Des routes et des ponts (5) – vers l'open source

Voici un nouveau chapitre de l'ouvrage de ***Nadia Eghbal*** Des routes et des ponts *que le groupe Framalang vous traduit semaine après semaine (si vous avez raté les épisodes précédents). Elle brosse un rapide historique qui permet de distinguer* Libre *et* open source *puis établit une liste d'avantages du code dont les sources sont ouvertes et librement modifiables.*

Une rapide histoire des logiciels publiquement disponibles et de leurs créateurs

Traduction Framalang : goofy, Julien/Sphinx, jums, xi, woof, Asta, Edgar Lori, jbm, Mika, penguin

Bien que nous ayons utilisé l'expression « *free software* » pour désigner des logiciels qui ne coûtaient rien à leurs utilisateurs, il faudrait plutôt employer l'expression « logiciel libre ». Cette expression aux riches connotations fait référence en particulier aux propriétés des licences avec lesquelles les logiciels sont publiés. Les partisans du logiciel libre soulignent le fait que « *free* » doit être compris sous l'angle de la liberté politique et non sous celui de la gratuité. Parfois, c'est le terme espagnol « *libre* » qui est utilisé pour marquer cette distinction (à la différence de « *gratis* » qui signifie gratuit).

Pendant les années 1970, lorsque les ordinateurs n'en étaient qu'à leurs balbutiements, les développeurs devaient construire leurs propres ordinateurs et écrire eux-mêmes des logiciels adaptés. Les logiciels n'étaient pas encore standardisés et n'étaient pas considérés comme des produits rentables.

En 1981, IBM a présenté le « IBM PC » pour « Personal Computer » (N.D.T. « ordinateur personnel »), qui a permis au grand public d'accéder au matériel informatique. En quelques années, les ordinateurs construits sur mesure tombèrent en déclin au fur et à mesure que tout le monde adoptait le standard IBM. IBM est ainsi devenu l'ordinateur le plus présent au sein d'un marché

fortement fracturé : en 1986, IBM avait conquis plus de la moitié du marché des ordinateurs personnels.

Avec la venue de matériel standardisé est apparue la possibilité de créer des logiciels standardisés. Soudain, tout le monde avait pour objectif de créer un business autour des logiciels. IBM a engagé une société inconnue à l'époque sous le nom de Microsoft pour écrire le système d'exploitation de son nouveau PC. Ce système d'exploitation, MS-DOS, fut publié en 1981. D'autres sociétés lui emboîtèrent le pas, proposant des logiciels sous licences commerciales. Ces licences empêchaient l'utilisateur de copier, modifier ou redistribuer les logiciels.

Il existe encore aujourd'hui de nombreux logiciels propriétaires comme Adobe Photoshop, Microsoft Windows, ou GoToMeeting par exemple. Alors que ces programmes propriétaires peuvent générer du profit pour les entreprises qui créent et distribuent ces produits, leurs restrictions limitent leur portée et leur diffusion. Toute modification apportée au design ou à la conception du programme doit provenir de l'entreprise elle-même. De plus, les logiciels propriétaires sont chers, ils coûtent souvent plusieurs centaines de dollars et n'autorisent l'acheteur dûment identifié à utiliser qu'une seule et unique copie.

Naturellement, certains informaticiens se sont sentis préoccupés par la direction fermée et propriétaire que prenaient les logiciels, estimant que cela nuisait au véritable potentiel du logiciel. Richard Stallman, un programmeur au laboratoire d'intelligence artificielle du MIT, a particulièrement ressenti la nécessité pour le logiciel d'être libre et modifiable.

Au cours des années qui suivirent, comme plusieurs de ses collègues se mettaient à travailler sur des projets de logiciels propriétaires, Stallman a estimé qu'il ne pouvait ignorer la situation plus longtemps. En 1983, il a lancé GNU, un système d'exploitation libre, et ce faisant, a déclenché ce qui est devenu le « mouvement du logiciel libre », qui a galvanisé un groupe de personnes qui croyaient que les logiciels pourraient avoir une plus grande portée et bénéficier à la société si ceux-ci étaient mis à disposition librement. Stallman a fondé plus tard la Free Software Foundation en 1985, afin de soutenir GNU ainsi que d'autres projets de logiciels libres.

Gnou par Benjamin Hollis (CC BY 2.0)

La Free Software Foundation définit le logiciel libre comme « un logiciel qui donne à l'utilisateur la liberté de le partager, l'étudier et le modifier ». GNU définit quatre libertés associées à de tels logiciels :

> Un programme est un logiciel libre si vous, en tant qu'utilisateur de ce programme, avez les quatre libertés essentielles :
>
> - la liberté d'exécuter le programme comme vous voulez, pour n'importe quel usage (liberté 0) ;
> - la liberté d'étudier le fonctionnement du programme, et de le modifier pour qu'il effectue vos tâches informatiques comme vous le souhaitez (liberté 1) ; l'accès au code source est une condition nécessaire ;
> - la liberté de redistribuer des copies, donc d'aider votre voisin (liberté 2) ;
> - la liberté de distribuer aux autres des copies de vos versions modifiées (liberté 3) ; en faisant cela, vous donnez à toute la communauté une possibilité de profiter de vos changements ; l'accès au code source est une condition nécessaire.

Le mouvement du logiciel libre a été et continue d'être profondément engagé

dans la défense d'intérêts sociaux. En 1998, lorsque Netscape libéra le code source de son navigateur populaire, le débat commença à passer de la politique à la technologie.

Certains technologues pensaient que se concentrer sur les bénéfices pratiques des logiciels libres permettrait de diffuser le message associé à un public plus large.

Ils ont par exemple souligné que le logiciel libre était moins cher à créer et qu'il permettait d'obtenir une meilleure qualité car le public pouvait trouver des bogues et contribuer en proposant des correctifs. Ce type de pragmatisme se détachait de l'obligation morale exprimée par Stallman et ses partisans quant à l'obligation de promouvoir le logiciel libre. Ces technologues se sont réunis à Palo Alto pour une séance de discussion stratégique.

Christine Peterson, une spécialiste des nanotechnologies qui était présente suggéra l'expression « open source ».

Peu de temps après, deux personnes qui assistaient aussi à cette rencontre, Bruce Perens et Eric Raymond, créèrent l'Open Source Initiative.

Un logiciel dont le code source est disponible publiquement sera qualifié d'« *open source* ». C'est un peu comme avoir une voiture et être capable d'ouvrir le capot pour connaître comment elle fonctionne plutôt que d'avoir le moteur verrouillé et inaccessible. Les licences *open source* incluent toujours des clauses qui permettent au public d'utiliser, de modifier et de redistribuer le code. Sous cet angle, il n'y pas de différence juridique entre les licences libres et les licences *open source*. En fait, certains font référence à l'*open source* comme une campagne de publicité pour le logiciel libre.

Cependant, la distinction la plus importante entre ces mouvements reste la culture qu'ils ont fait naître. Le mouvement du logiciel *open source* s'est écarté des aspects socio-politiques du mouvement du logiciel libre pour se concentrer sur les bénéfices pratiques du développement logiciel et encourager des applications créatives et commerciales plus larges. À ce propos, Stallman a écrit :

> *« l'open source est une méthodologie de développement ;*
>
> *le logiciel libre est un mouvement de société. »*

Bien que « logiciel libre » et « logiciel *open source* » soient souvent discutés ensemble, ils sont politiquement distincts, le premier étant plus étroitement lié à l'éthique et le second au pragmatisme (dans la suite de cet ouvrage on utilisera le terme « *open source* » afin de souligner son rôle essentiel dans l'infrastructure logicielle.) L'*open source* a ouvert un espace permettant l'émergence de différents styles et façons de développer du logiciel, libérés des complexités éthiques. Une organisation peut rendre son code public, mais n'accepter des changements que de certains contributeurs. Une autre organisation peut exiger que le code soit développé en public et accepter des changements de n'importe qui, de manière à ce que davantage de personnes puissent prendre part au processus. En 1997, Raymond a écrit un essai influent intitulé La cathédrale et le bazar (publié plus tard sous la forme d'un livre, en 1999) qui explore ces divers modes de développement.

Aujourd'hui, l'*open source* s'est répandue dans le monde du logiciel pour un certain nombre de raisons, liées à la fois à l'efficacité et au coût. C'est aussi comme cela qu'est bâtie une bonne partie de notre infrastructure numérique. Nous avons discuté de la façon dont la disponibilité de ces logiciels a bénéficié à toute la société, mais l'*open source* a aussi beaucoup apporté à ses créateurs.

L'*open source* revient moins cher à créer

Avant que les logiciels *open source* n'existent, les entreprises high-tech considéraient les programmes comme n'importe quel autre produit payant : une équipe d'employés développait le produit en interne puis on le vendait au grand public. Ce qui représentait un modèle économique très clair, mais impliquait aussi des coûts de développement accrus. Les logiciels propriétaires nécessitent une équipe payée à plein temps pour assurer le développement, ce qui inclut des développeurs, des designers, des commerciaux et des juristes. Il est bien moins coûteux de simplement confier le développement à une communauté de développeurs bénévoles qui conçoivent et assurent la maintenance du produit.

L'*open source* est plus facile à diffuser

On a plus envie d'adopter un logiciel dont l'usage est gratuit et de le modifier, plutôt qu'un logiciel dont la licence coûte des centaines de dollars et qui a été développé dans une boîte noire. Non seulement les développeurs vont vouloir l'utiliser sans frais, mais ils pourraient même inciter leurs amis à l'utiliser eux aussi, ce qui va amplifier sa diffusion.

L'*open source* est plus ouvert à la personnalisation

Les logiciels open source sont copiables et adaptables aux besoins de chacun, avec différents degrés de permission. Si un développeur veut améliorer un logiciel existant, il ou elle peut copier le projet et le modifier (une pratique appelée « forker » en franglais).

Beaucoup de projets à succès ont commencé comme une modification de logiciels existants, par exemple WordPress (gestionnaire de contenu utilisé par 23% des sites web dans le monde), PostgreSQL (l'une des bases de données parmi les plus populaires et dont l'adoption est croissante dans le monde entier), Ubuntu (un système d'exploitation) et Firefox (un des navigateurs web parmi les plus populaires). Dans le cas de WordPress, le logiciel a été *forké* depuis un projet existant appelé b2 (aussi connu sous le nom de cafelog). Deux développeurs, Matt Mullenweg et Mike Little, ont décidé qu'ils souhaitaient une meilleure version de b2 et ont donc *forké* le projet.
Mullenberg a décidé de copier b2, plutôt qu'un autre projet appelé TextPattern, car les licences b2 étaient plus permissives. Son idée d'origine, de 2003, est décrite ci-dessous :

> *Que faire ? Bon, TextPattern ressemble à tout ce que je rêve d'avoir, mais ça n'a pas l'air d'être sous une licence suffisamment en accord avec mes principes. Heureusement, b2/cafelog est sous GPL [GNU General Public Licence, une licence de logiciel libre], ce qui veut dire que je peux utiliser les lignes de code existantes pour créer un fork/une copie. […]*
> *Ce travail ne sera jamais perdu, car si je disparais de la surface de la Terre dans un an, tout le code que j'aurai écrit sera accessible par tout le monde ; et si quelqu'un d'autre veut continuer le travail, libre à lui.*

Si le logiciel était développé dans un environnement fermé et propriétaire, les développeurs n'auraient aucune possibilité de le modifier, à moins de travailler dans l'entreprise propriétaire. S'ils essayaient de réaliser leur propre version qui imite l'original, ils s'exposeraient à des poursuites en lien avec la propriété intellectuelle. Avec les logiciels *open source*, le développeur peut simplement modifier le logiciel lui-même et le distribuer publiquement, comme l'a fait Mullenweg. Les logiciels *open source* permettent ainsi une prolifération rapide des idées.

L'*open source* facilite l'adaptation des employés

Il faut du temps pour étudier une ressource logicielle, qu'il s'agisse d'un nouveau langage de programmation ou d'un nouveau *framework*. Si toutes les entreprises utilisaient leurs propres outils propriétaires, les développeurs auraient moins envie de changer d'entreprise, parce que leurs compétences techniques ne seraient applicables que sur leur lieu de travail actuel.

Il leur faudrait de nouveau apprendre à utiliser les outils propres à leur nouveau lieu de travail.

Quand les entreprises utilisent la technologie *open source*, un développeur a un ensemble de compétences réutilisables, ce qui lui donne plus de libertés pour travailler là où il préfère. Par exemple, de nombreuses entreprises utilisent le même langage de programmation Ruby pour leurs logiciels. De plus, si le produit des entreprises lui-même est *open source*, la production appartient autant au développeur qu'à l'entreprise. Le développeur peut emporter son travail avec lui s'il décide de quitter l'entreprise (alors qu'il pourrait par exemple être au contraire limité par une clause de confidentialité si le le code était propriétaire). Tous ces bénéfices offrent plus de moyens d'actions aux employés par rapport à ce que ces derniers auraient eu avec un logiciel propriétaire. De nos jours, de nombreuses entreprises mettent en avant leur utilisation de logiciels *open source* comme tactique de recrutement, parce que cette utilisation favorise le développeur.

L'*open source* est potentiellement plus stable et plus sûre.

Théoriquement, quand un projet de logiciel a de nombreux contributeurs et une communauté florissante, le code devrait être moins vulnérable aux failles de sécurité et aux interruptions de service. En effet, dans ce cas, on devrait avoir plus de personnes révisant le code, cherchant des bugs et résolvant tous les problèmes repérés.

Dans un environnement de logiciel propriétaire au contraire, seule l'équipe en charge du développement du code verra ce dernier. Par exemple, au lieu de 20 personnes pour examiner le code d'Oracle, un projet *open source* populaire pourrait avoir 2000 volontaires qui recherchent les failles du code (remarquons que cette croyance n'est pas toujours en accord avec la réalité, et a parfois créé le problème inverse : on a pu surestimer le nombre de personnes vérifiant des logiciels *open source*, alors même qu'en réalité personne n'en prenait la responsabilité. Ceci sera discuté dans une prochaine section).

Le logiciel *open source* a clairement certains avantages. Comment ces projets s’inscrivent-ils collectivement dans un écosystème plus large ?