Tous les autres pourraient avoir tort, mais c’est peu probable (Libres conseils 2/42)

Tous les autres pourraient avoir tort, mais c’est peu probable

par Evan Prodromou

Evan Prodromou est le fondateur de Wikitravel, StatusNet et du réseau social Open Source Identi.ca. Il participe aux logiciels Open Source depuis 15 ans en tant que développeur, écrit de la documentation et se distingue à l’occasion comme agitateur. Il vit au Québec, à Montréal.

La plus importante caractéristique du fondateur d’un projet Open Source, dans les premières semaines ou premiers mois avant de lancer son logiciel dans le monde, c’est une obstination de tête de mule face à l’écrasante évidence des faits. Si votre logiciel est si important, pourquoi personne ne l’a-t-il déjà écrit ? Peut-être que ce n’est même pas possible. Peut-être que personne d’autre que vous ne veut ce que vous êtes en train de faire. Peut-être que vous n’êtes pas assez bon pour le faire. Peut-être que quelqu’un l’a déjà fait et que vous n’êtes simplement pas assez malin pour le trouver avec Google.

Garder la foi à travers cette longue et sombre nuit est difficile ; seules les têtes de cochons opiniâtres et bornées peuvent y parvenir. Et nous arrivons à l’application de nos opinions de programmeur les plus fortement défendues. Quel est le meilleur langage de programmation à utiliser ? L’architecture de l’application ? Les standards d’écriture du code ? La licence logicielle ? Le système de gestion de version ? Si vous êtes le seul à travailler (ou à connaître !) le projet, vous devez décider, de façon unilatérale.

Quand vous vous lancez finalement, malgré tout, cette détermination bornée et cette forte opinion sont devenus préjudiciables, pas bénéfiques. Une fois que vous vous êtes lancé, vous aurez besoin de compétences tout à fait à l’opposé pour faire des compromis afin que votre logiciel soit davantage utile aux autres. Et

beaucoup de ces compromis vous sembleront vraiment mauvais.

Il est difficile de recevoir des avis d'« étrangers » (c.à.d. des personnes qui ne sont pas vous). D'abord parce qu'ils se focalisent sur des choses si triviales, sans importance (votre convention de nommage des variables par exemple, ou l'emplacement de certain boutons). Ensuite parce qu'ils ont invariablement tort . Après tout, si ce que vous avez fait n'est pas la bonne manière de faire, vous ne l'auriez pas fait ainsi dès le départ. Si votre façon de faire n'était pas la bonne, pourquoi votre code serait si populaire ?

Mais « mauvais » est relatif. Si faire un *mauvais* choix rend votre logiciel plus accessible aux utilisateurs finaux, ou aux « développeurs en aval » ou aux administrateurs ou les empaqueteurs, est-ce que ce n'est pas finalement juste ?

Et la nature de ce genre de commentaires et de contributions est généralement négative. Les retours de la communauté sont principalement des réactions, ce qui implique qu'elles sont critiques. Avez vous déjà rapporté un bug qui disait « J'aime beaucoup l'organisation du module hashtable.c » ou « Bravo d'avoir supprimé ce sous-sous-sous-menu » ? Les gens font un retour d'expérience car ils n'aiment pas la façon dont fonctionne votre logiciel à un instant T. Et ils ne sont pas toujours très diplomates à ce moment-là.

Il est difficile de répondre de façon positive à ce genre de retour. Nous engueulons parfois les expéditeurs sur nos listes des discussions de développement, ou fermons les rapports d'anomalies avec un rictus et un WONTFIX (NdT : NESERAPASRESOLU, employé dans les rapports de bug). Pire encore, nous nous retirons dans notre cocon, ignorant les suggestions externes ou les retours d'expérience, câlinant notre confortable code qui sied parfaitement à nos idées préconçues et à nos marottes.

Si le logiciel n'est que pour vous-même, vous pouvez garder le code source et les infrastructures qui l'entourent comme terrain de jeu personnel. Mais si vous voulez que votre logiciel soit utilisé, qu'il compte pour les autres personnes, qu'il change (peut-être) le monde, alors vous allez devoir construire une saine et solide communauté d'utilisateurs, de contributeurs principaux, d'administrateurs et de développeurs de modules. Les utilisateurs ont besoin d'avoir l'impression de posséder le logiciel, de la même façon que vous.

Il est difficile de se rappeler que chacune de ces voix dissidentes ne représente

qu'une infime minorité. Imaginez tous les gens qui entendent parler de votre logiciel qui ne prennent jamais le temps de l'essayer. Ceux qui le téléchargent mais ne l'installent jamais. Ceux qui l'installent, restent bloqués, et l'abandonnent en silence. Et ceux qui veulent vous faire un retour, mais qui ne trouvent pas votre système de rapport de bugs, listes de mails de développeurs, canaux IRC ou adresses mails personnelles. Étant donné les difficultés à faire passer leur message, il y a probablement une centaine de personnes qui veulent que des modifications soient faites pour une seule qui parvient à transmettre le message. Donc, être à l'écoute de ces voix, quand elle parviennent à vous atteindre, est essentiel.

Le responsable de projet est chargé de maintenir la vision et la finalité du logiciel. Nous ne pouvons vaciller, en faisant des allers et retours basés sur tel ou tel courriel d'utilisateur pris au hasard. Et s'il y a un principe de base en jeu, alors, bien sûr, il est important de garder cette base stable. Personne d'autre que le responsable de projet ne peut le faire.

Mais nous devons réfléchir : y a-t-il des questions non fondamentales qui puissent rendre le logiciel plus accessible, plus facile d'utilisation ? Finalement, la mesure de notre travail est dans la façon dont nous touchons les utilisateurs, comment notre logiciel est utilisé, et ce pourquoi il est utilisé. À quel point notre idée personnelle importe-t-elle « vraiment » pour le projet et pour la communauté ? Quelle part est uniquement ce que le responsable aime, personnellement ? Si ces problèmes non essentiels existent, alors il faut réduire les désaccords, répondre aux demandes, et faire les changements. Le projet n'en sera que meilleur pour tout le monde.

Tous les autres pourraient avoir tort, mais c'est peu probable

par Evan Prodromou

Evan Prodromou est le fondateur de Wikitravel, StatusNet et du réseau social Open Source Identi.ca. Il participe aux logiciels Open Source depuis 15 ans en tant que développeur, écrit de la documentation et se distingue à l'occasion comme agitateur. Il vit au Québec, à Montréal.

La plus importante caractéristique du fondateur d'un projet Open Source, dans

les premières semaines ou premiers mois avant de lancer son logiciel dans le monde, c'est une obstination de tête de mule face à l'écrasante évidence des faits. Si votre logiciel est si important, pourquoi personne ne l'a-t-il déjà écrit ? Peut-être que ce n'est même pas possible. Peut-être que personne d'autre que vous ne veut ce que vous êtes en train de faire. Peut-être que vous n'êtes pas assez bon pour le faire. Peut-être que quelqu'un l'a déjà fait et que vous n'êtes simplement pas assez malin pour le trouver avec Google.

Garder la foi à travers cette longue et sombre nuit est difficile ; seules les têtes de cochons opiniâtres et bornées peuvent y parvenir. Et nous arrivons à l'application de nos opinions de programmeur les plus fortement défendues. Quel est le meilleur langage de programmation à utiliser ? L'architecture de l'application ? Les standards d'écriture du code ? La licence logicielle ? Le système de gestion de version ? Si vous êtes le seul à travailler (ou à connaître !) le projet, vous devez décider, de façon unilatérale.

Quand vous vous lancez finalement, malgré tout, cette détermination bornée et cette forte opinion sont devenus préjudiciables, pas bénéfiques. Une fois que vous vous êtes lancé, vous aurez besoin de compétences tout à fait à l'opposé pour faire des compromis afin que votre logiciel soit davantage utile aux autres. Et beaucoup de ces compromis vous sembleront vraiment mauvais.

Il est difficile de recevoir des avis d'« étrangers » (c.à.d. des personnes qui ne sont pas vous). D'abord parce qu'ils se focalisent sur des choses si triviales, sans importance (votre convention de nommage des variables par exemple, ou l'emplacement de certain boutons). Ensuite parce qu'ils ont invariablement tort . Après tout, si ce que vous avez fait n'est pas la bonne manière de faire, vous ne l'auriez pas fait ainsi dès le départ. Si votre façon de faire n'était pas la bonne, pourquoi votre code serait si populaire ?

Mais « mauvais » est relatif. Si faire un *mauvais* choix rend votre logiciel plus accessible aux utilisateurs finaux, ou aux « développeurs en aval » ou aux administrateurs ou les empaqueteurs, est-ce que ce n'est pas finalement juste ?

Et la nature de ce genre de commentaires et de contributions est généralement négative. Les retours de la communauté sont principalement des réactions, ce qui implique qu'elles sont critiques. Avez vous déjà rapporté un bug qui disait « J'aime beaucoup l'organisation du module hashtable.c » ou « Bravo d'avoir

supprimé ce sous-sous-sous-menu » ? Les gens font un retour d'expérience car ils n'aiment pas la façon dont fonctionne votre logiciel à un instant T. Et ils ne sont pas toujours très diplomates à ce moment-là.

Il est difficile de répondre de façon positive à ce genre de retour. Nous engueulons parfois les expéditeurs sur nos listes des discussions de développement, ou fermons les rapports d'anomalies avec un rictus et un WONTFIX (NdT : NESERAPASRESOLU, employé dans les rapports de bug). Pire encore, nous nous retirons dans notre cocon, ignorant les suggestions externes ou les retours d'expérience, câlinant notre confortable code qui sied parfaitement à nos idées préconçues et à nos marottes.

Si le logiciel n'est que pour vous-même, vous pouvez garder le code source et les infrastructures qui l'entourent comme terrain de jeu personnel. Mais si vous voulez que votre logiciel soit utilisé, qu'il compte pour les autres personnes, qu'il change (peut-être) le monde, alors vous allez devoir construire une saine et solide communauté d'utilisateurs, de contributeurs principaux, d'administrateurs et de développeurs de modules. Les utilisateurs ont besoin d'avoir l'impression de posséder le logiciel, de la même façon que vous.

Il est difficile de se rappeler que chacune de ces voix dissidentes ne représente qu'une infime minorité. Imaginez tous les gens qui entendent parler de votre logiciel qui ne prennent jamais le temps de l'essayer. Ceux qui le téléchargent mais ne l'installent jamais. Ceux qui l'installent, restent bloqués, et l'abandonnent en silence. Et ceux qui veulent vous faire un retour, mais qui ne trouvent pas votre système de rapport de bugs, listes de mails de développeurs, canaux IRC ou adresses mails personnelles. Étant donné les difficultés à faire passer leur message, il y a probablement une centaine de personnes qui veulent que des modifications soient faites pour une seule qui parvient à transmettre le message. Donc, être à l'écoute de ces voix, quand elle parviennent à vous atteindre, est essentiel.

Le responsable de projet est chargé de maintenir la vision et la finalité du logiciel. Nous ne pouvons vaciller, en faisant des allers et retours basés sur tel ou tel courriel d'utilisateur pris au hasard. Et s'il y a un principe de base en jeu, alors, bien sûr, il est important de garder cette base stable. Personne d'autre que le responsable de projet ne peut le faire.

Mais nous devons réfléchir : y a-t-il des questions non fondamentales qui puissent rendre le logiciel plus accessible, plus facile d'utilisation ? Finalement, la mesure de notre travail est dans la façon dont nous touchons les utilisateurs, comment notre logiciel est utilisé, et ce pourquoi il est utilisé. À quel point notre idée personnelle importe-t-elle « vraiment » pour le projet et pour la communauté ? Quelle part est uniquement ce que le responsable aime, personnellement ? Si ces problèmes non essentiels existent, alors il faut réduire les désaccords, répondre aux demandes, et faire les changements. Le projet n'en sera que meilleur pour tout le monde.