problèmes, créaient des idées novatrices, s'acharnaient longuement à améliorer un logiciel jusqu'à ce qu'il soit d'une qualité suffisante, et j'en parlais au monde entier. Je n'ai jamais été à court d'histoires à raconter ni de personnes à présenter à tout le monde.

Mon dernier conseil est de conserver de la diversité. Il existe une telle richesse de projets intéressants à explorer, qui vous permettent d'apprendre et de progresser. Mais une fois que vous avez atteint une position de responsabilité, il

peut être tentant d'y rester. Après avoir aidé à créer une communauté pour Kubuntu, je repars temporairement vers un travail sur Bazaar, un projet très différent, orienté sur les développeurs plutôt que sur des utilisateurs novices en technologies. Je peux à nouveau apprendre comment le code devient une réalité utile, comment une communauté communique, comment la qualité est maintenue. Ce sera un défi amusant et j'ai hâte de m'y attaquer.

Intégrer un projet en se faisant connaître (Libres conseils 23/42)

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

Traduction Framasoft : Miki, peupleLà, dabou, Asta, elquan, Goofy, Julius22, okram, lamessen, Jej, lerouge, SaSha_01, Lycoris, meumeul, vvision

Ne soyez pas timide

Máirín Duffy Strode

Máirín Duffy Strode utilise des logiciels libres et open source depuis le lycée et y contribue depuis huit ans. Elle contribue aux communautés Fedora et GNOME et a travaillé sur l'identité visuelle des interactions, l'image de marque ou l'iconographie de plusieurs applications libres et open source importantes telles que Spacewalk, Anaconda, virt-manager, SELinux et SSSD. Elle s'est également engagée dans des

activités de sensibilisation en enseignant les techniques de design à des enfants à l'aide d'outils libres et open source tels que GIMP et Inkscape qu'elle défend ardemment. Elle est à la tête de l'équipe de conception graphique de Fedora et designer d'interaction senior chez Red Hat, Inc.

Je connaissais et utilisais des logiciels libres et *open source* bien avant de devenir contributrice. Ce n'est pas faute d'avoir essayé – il y a eu quelques faux départs auxquels je n'ai pas donné suite principalement parce que j'étais trop timide et que j'ai eu peur d'aller plus loin. Sur la base de ces tentatives avortées et de ce que m'ont rapporté d'autres designers qui se sont embarqués dans des projets libres et *open source*, j'ai cinq astuces à vous offrir si vous êtes un designer aspirant au statut de contributeur au logiciel libre et *open source*.

1. Sachez que l'on a besoin de vous et qu'on vous veut (très fort !)

Mon premier faux départ s'est produit alors que j'étais étudiante en première année d'informatique au Rensselaer Polytechnic Institute. Il y avait un projet particulier que j'utilisais beaucoup et auquel je voulais participer. Je ne connaissais personne au sein du projet (ni qui que ce soit investi dans le logiciel libre). J'ai donc fait une tentative à froid. Le site web du projet signalait que les contributeurs voulaient de l'aide et qu'ils avaient un canal IRC. J'y ai alors traîné pendant une semaine ou deux. Un jour, après une pause dans la conversation, j'ai osé élever la voix. J'ai dit que j'étais une étudiante en informatique intéressée par l'ergonomie et que j'adorerais participer.

On m'a répondu : « Dégage ! ». Qui plus est, on m'a fait comprendre que mon aide n'était ni nécessaire ni désirée.

Cela a retardé mon engagement de quelques années – il avait

suffi de quelques mots un peu rudes sur IRC pour me dissuader de réessayer pendant près de cinq ans. Je ne découvris que bien plus tard que la personne qui m'avait plus ou moins expulsée du canal IRC de ce projet était en marge du projet, qu'elle avait un lourd passif de ce genre et que je n'avais vraiment rien fait de mal. Si seulement j'avais persévéré dans mes tentatives d'approche et conversé avec d'autres personnes, j'aurais pu commencer à ce moment-là.

Si vous souhaitez contribuer au logiciel libre et o*pen source*, je vous garantis qu'il y a un projet quelque part qui a vraiment besoin de votre aide, en particulier si vous êtes orienté design ! Faites-vous du design web ? De l'iconographie ? De l'ergonomie ? De l'habillage ? Des maquettes d'interface utilisateur ? J'ai parlé à de nombreux développeurs de logiciels libres et *open source* qui non seulement sont désespérément à la recherche d'aide dans ce domaine, mais qui en plus l'apprécieraient vraiment et vous vénéreraient pour l'avoir apportée.

Si vous rencontrez des résistances la première fois que vous essayez de participer dans un projet, apprenez de mon expérience et n'abandonnez pas tout de suite. Si, en définitive, le projet n'est pas fait pour vous, ne vous inquiétez pas et passez votre chemin. Il y a des chances pour que vous trouviez un projet que vous adorerez et qui vous adorera en retour.

2. Aidez le projet pour qu'il vous aide à aider les autres

Bien des projets libres et *open source* sont aujourd'hui dominés par les programmeurs et les ingénieurs. Et si certains ont la chance qu'une ou deux personnes créatives s'investissent, dans la plupart des projets, un designer, un artiste ou une autre présence créative représente un rêve souvent-caressé-mais-jamais-réalisé. En d'autres termes, même

s’ils comprennent qu’ils ont besoin de vos compétences, ils peuvent ne pas savoir quelle sorte d’aide ils peuvent vous demander, quelle information ils doivent vous donner pour que vous puissiez être productif ni même les bases pour travailler avec vous efficacement.

Quand j’ai commencé à m’investir dans différents projets libres et *open source*, j’ai rencontré beaucoup de développeurs qui n’avaient jamais travaillé directement avec un designer auparavant. Au début, je me suis sentie un peu inutile. Je ne pouvais pas suivre toutes leurs conversations sur IRC parce qu’ils parlaient de leur cuisine interne et de détails techniques qui ne m’étaient pas familiers. Quand ils se sont donné la peine de me prêter attention, ils m’ont posé des questions comme « Quelle couleur dois-je mettre ici ? » ou « Quelle police dois-je utiliser ? ». Ce que je voulais vraiment en tant que designer d’interactions, c’était d’être associée à la prise de décision lorsqu’on abordait les contraintes spécifiques du projet. Si un utilisateur voulait une fonctionnalité particulière, je voulais avoir mon mot à dire sur le design – mais je ne savais pas où ni quand ces décisions se prenaient et je me sentais exclue.

Le design couvre une gamme assez large de compétences : l’illustration, la typographie, la conception des interactions, la conception visuelle, la conception d’icônes, la conception graphique, la rédaction, etc. et il y a peu de chances qu’un seul concepteur les possède toutes. Il est alors compréhensible qu’un développeur ne soit pas sûr de ce qu’il peut vous demander. Ce n’est pas qu’ils essaient de vous faire obstacle – c’est seulement qu’ils ne savent pas dans quelle mesure vous avez besoin de vous investir ou le désirez.

Aidez-les à vous aider. Montrez-leur clairement le type de contributions que vous pouvez apporter en fournissant des exemples de contributions antérieures. Faites-leur comprendre vos besoins de sorte qu’ils comprennent mieux comment vous aider à vous engager dans leur projet. Par exemple, lorsque

vous vous impliquez pour la première fois dans une initiative spécifique pour le projet, prenez le temps de présenter les grandes lignes de son processus de conception et postez cela dans la liste de développement principale afin que les autres contributeurs puissent vous accompagner. Si vous avez besoin d'idées sur des points particuliers, soulignez-les dans votre présentation. Si vous n'êtes pas certain de la façon dont certaines choses se produisent – comme le processus de développement d'une nouvelle fonctionnalité – entrez en contact avec quelqu'un en parallèle et demandez-lui de vous l'expliquer pas à pas. Si quelqu'un vous demande de faire quelque chose au-delà de vos capacités techniques – travailler sur de la gestion de versions, par exemple – et que vous n'êtes pas à l'aise avec ça, dites-le.

Communiquer sur votre processus et vos besoins vous évitera de jouer aux devinettes dans le projet et ses membres seront au contraire capables d'utiliser au mieux vos talents.

3. Posez des questions, beaucoup de questions ; il n'y a pas de question idiote

Nous avons parfois remarqué chez Fedora que, lorsque de nouveaux designers arrivaient à bord, ils avaient peur de poser des questions techniques, par crainte de paraître stupides.

Ce qu'on ne vous dit pas, c'est que les développeurs peuvent être tellement spécialisés qu'il y a beaucoup de détails techniques qui sortent de leur domaine de compétence et qu'ils ne comprennent pas non plus – cela se produit même au sein du même projet. La différence, c'est qu'ils n'ont pas peur de demander – donc vous ne devriez pas avoir peur non plus ! Dans mon travail de design des interactions, par exemple, j'ai dû contacter de nombreuses personnes du même projet pour

comprendre comment un processus se déroulait dans leur logiciel, car ce dernier comportait plusieurs sous-systèmes et tous les participants du projet ne comprenaient pas forcément comment chaque sous-système fonctionnait.

Si vous ne savez pas sur quoi travailler, que vous ne savez pas par quoi commencer ou que vous ne comprenez pas pourquoi ce que quelqu'un a dit sur le chat est si drôle – demandez. Vous avez des chances que quelqu'un vous réponde qu'il ne sait pas non plus et peu de risques de passer pour stupide. Dans la plupart des cas, vous allez apprendre quelque chose de nouveau qui vous aidera à devenir un meilleur contributeur. Il peut être particulièrement efficace de chercher un tuteur – certains projets ont même un programme de tutorat – et de lui demander s'il veut bien être votre référent quand vous avez des questions.

4. Partagez et partagez souvent, même si ce n'est pas encore prêt, surtout si ce n'est pas encore prêt

Nous avons aussi remarqué que de nouveaux designers pour Fedora et d'autres projets libres et *open source* sont un peu timides lorsqu'il est question de montrer leur travail. Je comprends qu'on ne veuille pas ruiner sa réputation en publiant quelque chose qui n'est pas ce qu'on peut faire de mieux ni même fini ; mais une grande partie du travail sur des projets libres et *open source* est de partager souvent et ouvertement.

Plus vous aurez avancé sur un élément avant de le partager, plus il sera difficile à d'autres de vous apporter un retour utilisable, de se lancer et de s'investir. Il est aussi plus difficile pour autrui de collaborer à votre travail et d'avoir un sentiment d'appartenance au projet, de le soutenir et de le pousser jusqu'à l'implémentation. Dans certains projets libres

et *open source*, ne pas être communicatif avec vos ébauches, compositions et idées est même considéré comme offensant !

Publiez vos idées, maquettes ou compositions sur le Web plutôt que par courriel, afin qu'il soit plus aisé pour les autres collaborateurs de faire référence à votre contribution en faisant un copier-coller de l'URL – c'est particulièrement pratique au cours d'une discussion. Plus vos éléments de design seront faciles à trouver, plus il est probable qu'ils seront utilisés.

Essayez ce conseil et gardez l'esprit ouvert. Partagez votre travail tôt et souvent. Rendez disponibles vos fichiers sources. Vous serez peut-être agréablement surpris par ce qui va se passer !

5. Soyez aussi visible que possible au sein de la communauté du projet

Un outil qui – de manière totalement involontaire – a fini par m'aider énormément à démarrer en tant que contributeur de logiciels libres et *open source* a été mon blog. J'avais commencé à entretenir un blog, simplement pour moi, à l'image d'un portfolio grossier des choses sur lesquelles j'avais travaillé par le passé. Mon blog est un énorme atout pour moi, parce que :

- En tant qu'enregistrement de l'historique des décisions de projet, il représente un moyen pratique pour rechercher d'anciennes décisions de design – comprendre pourquoi nous avons décidé de laisser tomber tel ou tel visuel à nouveau ou pourquoi une approche particulière, précédemment essayée, n'a pas fonctionné, par exemple ;
- En tant que dispositif de communication, il aide les autres contributeurs associés à votre projet et même les utilisateurs à être au courant des travaux en cours et des changements à venir pour le projet. De nombreuses

fois, j'ai omis quelque chose d'essentiel dans un design et ces personnes ont très rapidement posté un commentaire pour m'en informer !
- Il m'a aidé à construire ma réputation en tant que designer de logiciels libres et *open source*, ce qui m'a aidé à gagner la confiance des autres envers mes choix de design avec le temps.

Vous bloguez ? Trouvez quels agrégateurs de blogs lisent les membres du projet auquel vous participez et demandez à ce que votre blog y soit ajouté (il y a en général un lien pour cela dans la barre latérale). Par exemple, l'agrégateur de blogs que vous devrez rejoindre pour faire partie de la communauté Fedora se nomme Planet Fedora. Écrivez un premier billet pour vous présenter aux autres et leur faire savoir ce que vous aimez une fois que vous y aurez été ajouté – des informations du genre de celles listées dans la première astuce.

Le projet aura certainement une liste de diffusion ou un forum où les discussions ont lieu. Rejoignez-les et présentez-vous là aussi. Quand vous apportez une contribution au projet – peu importe qu'elle soit petite ou loin d'être aboutie – postez des billets sur ce que vous faites, téléchargez-le vers le wiki du projet, tweetez à ce sujet et envoyez des liens aux membres importants de la communauté via IRC afin d'avoir leur retour.

Rendez votre travail visible et les gens commenceront à vous associer à votre travail et à vous proposer des projets sympas ou d'autres opportunités, simplement grâce à ça. C'est tout ce que j'aurais aimé savoir quand j'ai essayé de m'investir pour la première fois dans le logiciel libre et *open source* comme designer. Si vous ne deviez retenir qu'un message de tout cela, c'est que vous ne devriez pas être timide – faites-vous entendre haut et fort, faites connaître vos besoins, faites savoir aux autres quels sont vos capacités et ils vous aideront à les utiliser pour que le logiciel libre envoie du lourd.

Apprenez de vos utilisateurs (Libres conseils 21/42)

21/42 ! Tiens, nous voilà déjà à mi-chemin de la traduction d'*Open Advice*.

Deux ou trois articles petits ou grands chaque jeudi, traduits en un temps record par une bande de furieux, ceux qui sont là depuis le début et ceux qui débarquent et demandent s'ils peuvent participer, ceux qui travaillent d'arrache-clavier et ceux qui en profitent pour ~~déconner~~ échanger des propos sur le chat du pad, ceux qui choisissent un pseudo et ceux qui restent *anonymous*, ceux qui négligent tranquillement l'orthographe et les *grammar nazis* qui rectifient, ceux qui traduisent avec *Google translate* et ceux qui se battraient pour une majuscule à Libre… on rencontre tout un peuple là, et jusqu'à présent tout se passe dans l'enthousiasme et la bonne humeur, l'échange et l'entraide devant un passage un peu ardu sur lequel on s'amuse à chinoiser…

Mais cette fois-ci c'est l'auteur de l'article lui-même qui nous a fait le plaisir de nous rejoindre pour contribuer à la traduction, ce qui est tout de même assez confortable. Merci Guillaume !

Malgré nos relectures croisées, nul doute que des coquilles auront échappé à notre vigilance et que nous trouverons des lecteurs pour les signaler, ce qui nous est précieux car la révision avant l'édition du framabook en sera d'autant facilitée.

D'ici là, en avant pour la deuxième moitié : chaque jeudi à 21h, rendez-vous sur le framapad de traduction, le travail collaboratif sera ensuite publié ici même.

Traduction Framalang : Nyx, Sphinx, peupleLà, Kalupa, Guillaume Paumier, Husi10, lenod, Sky, Julius22, Alpha, RavageJo, KoS, Sputnik, goofy, lamessen

Apprenez de vos utilisateurs

Guillaume Paumier

Guillaume Paumier est photographe et physicien, il habite à Toulouse. Wikipédien depuis longtemps, il travaille actuellement pour la Wikimedia Foundation, l'organisation à but non lucratif qui héberge Wikipédia. En tant que responsable de l'ergonomie multimédia, il a notamment étudié le comportement des utilisateurs afin de créer un nouveau système d'import de fichiers pour Wikimedia Commons, la médiathèque libre associée à Wikipédia.

Vous connaissez Wikipédia, l'encyclopédie libre et gratuite que tout le monde peut modifier ? Elle a été créée en 2001 et a récemment fêté son dixième anniversaire. Bien qu'elle soit l'un des dix sites les plus visités au monde, son interface reste très « 1.0 » quand on la compare aux possibilités qu'offrent les technologies web modernes. Certains peuvent trouver ça bien : Wikipédia est un « truc sérieux » et les utilisateurs n'ont pas à être distraits par des « paillettes » dans l'interface. Pourtant, si Wikipédia a eu du mal à recruter de nouveaux contributeurs ces dernières années, c'est en partie à cause de son interface que certains considèrent comme archaïque. Ceci explique peut-être pourquoi les enquêtes sur les participants à Wikipédia ont montré à maintes reprises qu'il s'agit principalement d'une population d'hommes jeunes, attirés par la technologie, la plupart ayant une formation en informatique et en ingénierie. En dehors du fait que la connaissance libre et les licences libres sont issues du terreau fertile du logiciel libre et open source, l'interface compliquée a découragé beaucoup de contributeurs éventuels.

En 2011, alors que la majorité des plates-formes de publication collaboratives en ligne (comme WordPress, Etherpad et Google Documents) offrent un éditeur graphique, même rudimentaire, Wikipédia utilise toujours, par défaut, un ancien éditeur de texte wiki qui utilise des guillemets (“”) et des crochets ([[]]) pour la mise en forme. Des efforts sont en cours afin de passer à un éditeur graphique par défaut en 2012, mais ce n’est pas un défi facile à relever.

Mais laissons l’éditeur de côté un moment. L’interface de Wikipédia demeure assez compliquée. Et de nombreuses fonctionnalités utiles sont difficiles à découvrir. Savez-vous que Wikipédia possède un système de contrôle de versions intégré ? Et que vous pouvez voir toutes les anciennes versions d’une page ? Savez-vous que vous pouvez voir la liste de toutes les modifications effectuées par un contributeur ? Savez-vous que vous pouvez créer un lien vers une version donnée d’une page ? Savez-vous que vous pouvez exporter une page en PDF ? Savez-vous que vous pouvez créer un vrai livre personnalisé à partir du contenu de Wikipédia ? Et que vous pouvez le faire livrer chez vous ?

Le modèle d’implémentation

La plupart des lecteurs de Wikipédia y arrivent via des moteurs de recherche. Les statistiques montrent qu’ils passent peu de temps sur Wikipédia une fois qu’ils ont trouvé l’information qu’ils cherchaient. Un petit nombre seulement s’attarde et explore les outils que propose l’interface. Par exemple, on critique régulièrement Wikipédia sur sa qualité et sur sa fiabilité. Nombre de ces outils rarement explorés et presque cachés pourraient s’avérer bien utiles aux lecteurs pour les aider à vérifier la fiabilité de l’information, telles que les « pages de discussion » qui témoignent des discussions (passées et en cours) entre les différents contributeurs de chaque article ayant abouti à son contenu actuel.

Wikipédia et ses projets frères (tels que Wikisource et Wikimedia Commons) sont propulsés par le moteur de wiki MediaWiki – et sont soutenus par la *Wikimedia Foundation* ; rien que ces noms, dans leur confusion, sont un péché contre l'ergonomie. Pendant longtemps, le développement de MediaWiki a été conduit par des développeurs de logiciels. La communauté MediaWiki est forte de nombreux développeurs ; à vrai dire, la communauté est presque exclusivement composée de développeurs. Ce n'est que récemment que des designers ont rejoint la communauté, et ils ont été recrutés par la *Wikimedia Foundation* pour ce rôle. Il n'y a quasiment aucun designer bénévole dans la communauté. De ce fait, l'application a été construite et « maquettée » exclusivement par des développeurs. Par conséquent, la forme de l'interface a naturellement suivi de très près le « modèle d'implémentation », c'est-à-dire la manière dont le logiciel est implémenté dans le code et les structures de données. Le modèle d'implémentation ne correspond que rarement au « modèle utilisateur », c'est-à-dire à la manière dont l'utilisateur imagine que le logiciel fonctionne.

Il serait injuste de dire que les développeurs ne se soucient pas des utilisateurs. Quand on crée un logiciel, le but – outre le plaisir d'apprendre des choses, d'écrire du code et de résoudre des problèmes – c'est de le publier afin qu'il puisse être utilisé. Ceci est particulièrement vrai dans le monde du logiciel libre et open source, où la plupart des développeurs donnent bénévolement de leur temps et de leurs connaissances. On pourrait avancer que les développeurs sont, de fait, des utilisateurs de leurs propres produits, notamment dans le monde du logiciel libre et open source. Après tout, ils les ont créés ou ont rejoint leurs équipes pour une bonne raison, et c'est rarement l'argent. Par conséquent, les développeurs de ce type de logiciels devraient être dans une position idéale pour savoir ce que veulent leurs utilisateurs.

Mais soyons honnêtes : si vous êtes en train de lire ceci,

c’est que vous n’êtes pas votre utilisateur lambda.

Le point de vue du développeur

Si vous êtes développeur, il vous est particulièrement difficile de vous mettre à la place de l’utilisateur. Tout d’abord, votre connaissance du code et de l’implémentation du logiciel vous force à observer ses fonctionnalités et son interface à travers un prisme très particulier. Vous connaissez chacune des fonctionnalités de l’application que vous avez créée. Vous savez où trouver chaque menu. Si quelque chose paraît légèrement bizarre dans l’interface, il est possible que vous l’ignoriez sans le vouloir, parce que vous savez inconsciemment que c’est lié à la façon dont la fonctionnalité est implémentée.

Imaginons que vous soyez en train de créer une application qui enregistre des données sous forme de tableau (par exemple, dans une base de données). Quand vous devrez ensuite afficher ces données pour les montrer à l’utilisateur, il est très probable que vous les représentiez comme un tableau, car c’est la façon dont vous avez implémenté leur stockage. Il vous paraîtra logique d’afficher les données dans un format qui est cohérent avec le format de stockage. Vous aurez probablement le même réflexe pour tout autre type de structure de données séquentielles : vous aurez tendance à l’afficher sous forme de séquence dans l’interface, peut-être comme une liste. Et pourtant, un autre format d’affichage aurait peut-être été plus pertinent et facile d’utilisation pour les utilisateurs, par exemple sous forme d’une série de phrases, d’un graphique ou d’une autre représentation visuelle.

Un autre défi est votre niveau d’expertise. Comme vous souhaitez que votre application soit extraordinaire, vous allez probablement vous documenter sur le sujet pour la concevoir. En fin de compte, vous n’allez pas seulement connaître votre application sur le bout des doigts, vous allez

également devenir un expert dans le domaine lui-même. Un grand nombre de vos utilisateurs n'auront pas ce niveau d'expertise – ou n'en auront pas besoin. Ils pourraient être perdus par le niveau de détail de certaines fonctionnalités ou ne pas être familiers avec des termes inconnus des profanes.

Alors, que pouvez-vous faire pour arranger cela ?

Observez les utilisateurs. Vraiment

Observer les utilisateurs à l'œuvre avec votre application est une expérience réellement révélatrice.

Bon, pour observer comment les gens utilisent votre application, vous pouvez faire appel à une société de conseil en ergonomie ; cette société va alors recruter des volontaires avec des profils différents au sein d'un vivier de plusieurs milliers de testeurs, elle va mettre au point une grille d'entretien, louer une salle dédiée aux tests d'ergonomie qui comprendra un dispositif pour enregistrer ce qui se passe sur l'écran et une caméra pointée vers le testeur, et vous serez derrière une vitre sans tain, dans une salle d'observation, à vous taper la tête contre les murs et à jurer à chaque fois que l'utilisateur fait quelque chose qui, selon vous, n'a aucun sens. Si vous avez les moyens de le faire, alors n'hésitez pas, foncez. Ce que vous y apprendrez vous permettra de complètement changer votre point de vue. Si vous n'avez pas les moyens de recourir à une procédure de test professionnelle, tout n'est pas perdu ; vous allez juste devoir le faire par vous-même. Asseyez-vous derrière un utilisateur pendant qu'il vous montre comment il effectue ses tâches et les intègre à son mode de travail. Soyez un observateur silencieux : votre but est d'observer et de noter tout ce qui se passe. Beaucoup de choses vont vous surprendre. Une fois que l'utilisateur a terminé, vous pouvez relire vos notes et lui poser des questions afin de mieux comprendre comment il fonctionne.

Pour vous aider à conduire ces tests vous-même, il existe d'excellents ouvrages comme *Don't Make Me Think: A Common Sense Approach to Web Usability* [NdT: Traduit en français : *Je ne veux pas chercher: Optimisez la navigation de vos sites et menez vos internautes à bon port*], écrit par Steve Krug, *About Face 3: The Essentials of Interaction Design*, d'Alan Cooper, Robert Reimann et David Cronin, et le projet OpenUsability. Être observé peut être un peu intimidant pour les utilisateurs, mais je parie qu'ils seront nombreux à se porter volontaires pour vous aider à améliorer votre application. Les utilisateurs qui ne peuvent pas contribuer au code sont généralement heureux de trouver d'autres façons de contribuer au logiciel libre : vous montrer comment ils utilisent le logiciel est une manière simple de le faire. Les utilisateurs sont reconnaissants du temps que vous avez donné pour développer le logiciel et veulent vous rendre la pareille.

Vous devrez garder un esprit critique et ne pas forcément accepter toutes les modifications demandées par vos utilisateurs. Écoutez attentivement leurs histoires : elles vous donneront l'occasion d'identifier des problèmes. Mais ce n'est pas parce qu'un utilisateur réclame une fonctionnalité qu'il en a absolument besoin ; peut-être que le meilleur moyen de résoudre le problème sous-jacent est de mettre en place une fonctionnalité complètement différente. Gardez du recul par rapport aux commentaires de vos utilisateurs. Mais cela, vous le saviez probablement déjà.

Et au passage, ne faites pas non plus appel à votre famille.

Je ne dis pas ça méchamment, je suis sûr que vos parents, frères et sœurs sont des gens très bien. Mais si vous créez une application comptable et que votre sœur n'a jamais tenu la moindre comptabilité, elle sera sans doute perdue. Vous perdrez plus de temps à lui expliquer ce qu'est la comptabilité en partie double qu'à tester votre logiciel. Par contre, votre mère, qui s'est acheté un appareil photo numérique l'année dernière, pourrait être un cobaye idéal si

vous travaillez sur une application de gestion de photos numériques ou d'envoi sur un site de partage en ligne populaire. Pour votre application de comptabilité, il vaut mieux demander à l'un de vos collègues ou amis qui a déjà quelques notions de comptabilité.

Variez vos cobayes

Pour des raisons qui resteront éternellement mystérieuses, les gens trouveront toujours d'innombrables façons d'utiliser et de maltraiter votre application. Ils trouveront des manières de la casser que vous n'auriez même pas imaginées dans vos pires cauchemars. Certains mettront en place des processus et des méthodes de travail avec votre application qui n'ont absolument aucun sens à vos yeux. Et, de désespoir, vous vous cognerez la tête contre les murs. D'autres utiliseront votre application avec tellement d'intelligence que vous vous en sentirez idiot. Essayez de rencontrer des gens qui utilisent votre application avec des objectifs différents.

Les utilisateurs sont de drôles d'oiseaux. Mais ils sont de votre côté. Apprenez d'eux.

Si vous ne retenez rien d'autre…

… alors retenez ceci :

- Vous serez tenté de modeler l'apparence et le comportement de votre interface sur la façon dont le logiciel fonctionne en coulisses. Vos utilisateurs peuvent vous aider à éviter ce piège.
- Les utilisateurs sont des oiseaux capricieux. Ils vont casser, maltraiter et optimiser votre application à un point que vous ne pouvez pas même pas imaginer.
- Apprenez de vos utilisateurs. Améliorez votre application en fonction de ce que vous avez appris. Vous avez tout à y gagner.

Grandir avec son projet (Libres conseils 20/42)

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

Traduction Framalang : peupleLà, SaSha_01, lerouge, Kalupa, lenod, michel, KoS, michel, goofy, HanX, Asta, lamessenJulius22

Mon projet m'a appris à grandir

Runa Bhattacharjee

Depuis 10 ans, Runa Bhattacharjee a traduit et travaillé à la localisation(1) de nombreux projets open source *– des interfaces de bureau aux outils de systèmes d'exploitation en passant par beaucoup de choses entre les deux. Elle croit fermement que les dépôts de codes en amont sont les meilleurs endroits pour soumettre toutes les modifications possibles. Elle gère également un portefeuille professionnel spécialisé dans la localisation chez RedHat. Runa traduit et maintient des traductions en Bengali (version indienne) mais est toujours heureuse d'aider quiconque débute dans la localisation.*

Introduction

Carburer tard dans la nuit est l'une des formes de rébellion préférée des jeunes partout dans le monde. Que ce soit pour

lire un livre avec une lampe de poche sous les couvertures, regarder les rediffusions TV ou (entre autres choses) traîner sur un canal IRC et s'acharner sur un problème agaçant dans son projet *open source* favori.

Comment tout a commencé

Voici comment tout a commencé pour moi. Permettez-moi d'abord d'écrire quelques lignes sur ma personne. Lorsque j'ai été présentée au groupe d'utilisateurs de Linux de ma ville, je partageais ma vie entre des emplois et mes études de master. Très vite, j'étais devenue contributrice sur quelques projets de localisation et j'avais commencé à traduire des interfaces de bureau (principalement). Nous utilisions des éditeurs de texte personnalisés avec des systèmes intégrés pour l'écriture et les polices. Les moteurs de rendu n'étaient pas assez matures pour afficher le scénario sans erreur sur les interfaces. Mais, malgré tout, nous continuions à traduire. Je me concentrais sur la méthode de travail que j'avais créée pour mon usage. Je récupérais le contenu à traduire des personnes qui savaient comment les choses fonctionnaient, je le traduisais du mieux que je pouvais, j'ajoutais des commentaires pour aider les réviseurs à comprendre comment j'avais appréhendé le texte, je renseignais les informations requises pour les copyrights et les crédits, puis je renvoyais tout ça aux coordinateurs.

Comment je faisais

C'était avant tout une manière simple de faire les choses. Mais, surtout, c'était ma manière à moi de les faire. Je prenais le temps de planifier mon travail sur les traductions. Celles-ci étaient ensuite révisées avant de m'être retournées pour modification. De nouveau, je planifiais quand je pourrais les reprendre en fonction de mon temps libre disponible entre les études et le travail. Lorsque je me mettais finalement au

boulot, je m’asseyais pour neuf à dix heures d’un coup, en général jusqu’à l’heure où blanchit la campagne, ressentant alors un grand sentiment d’accomplissement, jusqu’à la fois suivante.

Ce qui comptait

Ce que je ne savais pas, c’est que je jouais un rôle crucial sur un plan plus global. À savoir, la planification des releases(2). Donc, quand j’achevais mes modestes contributions et les envoyais aux autres, je ne prenais pas en compte leur possible inutilité du fait qu’elles arrivaient trop tard pour la version en cours et trop tôt pour la suivante (qui inclurait forcément de nombreux changements qui obligeraient à se remettre au travail). Au-delà de ça, je n’avais aucune idée de l’importance que ça prenait dans le processus de release : intégration, création de paquetages, tests de l’interface, suivi et résolution des bogues.

Comment cela m’a fait grandir

Tout cela a changé radicalement quand je me suis tournée vers un rôle plus professionnel. Subitement, je faisais la même chose mais d’une manière plus structurée. J’ai appris que faire cavalier seul comme j’en avais pris l’habitude n’était pas adapté quand on devait jongler avec deux ou trois versions planifiées. Il fallait être méticuleusement synchronisé avec les feuilles de route des projets. En travaillant sur une traduction d’interface de bureau, je devais aussi vérifier que le calendrier de traduction concordait avec celui du projet principal.

Les travaux devaient idéalement commencer immédiatement après le gel de tous les messages d’origine de l’interface. Les traducteurs pouvaient alors travailler librement jusqu’à l’échéance de la période de traduction, après quoi ils pouvaient marquer la traduction comme stable dans le dépôt

principal et, finalement, les paquetages pouvaient être générés. De plus, quelques distributions de systèmes d'exploitation se synchronisaient sur ce calendrier. Les traducteurs avaient donc la responsabilité supplémentaire de s'assurer que les pré-versions des systèmes d'exploitation embarquant ce bureau seraient un minimum testées afin de s'assurer que les traductions de l'interface avaient du sens et ne contenaient pas d'erreur.

Ce que j'aurais dû savoir

La transition ne fut pas aisée. Je fus soudain inondée par un flot d'informations que je devais gérer et par des tâches supplémentaires que je devais réaliser. Ce qui était au départ un passe-temps et plus important encore un anti-stress est devenu tout à coup une affaire sérieuse. En y repensant, je peux dire que cela m'a probablement aidée à comprendre le processus dans son intégralité étant donné que j'ai dû tout apprendre depuis le début. Ainsi armée de cette connaissance, je peux analyser des situations avec une meilleure compréhension de toutes leurs dimensions. Au moment où j'ai commencé à travailler sur les projets open *source* qui m'intéressaient, il y avait beaucoup moins de professionnels qui travaillaient à plein temps dans ce domaine. La plupart des contributeurs bénévoles travaillaient ailleurs la journée et voyaient ces projets comme un moyen d'alimenter les idées créatives qui s'étiolaient dans leurs tâches quotidiennes. Donc, beaucoup de nouveaux arrivants n'étaient jamais guidés vers une manière plus professionnelle d'organiser leurs projets. Ils ont grandi pour devenir merveilleusement doués dans ce qu'ils faisaient et ont finalement compris comment ils aimeraient équilibrer leur travail avec le reste de leurs activités.

Conclusion

Aujourd'hui, j'encadre les nouveaux arrivants et l'une des premières choses que je leur fais comprendre est comment et dans quelle partie du projet leur travail aura un impact. Élaborer un modèle de travail personnel est essentiel car cela permet de se construire un environnement où il est agréable de travailler. Cependant, avoir conscience de la structure qui est affectée par le travail inculque la discipline nécessaire pour pouvoir tenir bon face aux caprices.

(1) La *localisation* englobe tout le processus d'adaptation d'un produit logiciel ou documentaire à une région donnée. Cela comprend la traduction dans la langue de la région mais aussi l'adaptation aux normes, à la culture et aux besoins spécifiques de cette région du monde.

(2) Il s'agit de la publication d'un logiciel, sa mise à la disposition du public.

Apprendre à déléguer (Libres conseils 19/42)

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

Traduction Framalang : Nyx, lamessen, Sphinx, peupleLà, lerouge, Sky, Julius22, Astalaseven, Alpha, HgO, michel, Sputnik, goofy, HanX, KoS

Ne vous inquiétez pas, faites confiance

Shaun McCance

Shaun McCance est impliqué dans la documentation de GNOME depuis 2003 en tant que rédacteur, chef de la communauté et développeur d'outils. Il a passé la plupart de ce temps à se demander comment inciter davantage de personnes à écrire une documentation de meilleure qualité, avec un certain succès à long terme. Il propose son expérience de la documentation communautaire à travers sa société de conseil, Syllogist.

Alors que je m'apprêtais à écrire cet article, il s'est passé quelque chose d'énorme : GNOME 3 est sorti. C'était la première version majeure de GNOME depuis neuf ans. Tout était différent et toute la documentation existante devait être réécrite. Au même moment, nous changions notre façon de l'écrire. Nous avions jeté nos vieux manuels et étions repartis sur une nouvelle base, avec un système d'aide dynamique par sujet, en utilisant Mallard.

Quelques semaines avant la sortie, une partie d'entre nous s'est réunie pour élaborer la documentation. Nous passions nos journées à travailler, à planifier, à écrire et à réviser. Nous avons écrit des centaines de pages malgré les changements incessants liés aux ultimes modifications du logiciel. Nous avions des contributeurs en ligne qui proposaient de nouvelles pages et corrigeaient ce qui existait déjà. Je n'avais jamais vu notre équipe de documentation aussi productive.

À quoi avons-nous finalement abouti ? Beaucoup de facteurs sont entrés en jeu, et je pourrais écrire un livre entier sur les nuances de la documentation *open source*. Mais ce que j'ai fait de plus important a été de m'effacer et de laisser les autres faire le travail. J'ai appris à déléguer ; et à

déléguer dans les règles de l'art.

Revenons huit ans en arrière. J'ai commencé à m'impliquer dans la documentation de GNOME en 2003. Je n'avais pas vraiment d'expérience en tant que rédacteur technique à cette époque. Mon emploi m'amenait à travailler sur des outils de publication et j'ai commencé à travailler sur les outils et sur le visualiseur d'aide utilisés par la documentation de GNOME. Peu de temps après, je me suis retrouvé à la rédaction de la documentation.

En ce temps-là, la majeure partie de notre documentation était entre les mains de rédacteurs techniques professionnels de chez Sun. Ils s'occupaient d'un manuel, l'écrivaient, le relisaient et l'envoyaient sur notre dépôt CVS. Après quoi nous pouvions tous le regarder, y apprendre quelque chose et lui apporter des corrections. Mais il n'existait pas d'efforts concertés pour impliquer les gens dans le processus d'écriture.

Ce n'est pas que les rédacteurs de Sun essayaient de protéger ou de cacher quoi que ce soit. Ils étaient avant tout rédacteurs techniques. Ils connaissaient leur travail et le faisaient bien. D'autres personnes auraient pu les remplacer pour d'autres manuels mais ils auraient écrit leurs travaux d'une manière habituelle. Utiliser un groupe de collaborateurs novices, aussi enthousiastes soient-ils, pour chaque page, revient à perdre un temps inimaginable sur des détails. C'est tout simplement contre-productif.

De manière inévitable, le vent a tourné chez Sun et leurs rédacteurs techniques ont été affectés à d'autres projets. Cela nous a laissés sans nos rédacteurs les plus prolifiques, ceux qui disposaient des meilleures connaissances. Pire que cela, nous étions laissés sans communauté et personne n'était là pour ramasser les morceaux.

Il y a des idées et des processus standards dans le monde de

l’entreprise. J’ai travaillé dans le monde de l’entreprise. Je ne crois pas que quiconque remette ces idées en cause. Les gens font leur travail. Ils choisissent des missions et les terminent. Ils demandent aux autres de faire une relecture, mais ils n’ouvrent pas leur travail aux nouveaux venus et aux rédacteurs moins expérimentés. Les meilleurs rédacteurs écriront sans doute le plus.

Ces idées sont d’une plate évidence, mais elles échouent lamentablement dans un projet communautaire. Vous ne développerez jamais une communauté de contributeurs si vous faites tout vous-même. Dans un projet de logiciel, vous pouvez avoir des contributeurs compétents et suffisamment impliqués pour contribuer régulièrement. Dans la documentation, cela n’arrive presque jamais.

La plupart des gens qui s’essayent à la documentation ne le font pas parce qu’ils veulent être rédacteur technique ni même parce qu’ils aiment écrire. Ils le font parce qu’ils veulent contribuer. Et la documentation est la seule manière qu’ils trouvent accessible. Ils ne savent pas coder. Ils ne sont artistiquement pas doués. Ils ne maîtrisent pas assez une autre langue pour faire de la traduction. Mais ils savent écrire.

C’est là que les rédacteurs professionnels lèvent les yeux au ciel. Le fait que vous soyez instruit ne signifie pas que vous puissiez écrire une bonne documentation pour l’utilisateur. Il ne s’agit pas simplement de poser des mots sur le papier. Vous devez comprendre vos utilisateurs, ce qu’ils savent, ce qu’ils veulent, les endroits où ils cherchent. Vous avez besoin de savoir comment présenter l’information de façon compréhensible et savoir où la mettre pour qu’ils puissent la trouver.

Les rédacteurs techniques vous diront que la rédaction technique n’est pas à la portée de tous. Ils ont raison. Et c’est exactement pourquoi la chose la plus importante que les rédacteurs professionnels puissent faire pour la communauté

est de ne pas écrire.

La clé pour construire une communauté efficace autour de la documentation, c'est de laisser les autres prendre les décisions, faire le travail et en récolter eux-mêmes les fruits. Il ne suffit pas de leur donner du travail en continu. La seule solution pour qu'ils s'intéressent suffisamment et s'accrochent au projet, c'est qu'ils se sentent investis personnellement. Le sentiment de faire partie intégrante d'un projet est une source puissante de motivation.

Mais si vous ne travaillez qu'avec des rédacteurs débutants et que vous leur donnez tout le travail à faire, comment pouvez-vous avoir l'assurance que la documentation ainsi créée sera de qualité ? Une participation massive mais incontrôlée n'aboutit pas à de bons résultats. Le rôle d'un rédacteur expérimenté au sein de la communauté est d'être un professeur et un mentor. Vous devez leur apprendre comment rédiger.

Commencez par impliquer les gens tôt dans le planning. Planifiez toujours du bas vers le haut. La planification du haut vers le bas n'incite pas à la collaboration. Il est difficile d'impliquer les gens dans la réalisation d'une vue d'ensemble de haut niveau si tous n'ont pas la même perception de cette vue d'ensemble. Mais les gens sont capables de travailler sur des segments. Ils peuvent réfléchir à des sujets particuliers d'écriture, à des tâches que les gens réalisent, à des questions que les gens peuvent se poser. Ils peuvent regarder les forums de discussion et les listes de diffusion afin de voir ce que les utilisateurs demandent.

Écrivez vous-même quelques pages. Cela donne un exemple à imiter. Il faut ensuite répartir tout le reste du travail. Laissez à d'autres la responsabilité de rubriques ou de chapitres entiers. Précisez-leur clairement quelles informations ils doivent fournir, mais laissez-les écrire. C'est en forgeant qu'on devient forgeron.

Soyez constamment disponible pour les aider ou répondre aux questions. Au moins la moitié de mon temps consacré à la documentation est passée à répondre à des questions afin que les autres puissent effectuer leur travail. Quand des brouillons sont soumis, relisez-les et discutez des critiques et des corrections avec leurs auteurs. Ne vous contentez pas de corriger vous-même.

Cela vous laisse tout de même le gros du travail à faire. Les gens complètent les pièces du puzzle, mais c'est toujours vous qui les assemblez. Au fur et à mesure qu'ils acquièrent de l'expérience, les gens s'occuperont de pièces de plus en plus grandes. Encouragez-les à s'impliquer davantage. Donnez-leur davantage de travail. Faites en sorte qu'ils vous aident à aider plus de rédacteurs. La communauté fonctionnera toute seule.

Huit ans plus tard, GNOME a réussi à créer une équipe de documentation qui se gère elle-même, résout les problèmes, prend des décisions, produit une bonne documentation et accueille constamment de nouveaux contributeurs. N'importe qui peut la rejoindre et y jouer un rôle. Telle est la clé du succès pour une communauté *open source*.