SOFTWARE HERITAGE, UNE ARCHIVE POUR COLLECTER ET PRÉSERVER LE CODE SOURCE

Entretien avec Roberto Di Cosmo

Roberto Di Cosmo

Directeur de Software Heritage
Professeur en science informatique à l'université Paris-Diderot

Les missions de l'archive Software Heritage s'articulent autour de la collecte, la préservation et le partage des codes sources des logiciels publiquement disponibles. Or, l'existence de cette archive d'un genre nouveau a de très nombreuses répercussions dans les domaines d'activité des professionnels de l'information de tous horizons : évolution des pratiques de signalement ; enjeux de citation et d'édition scientifique ainsi que de reproductibilité de la recherche ; médiation à la culture informatique, formation et encapacitation des usagers ; évolution des compétences des professionnels de l'information, etc.

Mais avant d'aller plus loin, revenons très brièvement sur quelques notions clés : le logiciel permet à un utilisateur d'exécuter un ensemble de tâches sur son ordinateur. Un logiciel fonctionne notamment grâce à du code exécutable, compréhensible uniquement par des machines. Or, l'« âme » d'un logiciel réside bel et bien dans son code source, c'est-à-dire dans les instructions telles qu'elles sont rédigées pour être lisibles par l'humain. Le code exécutable traduit pour la machine les indications du code source. Plus précisément, l'ordinateur exécute du code binaire selon les indications du code source. Fin de la parenthèse introductive.

Si Software Heritage pèse aujourd'hui plusieurs centaines de téraoctets, cette infrastructure est née il y a seulement quelques années. Fin 2014, Software Heritage est à l'état de projet. Le défi s'avère protéiforme : à l'échelle des infrastructures, les codes sources sont dispersés dans des silos aux gouvernances très hétérogènes. Le périmètre de travail est inévitablement mondial. À l'échelle des individus, les pratiques de développement accentuent le phénomène de dispersion des codes rendant difficile le traçage des différentes versions d'un logiciel (Di Cosmo, 2017). En 2015, sous l'impulsion de Roberto Di Cosmo et de Stefano Zacchiroli, l'Inria devient l'établissement porteur du projet, ouvert au public l'année suivante. 2017 constitue un tournant, avec la signature d'un accord-cadre entre l'Inria et l'Unesco. Enfin, depuis 2018, Software Heritage est interfacée avec HAL.

Le mouvement de patrimonialisation du code source franchit une étape majeure avec la création de Software Heritage : il est reconnu comme un objet patrimonial à part entière. Ainsi, le discours prononcé par Irina Bokova, la directrice générale de l'Unesco, souligne les enjeux citoyens, informationnels et culturels inhérents au code

source et aux logiciels : « *C'est aussi un enjeu citoyen dans les nouvelles sociétés du savoir, car dans des sociétés où l'accès à l'information, la liberté d'expression, la communication, dépendent de plus en plus largement des outils logiciels, la maîtrise de ce langage devient une compétence citoyenne de base* » (Unesco, 2017)[1]. Le code source constitue bien plus qu'un artefact technique.

Roberto Di Cosmo, directeur de Software Heritage, répond aux questions du *Bulletin des bibliothèques de France*. Après avoir enseigné à l'École normale supérieure de Paris, Roberto Di Cosmo devient professeur en science informatique à l'université Paris-Diderot ; il est détaché à l'Inria pour y conduire le projet Software Heritage. Défenseur des logiciels libres, il dirige notamment l'Initiative de recherche et innovation sur le logiciel libre (IRILL), une structure de recherche consacrée à la qualité des logiciels libres et *open source*.

*

BBF : Alors que nous vivons dans un monde de données, comment expliquer la tardive prise de conscience du caractère patrimonial du code source ?

Roberto Di Cosmo : Si on regarde bien autour de nous, on trouve des logiciels partout, et pourtant on ne s'en rend pas forcément compte : on est plutôt habitués à cliquer sur des icônes ou tapoter sur des écrans de téléphones mobiles, et on ne réfléchit pas trop au fait que, derrière tout ça, il y a bien des logiciels pour exécuter toutes ces tâches qui nous simplifient la vie.

Or, si on a du mal à percevoir le logiciel en tant que tel, il est encore plus difficile, quand on n'est pas informaticien, de prendre conscience de l'existence du fait que ces logiciels ne tombent pas du ciel : ils sont écrits par des êtres humains en utilisant des langages de programmation et les résultats de cette écriture sont bien le code source.

L'absence, jusque très récemment, d'une formation générale aux principes de bases de la programmation peut expliquer cette prise de conscience tardive : il est difficile d'apprécier quelque chose qu'on n'a jamais vu !

Quelle est la politique de conservation de Software Heritage ?

Dans Software Heritage, nous conservons tout ce qui concerne les développements logiciels : il s'agit bien sûr des lignes de code mais aussi de tout l'historique des développements logiciels, tel qu'il est par exemple disponible dans les systèmes modernes de contrôle de version[2]. On peut ainsi accéder à chaque modification que les développeurs ont considérée comme utile de consigner dans l'historique. Cela est important pour comprendre l'évolution d'un logiciel, et les modifications qui y ont été apportées.

1 Ce tweet du 20 avril 2021, de Monique Calisti, résume aussi les enjeux inhérents aux logiciels et aux codes sources : « *More than ever relevant to ensure a* #fair *and* #inclusive *society. Software development is not just about* #DigitalTransformation *it is about ensuring* #fairness #transparency *and ultimately* #democracy @NGI4eu @DigitalEU @francesca_bria @marabales @Martel_Innovate @WeNetProject https://twitter.com/SWHeritage/status/1380143867102695429 ».

2 [NDLR] L'une des solutions les plus connues est Git.

Quand on raconte à des archivistes ou à des bibliothécaires ce que nous faisons chez Software Heritage, ce qui les surprend le plus au premier abord est bien le fait que nous conservons absolument tout ce que nous trouvons : nous ne nous posons pas la question traditionnelle « *est-ce que cet objet particulier mérite bien d'être conservé ?* ».

Il y a deux raisons fondamentales pour cela : l'une, d'ordre technique, et l'autre, d'ordre philosophique. Tout d'abord, grâce aux techniques de déduplication et de compression des codes sources que nous employons, nous ne sommes pas obligés de faire des choix : il est possible de tout conserver. Mais cela ne suffit pas : même si *on peut* tout conserver, *est-ce qu'on doit* tout conserver ? Dans le cas du logiciel, nous considérons que la réponse est *oui*. Avec l'essor du logiciel libre, on développe les logiciels de façon de plus en plus partagée et ouverte : on peut observer le code source d'un logiciel qui est juste à son enfance, et on ne peut pas savoir si, avec le temps, cela donnera un logiciel de grande importance, dont on a évidemment envie de garder toute l'histoire, ou si cela ne donnera rien du tout.

Nous préférons tout garder, la sélection se fera *a posteriori* : ce qui est important sera référencé dans des articles, des études, des documentations, des portails web, alors que ce qui l'est moins restera dans l'ombre, mais toujours accessible dans l'archive.

Peut-on parler d'obsolescence du code source ou s'agit-il au contraire d'une « grammaire » qui résiste au temps ?

Tout d'abord, il faut savoir qu'il existe plus de 8 000 langages de programmation[3] : eh oui, dans la bien courte histoire de l'informatique, on a réussi à créer plus de langages de programmation que de langages humains (on en recense seulement un peu plus de 7 000, et c'est en comptant les langues mortes). Cela donne une idée de la vitesse à laquelle évolue le domaine de l'informatique.

Pourtant, il y a des logiciels qui ont une très longue histoire, et continuent d'être développés en utilisant le même langage de programmation : le noyau Linux, qui fêtera 30 ans en 2021, est écrit en C, tout comme bien d'autres briques de base de l'Internet moderne. On peut donc dire que si, d'une part, on voit une évolution très rapide dans les langages de programmation et dans les logiciels, de l'autre, pour un logiciel important, on voit bien que les éléments de base perdurent dans le temps.

Techniquement, ce patrimoine informatique est désormais rendu accessible à tous. Mais comment démocratiser l'accès à ce type de patrimoine, au-delà des seuls experts, comment sensibiliser le grand public aux enjeux de culture informatique alors que la pandémie a mis en lumière l'illectronisme qui frappe une partie de la population ?

Il y a plusieurs niveaux de réponse à cette question très importante. Une étape essentielle est de généraliser la formation aux principes de bases de l'informatique et de la programmation : c'est essentiel pour former des citoyens capables de comprendre

3 Voir https://hopl.info/

les enjeux de la transition numérique que nous vivons, tout autant que l'étude des autres disciplines – comme l'histoire, la littérature, la philosophie, les mathématiques, la biologie ou la physique – est essentielle pour comprendre la dynamique de notre environnement naturel et social.

Cela commence à venir, avec l'introduction de cours d'informatique au lycée, et la création récente d'un CAPES « Numérique et sciences informatiques ».

Une autre étape essentielle est de construire des présentations et des expositions, physiques ou virtuelles, qui permettent à tout public de découvrir la grande aventure humaine qui se cache derrière les logiciels qu'on archive aujourd'hui : il ne s'agit pas seulement de codes arides, mais de fruits ingénieux du travail et de la passion d'êtres humains dont l'histoire doit être racontée. Il y a plusieurs musées de l'informatique dans le monde qui font très bien ce travail en exposant des machines. Le temps est venu de faire de même en exposant les codes sources, et un travail dans ce sens est en cours avec l'Unesco.

Quand on parle de patrimoine, se pose la question des personnes qui en sont à l'origine. Comment faire évoluer la place des femmes dans les communautés de développeurs ?

Les femmes ont eu un rôle important dans l'évolution de l'informatique, plus qu'on ne le croit[4]. Mais le monde du développement logiciel a ensuite progressivement vu se réduire à peau de chagrin la participation des femmes[5]. Aujourd'hui, des programmes existent qui essayent d'apporter plus de diversité, en incluant spécifiquement les femmes, comme Outreachy[6].

Et il est important de mettre en avant, dès le plus jeune âge, des *role models* qui montrent comment les femmes peuvent avoir des rôles de tout premier plan : on peut mentionner Ada Lovelace, qui écrivait les premiers programmes pour le Analytical Engine que Charles Babbage ne put finir de construire ; Grace Hopper, qui créa le langage Cobol ; Margareth Hamilton, qui dirigea l'équipe chargée du logiciel embarqué dans l'Apollo 11, mais il y en a beaucoup d'autres qu'il faudrait mettre en valeur. C'est pour cela aussi que la mise en place d'expositions relatant les histoires des personnes qui ont développé ces logiciels est importante.

Si on ne sait pas précisément ce que l'on cherche, comment procéder pour interroger une cathédrale comme Software Heritage et ses millions de codes ?

L'ambition de Software Heritage est de fournir à terme un moteur de recherche avancé, et spécifiquement conçu pour les codes sources des logiciels, mais nous n'en

4 Voir https://lien.bechamail.fr/JbeeVMJp et https://www.computerscience.org/resources/women-in-computer-science/

5 Voir Stefano ZACCHIROLI, *Gender Differences in Public Code Contributions : A 50-Year Perspective*. IEEE Softw. 38 (2): 45-50 (2021). En ligne : https://doi.org/10.1109/MS.2020.3038765

6 https://www.outreachy.org/

sommes pas encore là. Ce qui est possible aujourd'hui est de rechercher des logiciels à partir de (fragments de) l'URL du projet d'origine.

L'identifiant propre à Software Heritage[7] permet de renvoyer très précisément à l'objet désigné par l'auteur, ce qui est très utile quand un chercheur veut, par exemple, citer un code. Malgré cette avancée technique majeure, on peut imaginer que les pratiques des éditeurs demeurent très variables. Pourriez-vous nous donner une idée du spectre des pratiques des éditeurs à l'heure actuelle ?

Construire un solide réseau de connaissances qui dure au fil du temps est d'une importance capitale, et les liens entre les différents objets en sont un élément clé. Pour cette raison, des références, des citations et divers systèmes d'identificateurs ont été utilisés pendant des siècles, bien avant l'ère informatique. Ces systèmes d'identifiants se divisent en deux grandes catégories, qu'on connaissait bien avant l'essor des technologies numériques.

Nous avons d'une part les identifiants *extrinsèques* qui utilisent un *registre* pour conserver la correspondance entre l'identifiant et l'objet : les numéros de passeport, l'ISBN, l'ISSN en sont des exemples bien connus, et on sait qu'il est indispensable, pour que cela fonctionne, qu'une autorité se charge de maintenir le registre, en évitant les doublons et les contrefaçons. À l'ère d'Internet, ces schémas se sont retrouvés transposés dans le monde numérique, avec des systèmes comme le DOI, Handle, Ark et bien d'autres. Dans le monde de la publication académique, on retrouve un usage massif du système DOI, alors que dans le domaine des archives et des bibliothèques on voit un large usage du système Ark, et il y en a beaucoup d'autres.

On retrouve d'autre part les identifiants *intrinsèques* qui sont intimement liés à l'objet désigné : dans ces systèmes, il n'y a pas besoin d'un registre, ni d'une autorité qui le maintienne, il suffit d'un accord sur la méthode pour les calculer. Deux exemples bien connus de tous nous viennent de la chimie et de la musique. Nous avons appris au lycée que nous n'avons pas besoin d'un registre qui attribue des identifiants différents à tous les composés chimiques possibles, il suffit d'apprendre une fois pour toutes la nomenclature standard, ce qui garantit qu'un nom chimique parlé ou écrit ne laisse aucune ambiguïté sur le composé chimique auquel il fait référence. Par exemple, la formule du sel de table est écrite NaCl et se lit « chlorure de sodium ».

Nous avons aussi appris à l'école que pour désigner sans ambiguïté toutes les notes de musique possibles, il suffit d'apprendre les notes de base et les octaves, ou, mieux encore, la notation scientifique de la hauteur.

Avec l'essor d'Internet et de la cryptographie, on a rapidement développé des identifiants intrinsèques numériques, qui prennent la forme de signatures cryptographiques et permettent de prendre en charge des opérations décentralisées, d'assurer

7 Voir SOFTWARE HERITAGE. *Intrinsic and Extrinsic identifiers*. 2020. En ligne : https://www.softwareheritage.org/2020/07/09/intrinsic-vs-extrinsic-identifiers/ [consulté le 3 mai 2021].

une vérification indépendante et de minimiser le recours à des autorités auxquelles on doit faire confiance.

Ces identifiants intrinsèques numériques sont utilisés massivement dans les blockchains, dans les systèmes pair à pair, et dans les systèmes de contrôle de version modernes utilisés par les développeurs de logiciels. Il est naturel que Software Heritage adopte aussi cette approche, en fournissant les SWHID (Software Heritage Identifiers) pour les plus de 20 milliards d'artefacts logiciels qu'il archive.

Dans le monde de la publication scientifique, les identifiants intrinsèques sont encore relativement peu connus, mais pour ce qui concerne les codes sources des logiciels associés aux publications ou archivés dans des portails en accès ouvert, les pratiques sont en train d'évoluer. Des revues comme *IPOL* ou *eLife* et le portail HAL les ont adoptées et on pense que les bénéfices liés à leur usage mèneront à une adoption large et rapide.

Quels sont les principes de fonctionnement du moissonneur qui identifie les codes sources en ligne ?

Software Heritage collecte les codes sources publiquement disponibles à travers trois mécanismes complémentaires. D'abord, il dispose d'un moissonneur qui visite régulièrement une liste grandissante de plateformes collaboratives de développement logiciel : cela va des très grandes plateformes comme Bitbucket, GitHub, GitLab, jusqu'aux petites forges logicielles de laboratoire.

Ensuite, il offre une fonctionnalité appelée « Save code now », qui permet de pointer les moissonneurs vers un dépôt de code quelconque disponible en ligne. Et enfin, il dispose d'une interface de dépôt qui permet à des archives institutionnelles de verser dans l'archive du code source soumis par des chercheurs. C'est cette dernière fonctionnalité qui est utilisée pour interconnecter HAL à Software Heritage, et vous pouvez en apprendre plus en lisant l'interview de Morane Gruenpeter, Alain Monteil, Estelle Nivault et Jozefina Sadowska.

Quels sont les enjeux de gouvernance inhérents à la conservation sur le long terme d'un patrimoine pouvant impliquer aussi des industries ?

Software Heritage est une infrastructure transversale qui s'adresse à une variété d'acteurs : cela va de la préservation du patrimoine culturel embarqué dans les codes sources, à la construction du pilier logiciel de la Science ouverte, à la fourniture de service essentiels pour retrouver, tracer et vérifier des codes sources, pour l'industrie et les administrations. Et tout cela, à une échelle mondiale et sur le long terme.

C'est pour cela qu'on a fait le choix stratégique de viser la création d'une organisation indépendante, sans but lucratif, ouverte à toutes les parties prenantes, et d'envergure internationale. Les partenaires actuels, une vingtaine, préfigurent la diversité des acteurs qui seront à terme appelés à contribuer à la gouvernance de cette organisation, dans le but de préserver l'archive sur le long terme.

On y retrouve des industries, des universités, des administrations publiques, et des organisations publiques et privées.

Vous décrivez régulièrement le code source comme une forme de « littérature technique du XXIe siècle ». Dans un cadre d'écriture où on ne s'affranchit pas d'un objectif opérationnel, qu'est-ce qu'un code élégant ?

Même si un objectif opérationnel existe (et on pourrait dire c'est le cas aussi de certaines productions littéraires), le moyen de l'atteindre, et la forme utilisée, peuvent varier grandement.

Comme dans la littérature, on peut trouver aussi dans le code des productions qui sont « mal écrites » et d'autres qui sont « élégantes et sophistiquées ». Il s'agit du choix des structures de données, de l'agencement des instructions et de l'organisation architecturale du logiciel : on a besoin de connaître le langage de programmation, et d'avoir une certaine culture informatique, pour en apprécier les subtilités et prendre plaisir en les découvrant.

Finalement, est-ce tellement différent de ce qui nous arrive quand on lit un texte complexe ?

On associe le code informatique au monde des mathématiques, mais selon vous, des compétences plus littéraires peuvent-elles être utiles ?

La conception et le développement de logiciels font appel en priorité à des compétences de logique et algorithmique qui sont en effet très proches de celles qu'on retrouve dans le monde des mathématiques, mais la capacité d'analyse et de synthèse, qu'on retrouve bien dans les domaines littéraires, est très importante aussi. Et il est essentiel de pouvoir expliquer et documenter clairement les logiciels[8], au point que le rôle de *technical writer* est très prisé.

Une question de culture générale : pourquoi parle-t-on de « numérique » et aussi peu d'« informatique » ?

Le terme « informatique » a été introduit en France par Philippe Dreyfus dans les années 1960, comme contraction de « information » et « automatique », quelques années après que l'ingénieur allemand Karl Steinbuch crée le terme « informatik ». Il désigne l'ensemble des aspects scientifiques et techniques liés au traitement de l'information. L'informatique a eu un impact énorme dans tous les domaines de l'activité

8 [NDLR] Alors que cette étape est fondamentale dans la diffusion d'un logiciel, la rédaction de la documentation d'un logiciel soulève rarement des vagues d'enthousiasme parmi les développeurs. Pour résoudre ce problème, Adam Hyde et Tomas Krag se sont intéressés à une méthode d'écriture collaborative intensive : le *book sprint*. En quelques jours, des équipes de rédacteurs conçoivent ainsi la documentation technique. Le travail peut être effectué en face à face ou à distance, l'enjeu restant d'accomplir cette tâche sur un temps resserré.

humaine, et cela en seulement quelques décennies, au point qu'on peut bien parler d'une véritable révolution.

Le terme « numérique » est plus ambigu : il s'agit à l'origine d'un adjectif décrivant quelque chose qui traite des « nombres » (comme dans « calcul numérique », « le rapport numérique », « il est en infériorité numérique »). Or, en informatique, on a l'habitude de représenter l'information sous forme discrète, par des séquences de 0 et de 1 qu'on peut lire comme des nombres binaires. Une étape importante est donc de transformer l'information qu'on veut traiter dans une représentation numérique : c'est ce que font les « scanners », les senseurs et les capteurs photos dans nos téléphones. On parlait donc de « numérisation de l'information ». Plus récemment, l'adjectif s'est converti en substantif, et « le numérique » paraît vouloir couvrir tous les domaines dans lesquels l'usage de l'informatique a un impact, et pour lesquels on retrouve systématiquement l'usage de l'adjectif (la santé numérique, les médias numériques etc.).

Personnellement, je préfère sincèrement le terme « informatique », qui renvoie au traitement de l'information, indépendamment du fait qu'il s'agisse d'information représentée sous forme numérique ou pas, et du fait que l'exécution d'une série d'instructions se fasse avec ou sans ordinateur. Un algorithme de tri fonctionne très bien même quand on l'exécute à la main sur des objets qui ne sont pas numérisés (cela nous arrive encore de devoir trier des copies papier en ordre alphabétique, par exemple, et là, il s'agit bien d'informatique, mais sans aucune trace de numérique). •

Sources

- DI COSMO, Roberto. « Software Heritage : pourquoi et comment construire l'archive universelle du code source », *Bulletin de la société informatique de France*. Avril 2017, nº 10, p. 67-72. En ligne : https://lien.bechamail.fr/LHtDyFev [consulté le 3 mai 2021].
- UNESCO. *Discours de la Directrice générale de l'UNESCO, Irina Bokova, à l'occasion de la signature de l'accord entre l'UNESCO et INRIA portant sur la préservation et le partage du patrimoine logiciel*. 2017. En ligne : https://lien.bechamail.fr/qoknDth2 [consulté le 3 mai 2021].

Aller (encore) plus loin : sélection du BBF

- CLASS'CODE. « Penser le numérique », Blog *Class´Code et ses pixees*. 2017. En ligne : https://pixees.fr/penser-le-numerique/ [consulté le 3 mai 2021].
- DI COSMO, Roberto. « Software Granularity. Intrinsic and extrinsic identifiers to the rescue ». En ligne : https://annex.softwareheritage.org/public/talks/2021/2021-04-21-RDA-Data-Granularity.pdf [consulté le 3 mai 2021].
- DI COSMO, Roberto. « Report from the Task Force on Scholarly Infrastructures for Research Software (SIRS) ». En ligne : https://lien.bechamail.fr/Msjkah6R [consulté le 3 mai 2021].

- EOSC EXECUTIVE BOARD ET EOSC SECRETARIAT. *Scholarly infrastructures for research software. Report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS*. European Commission. Directorate General for Research and Innovation. 2020. En ligne : https://data.europa.eu/doi/10.2777/28598 [consulté le 3 mai 2021].
- MAGRON, Agnès. *Préservation des logiciels : une collaboration Software Heritage et CCSD*. 2018. En ligne : https://lien.bechamail.fr/3HJL9szT [consulté le 6 mai 2021].
- KLAUS, Rechert, Jurek OBERHAUSER, et Rafael GIESCHKE. « Ensuring usability of a scientific code base ». En ligne : https://lien.bechamail.fr/PRzpa-uZ [consulté le 3 mai 2021].
- PAIK, Ray, et Sofia WALLIN. « Making Documentation a First-class Citizen in Open Source Projects ». En ligne : https://lien.bechamail.fr/DnZbETrs/ [consulté le 1er février 2021].
- SOFTWARE HERITAGE. *Roadmap 2021 – Software Heritage - Development Documentation documentation*. 2021. En ligne : https://docs.softwareheritage.org/devel/roadmap/roadmap-2021.html [consulté le 3 mai 2021].

Résumé

Directeur de Software Heritage, Roberto Di Cosmo est chercheur en informatique. Il dirige l'« Initiative de recherche et innovation sur le logiciel libre » (Irill), une structure de recherche. R. Di Cosmo explicite la conception du patrimoine logiciel à l'origine de l'archive Software Heritage, initiative soutenue par l'Unesco et l'Inria.