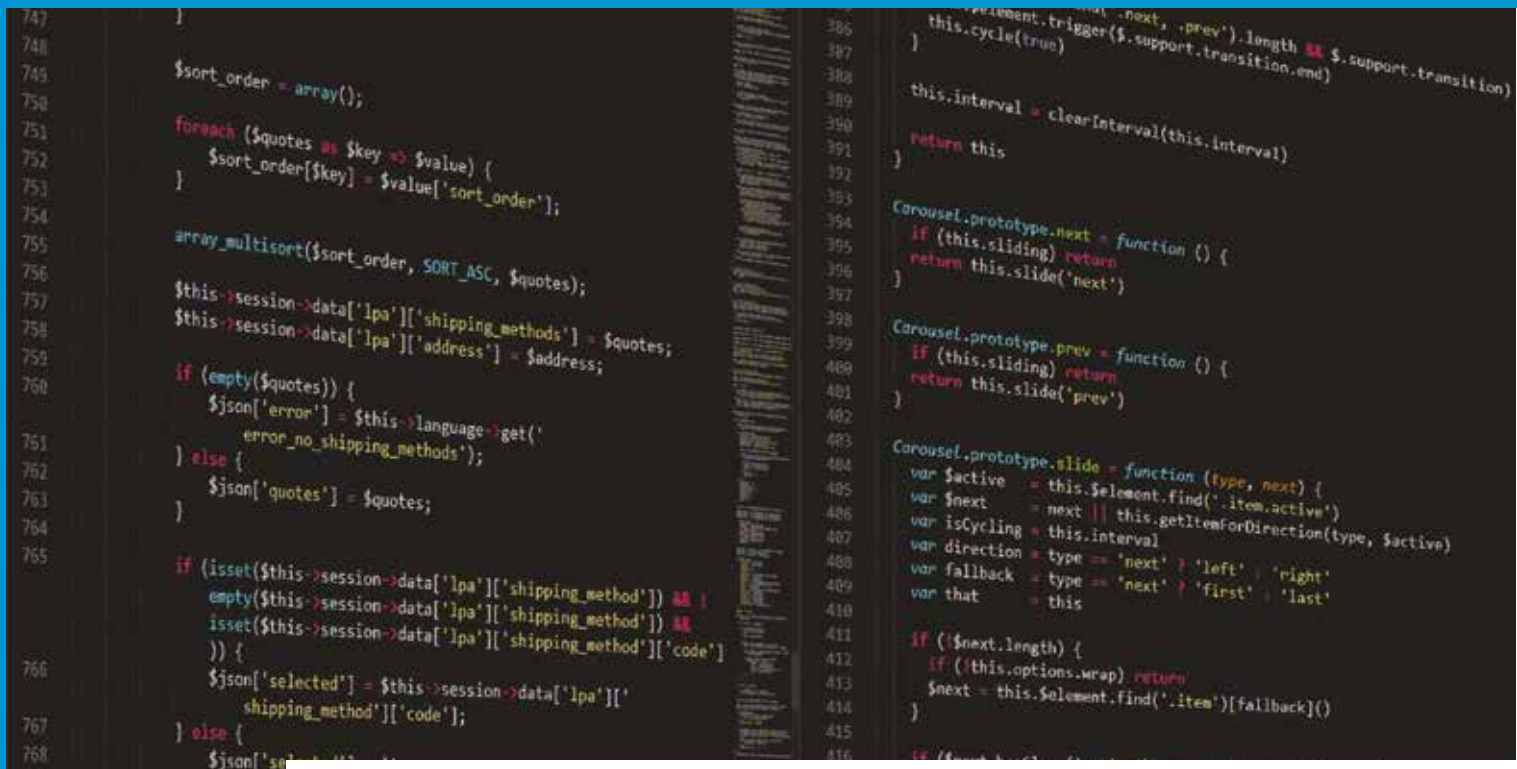


LE CODE SOURCE DES LOGICIELS

UN PATRIMOINE IMMATÉRIEL QU'IL EST ESSENTIEL DE PRÉSERVER !

LIGNE DE CODES HTML, 2015
James Osborne -Pixabay
Licence

Roberto DI COSMO
Software Heritage
Inria et IRIF,
Université de Paris
www.dicosmo.org

Apparu dans les années 1950[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 tous types de transactions et opérations, nous organiser en société et former nos opinions politiques. Mais ce n'est que la partie émergée de l'iceberg : le logiciel contrôle le système embarqué dans nos moyens de transport ou de communication, les échanges commerciaux et financiers. Il est au cœur des équipements et des dispositifs médicaux ; il assure le bon fonctionnement des réseaux de transport et de communication, des banques et des établissements financiers. Le logiciel est crucial dans le fonctionnement des organisations économiques, sociales et politiques, qu'elles soient publiques ou privées, que ce soit sur des terminaux mobiles ou sur le « cloud »[2]. Il est aussi le médiateur indispensable qui permet d'accéder à toute l'information numérique, et le pilier de la science moderne[3]. En un mot, on peut dire que le logiciel est en train de devenir la *composante principale* de notre patrimoine scientifique, technique et industriel.

Pour accéder à la vraie connaissance qui est contenue dans les logiciels, il ne suffit pas de disposer des programmes exécutables, qui sont prévus pour une exécution sur un ordinateur, et peuvent être incompréhensibles pour un être humain. La vraie connaissance est contenue dans le « code source »[4] des logiciels, qui selon la définition utilisée dans la licence GPL, est « la forme préférée pour un développeur pour apporter une modification au programme »[5].

Pour mieux comprendre cela, prenons le cas d'un programme tout simple, qui ne fait rien d'autre qu'afficher le message « Hello World » sur un écran quand il est lancé. Un simple utilisateur du logiciel voit simplement le message s'afficher :

```
Hello World
```

Or, imaginons que l'on souhaite modifier le programme pour que le message affiché soit différent, par exemple « Bonjour le monde » à la place de « Hello World ». Pour ce faire, on a essentiellement deux possibilités.

La première est de modifier le programme exécutable, dont un extrait est montré dans la figure suivante :

Executable (extrait)

```
4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
4004fb: c3
```

Il suffit d'un regard et on comprend tout de suite que ce ne sera pas une tâche facile : cette forme du logiciel est faite pour l'exécution sur une machine, et non pas pour qu'un être humain puisse la lire et la comprendre, et encore moins la modifier.

La deuxième option est de trouver la forme sous laquelle le développeur originaire a écrit le programme, qu'on appelle justement le « code source » du logiciel. Dans le cas de notre exemple très simple, cela ne fait que quelques lignes, que l'on voit dans la figure suivante :

Code source

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

Dans ce code source, il est très facile de comprendre ce qu'il faut changer pour modifier le message affiché, même pour une personne qui n'a jamais écrit un programme !

Le code source est ainsi une forme de connaissance vraiment spéciale, qui est à la fois faite pour être *comprise par un être humain*, le développeur, et qui peut être mécaniquement traduite dans une forme pour être *exécutée* directement sur une machine.

La terminologie même utilisée par la communauté informatique souligne cet aspect de création humaine : on utilise des « langages de programmation » pour « écrire » des logiciels. Comme Harold Habelson l'écrivait déjà en 1985, « les programmes doivent être écrits d'abord pour que d'autres êtres humains puissent les lire »[6].

Et dans le code source, les développeurs peuvent effectivement laisser un grand nombre d'annotations, appelées des « commentaires », qui n'ont aucune utilité pour la machine, et ne se retrouvent plus dans les exécutables. Ces commentaires sont des messages envoyés aux futurs développeurs pour les aider à mieux comprendre le programme, et comment le modifier, ou pas.

Un exemple que j'aime particulièrement est le code source du système de guidage de l'Apollo 11, qui était écrit dans un langage de programmation très proche de la machine, et avait donc besoin d'un grand nombre de commentaires pour pouvoir être compris. Dans l'image suivante, on voit un extrait de la routine principale d'allumage du réacteur, tel qu'elle se retrouve sauvegardée dans l'archive de Software Heritage.

La partie mise en évidence est particulièrement jolie : les concepteurs du programme font preuve de créativité pour dire aux autres développeurs que cette partie du code ne doit être modifiée que par ceux qui savent vraiment ce qu'ils font !

CODE SOURCE COMPLET DU PROGRAMME « HELLO WORLD »

1. Martin Campbell-Kelly, *Une histoire de l'industrie du logiciel*, Paris, Vuibert Informatique, 2003.

2. Serge Abiteboul et Gilles Dowek, *Le temps des algorithmes*, Paris, Éditions Le Pommier, 2017.

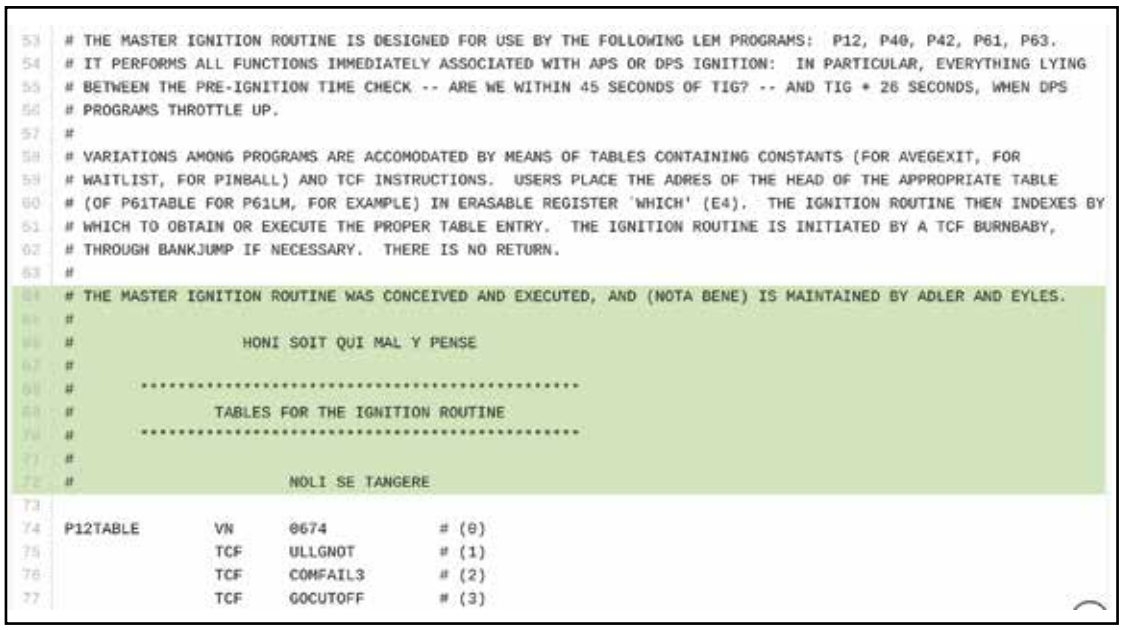
3. Il n'y a pas un domaine scientifique qui n'a pas besoin de logiciels, Richard Van Noorden, Brendan Maher, and Regina Nuzzo, « The top 100 papers », *Nature*, p. 550–553, October 4, 2014.

4. Le code source est une suite d'instructions logiques écrite dans un langage de programmation.

5. GNU91. Gnu general public license, version 2, 1991. Retrieved September 2015.

6. « Preface to the First Edition », in H. Abelson, G. J. Sussman, and J. Sussman, *The Structure and Interpretation of Computer Programs*, MIT Press, 2d ed. 1996, p. xxiii, traduction de l'auteur.

(COURT) EXTRAIT DU PROGRAMME « HELLO WORLD » EXÉCUTABLE



```
# THE MASTER IGNITION ROUTINE IS DESIGNED FOR USE BY THE FOLLOWING LEM PROGRAMS:  P12, P40, P42, P61, P63.
# IT PERFORMS ALL FUNCTIONS IMMEDIATELY ASSOCIATED WITH APS OR DPS IGNITION:  IN PARTICULAR, EVERYTHING LYING
# BETWEEN THE PRE-IGNITION TIME CHECK -- ARE WE WITHIN 45 SECONDS OF TIG? -- AND TIG + 26 SECONDS, WHEN DPS
# PROGRAMS THROTTLE UP.
#
# VARIATIONS AMONG PROGRAMS ARE ACCOMODATED BY MEANS OF TABLES CONTAINING CONSTANTS (FOR AVEGEXIT, FOR
# WAITLIST, FOR PINBALL) AND TCF INSTRUCTIONS.  USERS PLACE THE ADRES OF THE HEAD OF THE APPROPRIATE TABLE
# (OF P61TABLE FOR P61LM, FOR EXAMPLE) IN ERASABLE REGISTER 'WHICH' (E4).  THE IGNITION ROUTINE THEN INDEXES BY
# WHICH TO OBTAIN OR EXECUTE THE PROPER TABLE ENTRY.  THE IGNITION ROUTINE IS INITIATED BY A TCF BURNBABY,
# THROUGH BANKJUMP IF NECESSARY.  THERE IS NO RETURN.
#
# THE MASTER IGNITION ROUTINE WAS CONCEIVED AND EXECUTED, AND (NOTA BENE) IS MAINTAINED BY ADLER AND EYLES.
#
#                HONI SOIT QUI MAL Y PENSE
#
#       ********************************************
#             TABLES FOR THE IGNITION ROUTINE
#       ********************************************
#
#                     NOLI SE TANGERE

P12TABLE        VN      0674            # (0)
                TCF     ULLGNOT         # (1)
                TCF     COMFAIL3        # (2)
                TCF     GOCUTOFF        # (3)
```

ROUTINE PRINCIPALE D'ALLUMAGE D'APOLLO 11. ON REMARQUE LE COMMENTAIRE EN LATIN « NOLI SE TANGERE » (NE PAS TOUCHER) AU TOUT DÉBUT DES TABLES UTILISÉES PAR LA ROUTINE D'ALLUMAGE DES RÉACTEURS.
Source : https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa;lines=64-72;origin=https://github.com/chrislgarry/Apollo-11/

Le code source des logiciels est un objet numérique très spécial : il est souvent vivant, modifié régulièrement pour être adapté à de nouveaux besoins, et, pour comprendre son évolution, l'accès à son historique de développement est essentiel.

On est donc bien face, quand on parle du code source des logiciels, à une trace très importante de l'évolution technique, scientifique et industrielle des dernières décennies.

Il s'agit d'un patrimoine de grande valeur, et pourtant, 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 2006[7], dont la traduction française apparaît pour la première fois dans cette même revue, ou par Donald Knuth[8], jusqu'ici presque aucune attention n'a été portée à sa préservation.

Or, s'il y a des nombreuses initiatives pour préserver des documents numériques comme les images, les vidéos, la musique, des textes, des pages web, et même des binaires exécutables, le code source était jusqu'à maintenant le grand oublié.
Aujourd'hui, il est important et urgent d'agir, pour plusieurs raisons, dont voici les plus immédiatement visibles.

LA DIASPORA LOGICIELLE

Grâce à l'essor spectaculaire du logiciel libre[9], 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 les innombrables forges[10] disponibles un peu partout sur la planète. En outre, 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.

Pour la distribution des logiciels, opérations distinctes de leur développement, 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 Comprehensive P'erl Archive Network[11], Comprehensive TeX Archive Network[12], Comprehensive R Archive Network[13], etc. Enfin, différents systèmes de paquets ont aussi leurs propres copies d'un code source qui a été récupéré ailleurs.

Il manque clairement 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.

LE CODE SOURCE EST FRAGILE

L'information numérique est fragile, c'est un fait connu : 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 paraissait lointain : la responsabilité de la préservation revenait aux plateformes, pas aux auteurs des logiciels.

7. L. J. Shustek, « What should we collect to preserve the history of software? », *IEEE Annals of the History of Computing*, 28/4, 2006, p.110-112.

8. [Knu84] D. E. Knuth, « Literate programming », *Comput. J.*, 27/2, 1984, p. 97-111.

9. Un logiciel est libre lorsqu'il est accessible gratuitement et qu'il est possible de l'exécuter, d'étudier son fonctionnement, de l'améliorer (grâce à la transparence de son code), d'en redistribuer des copies et de publier les améliorations et tout cela sans restrictions.

10. En informatique, une forge, ou plate-forme d'hébergement de projets logiciels, désigne un environnement Web constitué d'un ensemble d'outils du travail coopératif et du génie logiciel pour le développement collaboratif et distribué de logiciels.

11. CPAN est une archive fondamentale pour tout utilisateur du langage Perl, qui réunit énormément de documentation à son sujet et des modules (pm) librement utilisables.

12. CTAN est un réseau de serveurs FTP stockant de la façon la plus exhaustive possible des programmes, des macros, des polices et de la documentation pour TeX.

NOMBRE DE PROJETS, FICHIERS SOURCES, ET VERSIONS ARCHIVÉES DANS SOFWARE HERITAGE AU MOIS D'AVRIL 2019
https://www.softwareheritate.org/archive)

Or, courant 2015, deux plateformes de développement de logiciel libre très connues, Gitorious[14] et Google Code[15], 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. Ces évènements ont fait comprendre à la communauté des informaticiens et des développeurs que la mission de préservation ne pouvait pas être déléguée à des entités qui n'en font pas leur mission première. Il manquait une archive pour assumer cette mission, et garantir que si le code source d'un logiciel n'est plus disponible sur une plateforme de développement, ou si la plateforme elle-même disparait, on puisse quand même le retrouver dans l'archive.

UN GRAND INSTRUMENT DE RECHERCHE SUR LE CODE SOURCE DES LOGICIELS

La sûreté et la qualité logicielles sont cruciales dans de nombreux secteurs industriels comme l'aérospatiale, la médecine, l'énergie, le transport, la sécurité, qui sont au cœur de notre société. Actuellement, les logiciels industriels sont développés en assemblant de nombreux composants, dont il est impératif d'assurer la traçabilité et la qualité.

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

SOFTWARE HERITAGE : UNE ARCHIVE UNIVERSELLE DU CODE SOURCE

Software Heritage a été lancé pour répondre à ces trois défis majeurs, avec le soutien initial fort et total de l'Institut national de recherche en informatique et en automatique (Inria)[16]. Ses missions sont très précisément de *récolter, organiser, préserver et rendre facilement accessible* l'ensemble du code source disponible publiquement sur la planète, indépendamment d'où et comment il a été développé ou distribué. Le but 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'elle n'y parait : il suffit de passer en revue les défis à relever juste pour la phase de récolte des codes sources (et il y en a bien d'autres pour les autres phases). Archiver les codes exige de recenser les endroits où le code source se trouve, depuis les plateformes de développement très connues jusqu'à des archives posées sur des obscures pages web. Il n'y a pas de catalogue universel, il est à construire. Il est donc nécessaire d'appréhender les protocoles propres à chaque plateforme de développement, afin de lister leurs contenus, et maintenir à jour l'archive en suivant l'évolution dans les codes sources qui y sont hébergés. Il n'y a pas de standard, la conception d'un protocole minimal est prévue dans le futur par les porteurs du projet.

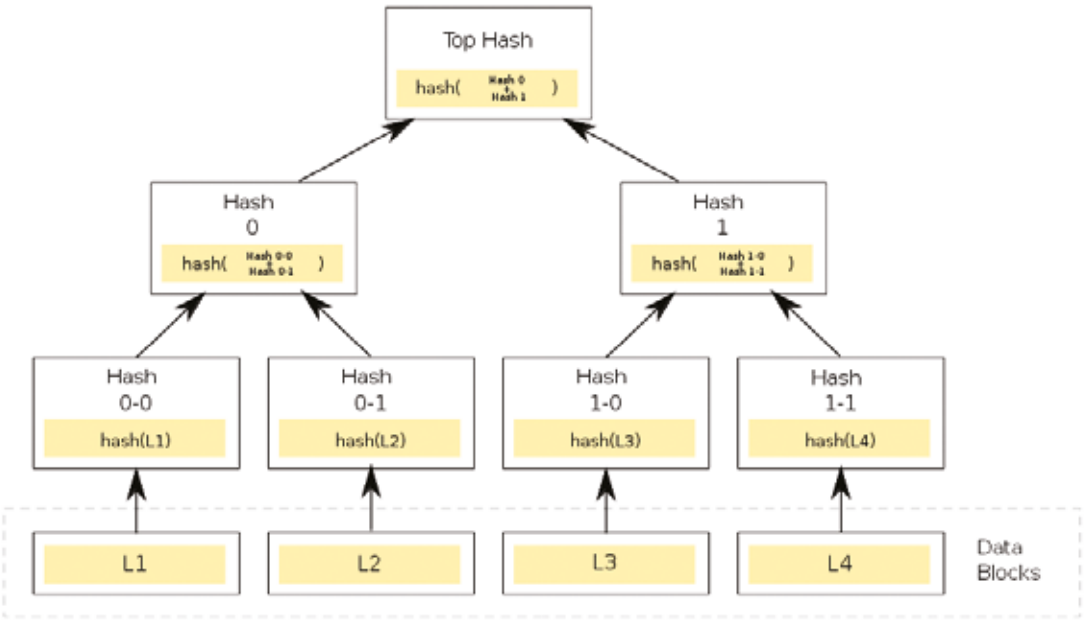
L'archivage repose sur la lecture de l'historique de développement qui se retrouve dans une grande variété de systèmes de contrôle de version : git, mercurial, subversion, darcs, bazaar, cvs, ce ne sont que quelques exemples des outils à étudier afin d'importer l'historique dans le modèle de données de Software Heritage. Cette mission d'universalité oblige à les regarder tous : il n'est pas question 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 : la construction d'un modèle de donnée capable de les capturer toutes est indispensable.

13. CRAN : ce site réunit les contributions relatives au langage R, c'est-à-dire au langage et à l'environnement destiné à faire de l'informatique statistique et des graphiques, développé par le GNU. GNU (GNU is not Unix) est le nom d'un projet de développement de logiciels libres mené en 1984 par la Free Software Foundation.

14. Gitorious était une forge logicielle libre et un service communautaire basé sur git. D'abord racheté en août 2013 par Powow AS (une société norvégienne et polonaise), il a ensuite été racheté par Gitlab le 3 mars 2015, lequel continue à proposer le code source de la forge sous licence libre sur son propre site, mais a fermé le site de Gitorious.

15. Lancé en 2006, Google Code était destiné aux développeurs qui y partageaient leurs projets « open source ». La plateforme a fermé le 26 janvier 2016. http://www.lemonde.fr/pixels/article/2015/03/13/google-code-ferme-ses-portes-au-profit-de-github_4592887_4408996.html

16. Créé en 1967, Inria est un établissement public à caractère scientifique et technologique spécialisé en mathématiques et informatique, placé sous la double tutelle du ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation et du ministère de l'Économie et des Finances.



EXEMPLE D'ARBRE DE MERKEL.
Azaghal - Own work, CC0 https://commons.wikimedia.org/w/index.php?curid=18157888

Enfin, d'un point de vue stratégique, il faut bien sûr une légitimité institutionnelle, mais aussi une véritable capacité d'ouverture pour fédérer un consensus large. L'accord-cadre signé entre Inria et UNESCO, le 3 avril 2017, est dans ce sens à la fois une reconnaissance de l'importance de la mission, et une grande opportunité pour établir des collaborations au niveau mondial pour l'accomplir.

QUELQUES PRINCIPES

La mission de Software Heritage est complexe, et comporte un engagement sur le long terme. Des principes fondateurs ont été élaborés, pour donner les plus grandes chances de succès avec des bases solides.

• Logiciel libre et transparence

Pour préserver de l'information à long terme, le fonctionnement de tous les outils mis en œuvre doit être connu : 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 des simples décisions économiques, voire juridiques. Il ne sera pas possible d'éviter toutes ces menaces, et il y aura des erreurs et des imprévus. Donc, au lieu de prétendre construire un système sans erreur, les porteurs de Software Heritage ont choisi de travailler à mettre en place un système qui tolère les erreurs. Pour cela, ils cherchent 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 leur infrastructure a précisément pour vocation de faciliter la création de nouveaux miroirs par des acteurs divers.

• Sans but lucratif et multi partenaires

L'expérience a montré qu'une seule entité, aussi puissante soit-elle, n'apporte pas assez de garanties sur le long terme. Pour assurer la mission de Software Heritage, il est donc indispensable de construire une fondation sans but lucratif ayant pour objectif précis la collecte, l'organisation, la préservation et la mise à 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.

• Un projet techniquement solide

Software Heritage est aujourd'hui une infrastructure qui grandit jour après jour.

Même si l'exhaustivité est encore loin d'être atteinte, le plus grand corpus de codes sources disponible sur la planète a déjà été réuni, avec plus de 90 mil-

lions d'origines archivées, pour plus de 6 milliards de fichiers sources uniques, chacun équipé d'un identifiant intrinsèque basé sur des signatures cryptographiques[17]. Il ne s'agit pas simplement d'une collection de répertoires mis les uns à côté des autres, mais d'un graphe de Merkel[18] qui partage au maximum les fichiers, les dossiers et les commits[19]. Cette structure de données est pensée pour pouvoir suivre la croissance, à une époque où le développement collaboratif, sur des plateformes comme GitHub, passe par la création de copies entières (forks) du projet originel.

Ce projet doit beaucoup à Inria, qui a su dès le départ comprendre l'importance de la mission de Software Heritage. Software Heritage vit grâce à l'engagement sans faille d'une équipe de personnes passionnées qui y travaillent chaque jour : en premier lieu Stefano Zacchiroli, qui pilote les aspects techniques du projet, et toute l'équipe qui inclut aujourd'hui Nicolas Dandrimont et Antoine Dumont. Ils ont bénéficié de l'expérience inestimable de Jean-François Abramatic, dont l'expérience de direction du W3C au moment de sa création est précieuse dans cette phase de jeunesse du projet. Le comité de pilotage comporte également trois autres membres : Serge Abiteboul, Gerard Berry et Julia Lawall.

L'archive de Software Heritage constitue déjà la collection de codes sources la plus importante de la planète, mais le chemin à parcourir est encore long : il est nécessaire de continuer à réunir les compétences scientifiques et techniques, ainsi que les ressources financières et humaines, afin de pouvoir construire la mémoire d'une partie importante de la technologie et de la science qui est au cœur de la transition numérique, à un moment où l'on peut encore espérer avoir accès à tout ce qui a été mis en œuvre dès le début de l'histoire, encore courte, de l'Informatique.

POUR EN SAVOIR PLUS

On peut savoir plus sur le projet en visitant www.softwareheritage.org, annex.softwareheritage.org et wiki.softwareheritage.org.

Il est possible d'explorer aisément les codes sources contenus dans Software Heritage sur archive.softwareheritage.org.

LICENCE

17. R. Di Cosmo, M. Gruenpeter, S. Zacchiroli : « Identifiers for Digital Objects: the Case of Software Source Code Preservation », iPres2018, https://hal.archives-ouvertes.fr/hal-01865790v3

18. Le graphe de Merkel est une généralisation des arbres de Merkle (ou arbres de hachage), une structure de données arborescente contenant dans chaque nœud un résumé cryptographique du sous-arbre correspondant.

19. En informatique, « commit » désigne l'enregistrement effectif d'une transaction ; dans les systèmes de bases de données et de révision de fichier, il est synonyme d'archivage ou de validation.