Software Heritage : pourquoi et comment construire l'archive universelle du code source

Roberto Di Cosmo [1]

Le logiciel est aujourd'hui présent partout autour de nous : il est le moteur de notre industrie, le carburant de l'innovation, l'instrument essentiel que nous utilisons pour communiquer, nous entretenir, réaliser tout type de transaction et opération, nous organiser en société et former nos opinions politiques. Il est aussi le médiateur indispensable qui nous permet d'accéder à toute l'information numérique, et le pilier de la science moderne [2].

En un mot, on peut dire que le logiciel est en train de devenir la composante principale de notre patrimoine scientifique, technique et culturel.

En regardant de plus près, quand on parle de logiciel, la vraie connaissance n'est pas contenue dans les exécutables, qui sont prévus pour une exécution directe sur la machine spécifique, et peuvent, une fois optimisés, être incompréhensibles pour un être humain. Elle est contenue dans le « code source » des logiciels qui, selon la définition très bien conçue utilisée dans la licence GPL, est « la forme préférée pour un développeur pour apporter une modification au programme » [GNU91].

Oui, le code source est une forme unique de connaissance qui est à la fois faite pour être comprise par un être humain, le développeur, et pour être exécutée sur une machine. Il s'agit d'un objet numérique très spécial : il est souvent vivant, modifié régulièrement pour l'adapter à de nouveaux besoins et, si on veut vraiment comprendre son évolution, avoir accès à son historique de développement est essentiel.

Malgré des analyses relativement anciennes et très pertinentes de l'importance du code source, comme celle faite par Len Shustek dans son très bel article de

1. Inria et Université Paris Diderot, www.dicosmo.org

2. Il n'y a pas un domaine scientifique qui n'a pas besoin de logiciels [VNMN14].

CC BY NC

2006 [Shu06], ou par Donald Knuth [Knu84], jusqu'ici peu d'attention a été portée à la préservation du patrimoine contenu dans nos codes sources.

On archive beaucoup de documents numériques : images, vidéos, musique, textes, pages web, même des binaires exécutables ; le code source, non. Pourtant, le besoin est bien là, pour de multiples raisons. Voyons-en les trois plus évidentes.

La diaspora logicielle

Aujourd'hui, avec l'essor spectaculaire du logiciel libre, des millions de projets logiciels sont construits en utilisant des plateformes de développement, comme GitHub, GitLab.com, Bitbucket, l'ancien Sourceforge et bien d'autres, sans compter la myriade de forges plus ou moins institutionnelles éparpillées partout sur la planète, ou les personnes qui distribuent encore leurs sources comme des archives téléchargeables à partir de leur page web. Et pour compliquer le tout, pendant la vie d'un logiciel, les développeurs ont souvent tendance à changer de plateforme de développement, en passant de l'une à l'autre selon la mode du moment ou l'évolution des besoins et des habitudes des contributeurs.

Une fois une version particulière d'un logiciel stabilisée, se pose le problème de le distribuer. Là aussi, la situation n'est pas plus simple : certains développeurs utilisent la plateforme de développement pour la distribution (la plupart des forges permettent de faire ça). D'autres communautés ont leurs archives dédiées, comme CPAN, CTAN, CRAN, etc. Et puis il y a les différents systèmes de paquets, qui ont aussi leurs propres copies d'un code source qui a été récupéré ailleurs.

Se retrouver au milieu de ce véritable tas de spaghetti n'est pas chose facile : il nous manque cruellement un point d'entrée unique où l'on puisse retrouver et suivre l'évolution du développement du code source de tous les logiciels publiquement disponibles, et qui retrace toutes les plateformes de distribution.

Notre code source est fragile

Nous savons depuis longtemps que l'information numérique est fragile : erreur humaine, pannes matérielles, incendies, piratages, peuvent facilement détruire des données précieuses, et cela s'applique tout aussi bien au code source. C'est bien pour cela que l'on se doit d'effectuer des sauvegardes régulières.

Pour qui développe du logiciel libre, en utilisant une des plateformes de développement gratuites, privées ou institutionnelles, ce problème pouvait sembler lointain : la charge de la préservation revenait aux plateformes, pas aux auteurs des logiciels.

Mais courant 2015, deux plateformes de développement de logiciel libre très connues, Gitorious et Google Code, ont annoncé pour différentes raisons qu'elles allaient fermer leurs portes. Plus d'un million et demi de projets ont dû trouver un nouvel hébergement depuis, et cela dans un temps extrêmement court pour ce qui concerne Gitorious.

Cela nous a fait comprendre que la mission de préservation ne peut pas être déléguée à des entités qui n'en font pas leur mission première. Il nous manque une *archive* qui assume cette mission, et qui nous garantisse que si le code source d'un logiciel n'est plus disponible sur une plateforme de développement, ou si la plateforme elle-même disparaît, on puisse quand même le retrouver dans l'archive.

Un grand instrument de recherche sur le code source des logiciels

Avec l'importance grandissante du logiciel, qui pénètre toutes les strates de notre société et nos infrastructures, nous savons tous combien il est essentiel de se donner les moyens pour améliorer la qualité, la sûreté et la sécurité de nos codes sources.

Pourtant, nous manquons cruellement d'un grand instrument de recherche qui permette d'analyser l'ensemble du corpus de code source disponible, avec son historique de développement : pour cela il faut que l'ensemble de l'information soit disponible sous une forme homogène qui permette d'appliquer des techniques diverses sur l'ensemble du corpus, indépendamment de l'origine de chaque source.

La mission de *Software Heritage*

Pour répondre à ces trois défis majeurs, a été lancé, avec le soutien initial fort et total d'Inria, *Software Heritage*, une initiative qui a pour mission très précisément de *récolter, organiser, préserver et rendre facilement accessible* l'ensemble du *code source* disponible publiquement sur la planète, indépendamment de ou et comment il a été développé ou distribué.

Le but affiché est de construire une infrastructure commune, qui permettra une multiplicité d'applications : bien sûr, préserver sur le long terme le code source contre les risques de destruction, mais aussi permettre des études à grande échelle sur le code et les processus de développement actuels, afin de les améliorer et préparer ainsi un futur meilleur.

Une tâche complexe

Il s'agit d'une tâche bien plus complexe qu'il n'y paraît : il suffit de passer en revue les défis qu'il faut relever, ne serait-ce que pour la phase de récolte des codes sources (et il y en a bien d'autres pour les autres phases).

Nous devons recenser les endroits ou le code source se trouve, qui vont des plateformes de développement très connues jusqu'à des archives déposées sur d'obscures pages web. Il n'y a pas de catalogue universel, nous devons le construire.

Nous devons appréhender les protocoles propres à chaque plateforme de développement, afin de lister leurs contenus, et maintenir à jour notre archive en suivant l'évolution dans les codes sources qui y sont hébergés. Il n'y a pas de standard, nous espérons en promouvoir un minimal dans le futur.

Nous devons lire l'historique de développement qui se retrouve dans une grande variété de systèmes de contrôle de version : git, mercurial, subversion, darcs, bazaar, cvs, ce ne sont que quelques exemples des outils que nous devons étudier afin d'importer l'historique dans notre modèle de données. Notre mission d'universalité nous oblige à les regarder tous, nous n'avons pas le luxe d'en imposer un comme peuvent le faire certaines plateformes de développement. Et il n'y a pas de standard pour les structures des systèmes de contrôle de version : nous devons construire un modèle de donnée qui puisse les capturer toutes.

Face à un tel défi, il est important que nous, informaticiens, nous saisissions de la mission : le code source est l'ADN de notre discipline, et il nous revient d'être en première ligne quand il s'agit de le préserver, l'étudier et le rendre meilleur.

Quelques principes

Accomplir la mission de *Software Heritage* est une tâche complexe, et comporte un engagement sur le long terme. Il est donc important, pour se donner les plus grandes chances de succès, de fonder ce travail sur des bases solides. Nous avons donc identifié des principes fondateurs, qui sont les suivants.

Logiciel libre, et transparence

Pour préserver de l'information à long terme il faut connaître le fonctionnement de tous les outils mis en œuvre : *Software Heritage* a donc fait le choix de n'utiliser que des composants en logiciel libre, et de publier l'ensemble de son propre code source comme logiciel libre, en toute transparence.

Réplication à tous les niveaux

On l'a vu en analysant les raisons de la fragilité de l'information numérique, il y a pléthore de menaces, allant des défaillances techniques à la malveillance, à de simples décisions économiques, voire même juridiques. Nous savons qu'on ne pourra pas éviter toutes ces menaces, et qu'il y aura des erreurs et des imprévus. Donc au lieu de prétendre construire un système sans erreurs, nous préférons travailler à mettre en place un système qui tolère les erreurs. Pour cela, nous cherchons à construire une infrastructure répliquée et diversifiée à tous les niveaux : un réseau de miroirs, possiblement avec une variété de technologies différentes, sous des contrôles administratifs et dans des juridictions différentes. Le choix de diffuser comme logiciel libre tout le code source de notre infrastructure a précisément pour vocation de faciliter la création de nouveaux miroirs par des acteurs divers.

Sans but lucratif, et multipartenaire

L'expérience a montré qu'une seule entité, aussi puissante soit-elle, n'apporte pas assez de garanties sur le long terme. Nous pensons que pour assurer la mission de *Software Heritage* il est indispensable de construire une fondation sans but lucratif ayant pour objectif précis la collecte, l'organisation, la préservation et la mise a disposition du patrimoine précieux qui est le code source des logiciels. Pour minimiser les risques d'avoir des points uniques de défaillance, cette fondation doit s'appuyer sur des partenaires divers, allant de la société civile à l'industrie et au gouvernement, et doit s'adresser à tous les domaines susceptibles de tirer profit de l'existence de cette infrastructure, allant de la préservation du patrimoine culturel à la recherche, à l'industrie et à l'éducation.

Où nous en sommes aujourd'hui

Conçue vers la fin 2014, soutenue et mise en route par Inria courant 2015, *Software Heritage* est aujourd'hui une infrastructure qui grandit jour après jour. Même si on est encore loin d'être exhaustifs, on a déjà réuni le plus grand corpus de code source disponible sur la planète, avec plus de 55 millions d'origines archivées, pour plus de trois milliards de fichiers sources uniques, chacun équipé d'un identifiant intrinsèque basé sur un hachage cryptographique. Il ne s'agit pas simplement d'une collection de répertoires mis les uns à côté des autres, mais d'un graphe de Merkle qui partage au maximum les fichiers, les dossiers et les *commits*, seule structure à même de passer à l'échelle à une époque où le développement collaboratif, sur des plateformes comme GitHub, passe par la création de copies entières (*forks*) du projet originaire.

Un formidable objet d'étude

L'archive de *Software Heritage* constitue déjà un objet unique d'étude : c'est la première fois que l'on dispose de l'ensemble du code source de millions de projets, avec tout leur historique de développement, entièrement stockés dans un graphe partagé, et avec un modèle de données uniforme, indépendamment du système de contrôle de version. On peut se poser beaucoup de questions sur la nature et les propriétés de ce graphe, sur comment y retrouver la phylogénie des projets qui ont subi la diaspora logicielle, sur les algorithmes efficaces pour effectuer des recherches de motifs à la fois dans le code et dans l'historique de développement.

Et l'infrastructure de *Software Heritage* elle-même pose des défis significatifs : quelle structure de données est la plus adaptée pour exploiter l'archive dans une optique de *big data* ou de *machine learning* ? Quelle structure distribuée serait la plus adaptée pour assurer une réplication efficace et résiliente ? S'agissant d'une archive qui ne permet pas d'effacer ou modifier, mais seulement d'ajouter du contenu, quel

est le meilleur compromis entre la consistance des données, la disponibilité et la tolérance au partitionnement dans le futur réseau de miroirs ?

Ce ne sont que quelques-unes des questions qui viennent immédiatement à l'esprit, mais il peut y en avoir bien d'autres.

Conclusion

Software Heritage n'est pas encore prête à ouvrir pleinement l'accès à son contenu (il faut des ressources considérables pour cela), mais on peut espérer que la recherche scientifique française, dont nous sommes fiers, s'empare d'ores et déjà de ces questions et profite de l'occasion unique qui s'ouvre à nous actuellement, et que des développeurs passionnés contribuent à l'évolution du projet [3].

Références

[GNU91] GNU. Gnu general public license, version 2, 1991. retrieved September 2015.

[Knu84] Donald E. Knuth. Literate programming. *Comput. J.*, 27(2) :97–111, 1984.

[Shu06] Leonard J. Shustek. What should we collect to preserve the history of software ? *IEEE Annals of the History of Computing*, 28(4) :110–112, 2006.

[VNMN14] Richard Van Noorden, Brendan Maher, and Regina Nuzzo. The top 100 papers. *Nature*, pages 550–553, October4 2014.

3. Pour s'informer sur l'état du projet, on peut explorer les sites web `www.softwareheritage.org`, `annex.softwareheritage.org` et `wiki.softwareheritage.org`, s'inscrire sur la liste de diffusion `https://sympa.inria.fr/sympa/info/swh-science` et contribuer au développement sur `forge.softwareheritage.org`.