HAL
open science

La science du logiciel libre par Roberto Di Cosmo. Entre auto-organisation et méritocratie, entretien avec Jean-Bernard Stefani, propos recueillis par Dominique Chouchan.

Roberto Di Cosmo, Jean-Bernard Stefani

To cite this version:

Roberto Di Cosmo, Jean-Bernard Stefani. La science du logiciel libre par Roberto Di Cosmo. Entre auto-organisation et méritocratie, entretien avec Jean-Bernard Stefani, propos recueillis par Dominique Chouchan.. Les Cahiers de l'INRIA - La Recherche, 2009, Les promesses des énergies renouvelables, 436 décembre 2009. inria-00511691

HAL Id: inria-00511691

https://inria.hal.science/inria-00511691v1

Submitted on 26 Aug 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

les cahiers de l'INRIA

Rendez-vous avec les sciences et technologies de l'information et de la communication

GÉNIE LOGICIEL

La science du logiciel libre

La qualification de « libres » pour des logiciels ne rime pas avec simplicité, loin de là : ces derniers posent des problèmes théoriques spécifiques liés à leur mode d'élaboration.

Les collaborations avec des partenaires dispersés géographiquement ne datent pas d'aujourd'hui, notamment en science. Mais jusqu'à une époque récente, mieux valait se rencontrer physiquement pour discuter sérieusement et échanger les informations nécessaires. Avec l'évolution des technologies fondées sur l'Internet, la situation a radicalement changé. Le phénomène des logiciels libres s'inscrit dans ce contexte. On voit se développer des projets collaboratifs tels que Wikipedia ou, dans un autre genre, Linux, tous ayant en commun d'être le fruit d'une coopération entre un grand nombre de contributeurs*. Au-delà de l'aspect socio-économique, déjà largement étudié, comment de tels systèmes peuvent-ils fonctionner ? Sur quels principes scientifiques se fondent-ils et quelles sont les recherches à mener pour en améliorer la stabilité et la fiabilité ?

Avec Wikipedia, le défi était de construire une encyclopédie alimentée et entretenue par une communauté dont les membres n'appartiennent pas au monde de l'édition. Cette encyclopédie libre a connu une croissance exponentielle, depuis sa création en 2001, sans aucune coordination centralisée pour en assurer la cohérence. Le fait que certains articles manquent, ou soient obsolètes, voire faux, ne l'empêche pas de remplir sa fonction.

C'est en quelque sorte un exemple extrême de travail massivement collaboratif. Quant à Linux*, comme de nombreux logiciels libres, il symbolise un défi de nature très différente : celui d'élaborer un noyau de système d'exploitation robuste dont la maintenance soit assurée par une communauté de développeurs qui, eux non plus, n'appartiennent pas au milieu traditionnel de l'édition de logiciels.

Ces deux exemples ne répon-

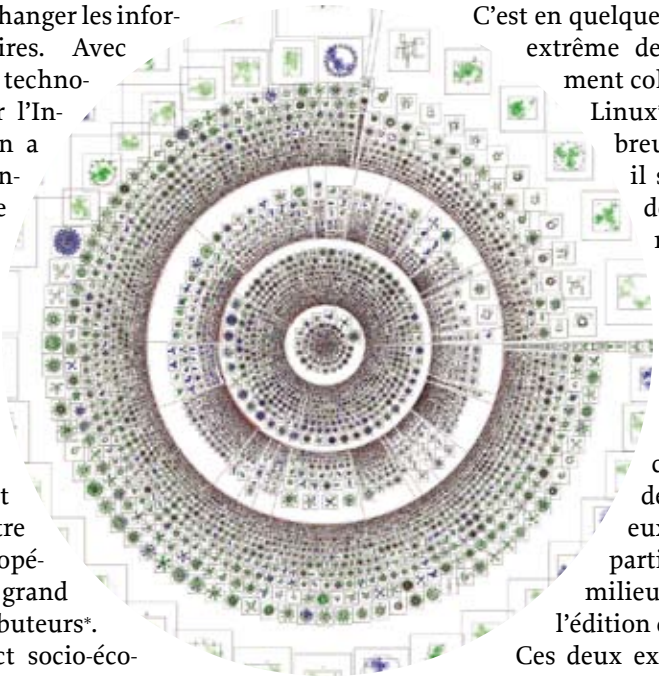
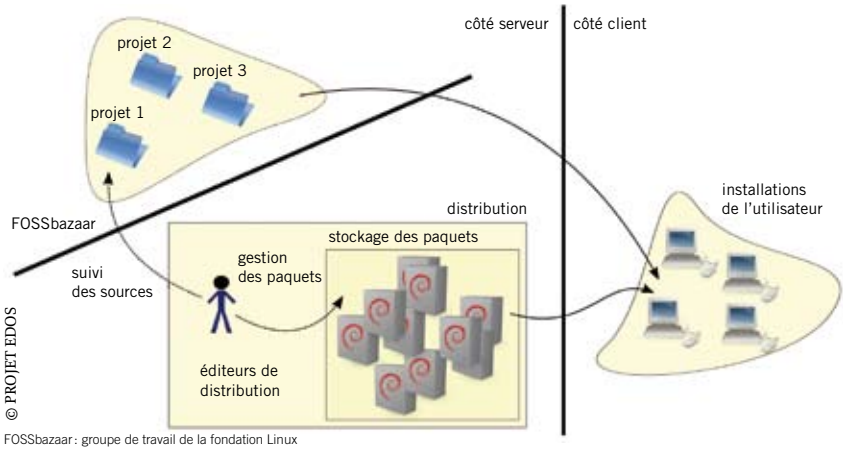


Fig. 1 : Ce graphique donne une représentation d'une partie du noyau Linux (version 2.6.17.6, 2006) qui contient 5 millions de lignes de code (produit avec le logiciel libre LGP).

dent donc pas au mêmes critères d'analyse : contrairement à Wikipedia, le noyau Linux, développé depuis le début des années 1990, est un objet informatique éminemment technique, dans lequel chaque erreur, ou chaque fonction manquante ou obsolète, peut entraîner des dysfonctionnements majeurs qui rendent l'ensemble inutilisable. D'où le caractère encore plus surprenant du « phénomène Linux » qui a abouti à des versions de plus en plus plus complètes, fonctionnelles et stables. Le succès des logiciels libres de qualité repose en fait sur des approches méthodologiques très structurées. Il va de soi que si chaque développeur écrivait ou modifiait quelques lignes de code dans son coin sans aucune coordination avec les autres, comme on serait tenté de le croire si on a en tête le modèle Wikipedia, le résultat final serait désastreux.



FOSSbazaar : groupe de travail de la fondation Linux

Fig. 2 : Les distributions de logiciels libres jouent un rôle d'intermédiaire entre les utilisateurs et les dizaines de milliers de logiciels libres disponibles. Chaque utilisateur peut choisir les paquets logiciels prêts à l'emploi répondant à ses besoins.

L'organisation modulaire du code source, la sélection sévère des contributions et l'existence d'un noyau central de développeurs s'assurant de la cohérence de l'ensemble ont en effet joué un rôle essentiel dans la réussite de Linux. Reste que la taille atteinte aujourd'hui par ce logiciel est telle que l'on est aux limites de ce qui peut être maîtrisé en s'appuyant seulement sur les capacités organisationnelles des communautés de développeurs (fig. 1). Par la force des choses, les logiciels libres sont donc devenus un champ d'étude à part entière pour les informaticiens. Les questions scientifiques portent notamment sur les liens entre les composants du code et sur la manière d'adapter les outils classiques, conçus à une époque où ce dernier était indisponible et peu évolutif. Dans le monde du libre, le code est disponible, avec des contributions variées et des interactions fortes et rapides entre composants.

Typiquement, comment gérer rapidement la modification d'un appel de fonction résultant de l'ajout, du changement ou de la suppression d'un paramètre ? Un exemple de ce cas de figure : entre deux versions successives du noyau de Linux, une fonction servant à représenter un dispositif périphérique (clé USB ou autre) s'est vue caractérisée par deux paramètres au lieu d'un. Ce sont alors des milliers de modules de code source qui sont affectés, au travers de toutes les lignes utilisant la fonction en question : on parle d'« évolutions collatérales » du code. Le travail de correction est long, ingrat et source de maintes erreurs, même pour les programmeurs les plus méthodiques.

Dans ce contexte, le projet de recherche Coccinelle*, démarré en 2006, a fourni une contribution majeure en introduisant la notion de « patch sémantique »[1,2]. Un patch sémantique permet de créer une sorte de description concise et abstraite (de haut niveau) des modifications induites dans le noyau Linux par un changement dans les paramètres d'une fonction. Sur la base de cette description, un outil automatique, intégrant des algorithmes sophistiqués construits à partir de règles logiques, propage les changements dans les modules sources concernés. Il prend en compte tous les détails comme le changement de nom des variables, les appels imbriqués dans des expressions logiques conditionnelles (comme : « si..., alors... »), etc. Il permet ainsi aux développeurs de se concentrer sur des tâches plus nobles et facilite le développement collaboratif. Cet outil est en train d'être adopté par les développeurs de Linux qui ont déjà réalisé plus de 300 patchs sémantiques.

Une autre question, qui se pose à un niveau plus macroscopique, autrement dit à celui d'un ensemble de composants logiciels, porte sur les conditions d'utilisation d'un tel logiciel. Il est en effet théoriquement possible, pour un utilisateur averti, de télécharger le code source d'un logiciel libre, de le compiler,

* Sur les logiciels libres, voir aussi http : //interstices.info/libre-developpement

* Pour la clarté de l'exposé, j'ai pris le parti d'évoquer essentiellement le logiciel libre Linux. Il en existe bien d'autres, comme OpenOffice (bureautique), Mozilla Firefox (Internet), VLC media player (multimédia), etc.

* Coccinelle associe le Laboratoire d'informatique de Paris 6 (LIP6), l'Inria, le Centre d'innovation et de recherche en informatique sur le logiciel libre (Cirill) et le département d'informatique de l'université de Copenhague (DIKU) (voir http : // coccinelle.lip6.fr/)

* Un paquet logiciel est une collection de fichiers, d'instructions de configuration et de métadonnées permettant d'installer un logiciel.

* Le projet européen Edos, ou *Environment for the development and Distribution of Open Source Software*, associe des acteurs privés et publics (dont l'Inria), européens et israéliens (voir www.edos-project.org)

* Un solveur SAT est un logiciel capable de déterminer si une formule booléenne peut être satisfaite ou si elle est incohérente.

* Le projet européen Mancoosi associe des acteurs privés et publics (dont les universités Paris Diderot et de Nice-Sophia-Antipolis), européens, israéliens et argentins (voir www.mancoosi.org)

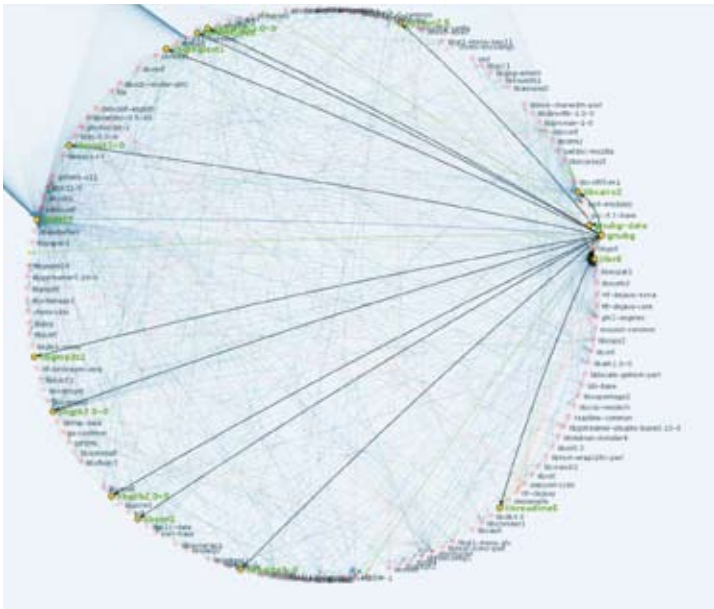


Fig. 3 : Les relations entre composants logiciels peuvent être très enchevêtrées. Par exemple, pour jouer au BackGammon en utilisant un logiciel appelé gnubg, les gestionnaires de paquets doivent démêler dépendances et conflits entre plus d'une centaine de paquets. Ces dépendances sont schématisées ici par un outil issu du projet Edos.

de l'installer et de le configurer sur sa machine, en résolvant « à la main » les problèmes éventuels rencontrés, quitte à réitérer l'opération chaque fois qu'une nouvelle version est disponible (fréquemment dans le monde du logiciel libre). Mais, en pratique, presque plus personne ne s'aventure dans ce processus. Il est trop long et trop fragile, même pour une configuration minimale.

D'où l'intérêt des distributions de logiciels libres comme Debian, Mandriva, RedHat, FreeBSD, NetBSD, etc., toutes construites sur des systèmes d'exploitation libres comme Linux ou BSD. Une distribution est un ensemble cohérent de composants logiciels qui a pour fonction de prendre en charge, à la place de l'utilisateur, la tâche consistant à suivre l'évolution de chaque logiciel libre, à le compiler et à le mettre à sa disposition sous la forme d'un « paquet logiciel* » (fig. 2). Celui-ci est conçu et testé pour fonctionner correctement avec les autres paquets gérés par la distribution (contenant parfois plus de vingt-mille composants logiciels). Chaque paquet contient des informations (métadonnées) qui indiquent les composants dont il a besoin pour fonctionner correctement et ceux avec lesquelles il ne peut cœxister (dépendances et conflits). Des outils spécialisés, dits gestionnaires de paquets, permettent ensuite à l'utilisateur de choisir les paquets qu'ils souhaite installer.

Mais là encore, les outils dont on dispose ont atteint leurs limites, vu la taille du problème (fig. 3). Les recherches menées dans le cadre du projet Edos* s'inscrivent dans ce contexte. Ce dernier a d'abord montré que le problème d'installation des composants appartient à une catégorie de problèmes (que l'on appelle NP-complets) parmi les plus complexes dans la hiérarchie de la complexité[3]. Il s'agit donc là d'un problème posé à la recherche théorique. Il a également permis de préciser comment des variantes de solveurs SAT* peuvent être utilisées pour détecter très rapidement les paquets qui ne peuvent en aucun cas être utilisés (paquets dits « cassés »). Les outils développés par Edos sont aujourd'hui couramment utilisés par des distributions Linux comme Debian ou Mandriva afin de ne pas propager chez les utilisateurs des composants défectueux.

Un autre projet de recherche, baptisé Mancoosi* et lancé en février 2008, vise à développer des algorithmes efficaces pour permettre aux utilisateurs d'exprimer leurs préférences lors de mises à jour et d'obtenir des installations personnalisées : les besoins diffèrent évidemment selon que l'on gère une grosse entreprise industrielle ou financière, ou que l'on utilise sa machine pour des jeux multimédias. L'idée à terme est de leur fournir des outils nettement plus flexibles et efficaces que ceux d'aujourd'hui[4].

Pour les trois projets mentionnés dans cet article (Coccinelle, Edos, Mancoosi), les recherches menées dans les laboratoires n'auraient pu l'être sans le logiciel libre : d'un côté il est essentiel pour les chercheurs d'avoir accès sans restriction à l'ensemble de l'information relative au développement du code et des composants, de l'autre les problèmes étudiés ne deviennent intéressants que lorsque la vitesse des changements, le nombre de composants et le nombre de développeurs sont très élevés, ce qui est la caractéristique principale du développement collaboratif propre au logiciel libre. Ce type de recherche est crucial à l'heure où l'usage des logiciels libres progresse rapidement.

Roberto Di Cosmo est professeur à l'université Paris-Diderot-Paris 7 et membre de l'unité mixte de recherche Preuves, programmes et systèmes (CNRS, université Paris Diderot). Il pilote la mise en place du Centre d'innovation et de recherche en informatique sur le logiciel libre (Cirill) dont la création a été annoncée début octobre 2009 par Michel Cosnard, directeur de l'Inria.

[1] J. Brunel et *al.*, *A Foundation for Flow-Based Program Matching Using Temporal Logic and Model Checking*, The 36th Annual ACM SIGPLAN - SIGACT Symposium on POPL, p. 114, Savannah, 2009.

[2] Y. Padioleau et *al.*, *Documenting and Automating Collateral Evolutions in Linux Device Drivers*, EuroSys 2008, p. 247, Glasgow, 2008.

[3] F. Mancinelli et *al.*, *Managing the Complexity of Large Free and Open Source Package-Based Software Distributions*, 21st IEEE/ACM International Conference on ASE 2006, p. 199, Tokyo, 2006.

[4] P. Abate et *al.*, *Strong Dependencies between Software Components*, 3rd International Symposium on ESEM, Orlando, 2009.

Entretien avec Jean-Bernard Stephani

Entre auto-organisation et méritocratie

Les problèmes posés par le développement collectif de logiciels complexes sont amplifiés avec les logiciels libres du fait de la participation de nombreux contributeurs.

Quelles sont les principales règles de construction d'un logiciel libre ?

Jean-Bernard Stephani : Elles sont en fait communes à tout logiciel complexe et s'articulent autour de trois grandes catégories de questions. Les premières portent sur le processus de développement : répartition des tâches entre développeurs, identification des étapes de conception... Le modèle d'organisation est d'autant plus complexe que le nombre de contributeurs est élevé et que ces derniers sont dispersés géographiquement, de cultures et de méthodes de travail différentes..., ce qui est le cas pour les grands logiciels libres. La seconde catégorie de questions concerne la conception. Une idée clé est de concevoir des logiciels les plus modulaires possibles, avec des modules aussi indépendants que possible les uns des autres. Une telle structure simplifie non seulement leur développement mais aussi leur maintenance, leur évolution, leur déploiement et leur administration. Dans le cas du libre, cette modularité devient cruciale pour l'intégration des diverses contributions. Enfin, des outils informatiques sont nécessaires pour aider à la mise en œuvre du processus et pour faciliter la conception : c'est la troisième catégorie de questions.

Mais comment s'organise-t-on si tout le monde est contributeur potentiel ?

J.-B. S. : Dans le domaine du libre, il n'existe pas d'autorité centralisée *a priori*. En revanche, il se dégage une autorité de fait : le contributeur initial, qui devient alors l'autorité de référence (cas de Linux), ou bien un groupe initial de développeurs d'accord sur un projet commun (cas du logiciel de serveur HTTP Apache). On a affaire à une forme d'« auto-organisation » fondée sur une méritocratie : au fil du temps, ceux qui se révèlent particulièrement utiles acquièrent une certaine autorité.

Vous développez vous-même des logiciels dits intergiciels*. Pourquoi le choix de logiciels libres ?

J.-B. S. : Nous menons notamment ce travail dans le cadre du consortium OW2*. Les intergiciels qui nous intéressent doivent faciliter la programmation et la mise en œuvre d'applications sur des ensembles de machines en réseau, par exemple pour les calculs distribués. Il est intéressant que ces logiciels soient libres : ils sont naturellement appelés à devenir des produits de base (*commodities* en anglais), la valeur ajoutée sur laquelle les entreprises feront du profit étant alors réservée à des fonctions de plus haut niveau. Au sein de OW2, l'Inria pilote par exemple le projet Fractal*. Celui-ci propose des spécifications pour le développement de logiciels modulaires (programmation par composants) ainsi que des outils de construction répondant à ces spécifications.

D.R.

Jean-Bernard Stephani, ingénieur général des Mines, est responsable scientifique de l'équipe Sardes de l'Inria Grenoble-Rhône-Alpes. Il a été l'un des cofondateurs et animateurs du consortium ObjectWeb associant des acteurs européens publics et privés, auquel a succédé le consortium OW2 à partir de 2007.

Finalement, le logiciel libre permet des avancées dans le génie logiciel ?

J.-B. S. : Le développement de logiciels libres complexes met en effet en évidence des questions difficiles de génie logiciel, mais à l'inverse, la disponibilité de nombreux logiciels libres permet aux chercheurs en génie logiciel de disposer d'une base d'expérimentation très intéressante. Plusieurs outils d'aide au développement originaux ont pu ainsi être testés et validés sur la base du code Linux.

Propos recueillis par Dominique Chouchan

* Un **intergiciel**, traduction du mot anglais *middleware*, est un logiciel qui sert en quelque sorte d'interface entre le *hardware* et le système d'exploitation, d'un côté, et les applications, de l'autre.

* Le **consortium international OW2** s'est fixé pour objectif de développer des intergiciels libres de qualité industrielle (www.ow2.org). Il associe des organismes publics (dont l'Inria) et des partenaires privés tels que Bull, France Telecom, Thales, Red Hat...

* Pour en savoir plus, voir : http : //fractal.ow2.org.

Réalisation rédactionnelle : Dominique Chouchan - Réalisation graphique : E-Graphics