LES LIVRETS BLEUS DU

LOGICIEL LIBRE

Qualité logicielle

ROBERTO DI COSMO

PHILIPPE VAILLERGUES

FABRICE BERNHARD

Contribution de
FRÉDÉRIC LEPIED

PATRICK MOREAU

Préface de
STÉFANE FERMIGIER

Systematic
Paris Region Systems & ICT Cluster

LOGICIEL LIBRE

Ont contribué à ce livret bleu :

Roberto Di Cosmo (IRILL, Université Paris 7 Denis Diderot, vice-président du GTLL),

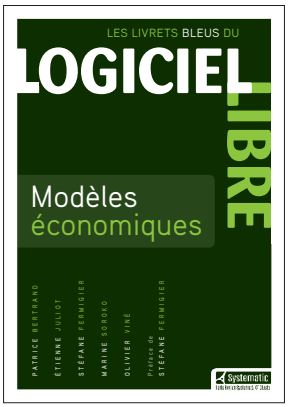
Philippe Vaillergues (Henix),

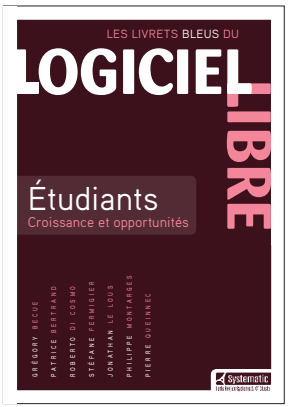
Fabrice Bernhard (Theodo),

Stéfane Fermigier (Abilian, président du GTLL).

Remerciements à Muriel Shan Sei Fan pour le travail d'édition, et à Didier Méresse (Nord Compo) pour la conception.

LES LIVRETS BLEUS DU GTLL : DES REPÈRES POUR COMPRENDRE





Sommaire

Préface

Chapitre 1
Qualité Logicielle, de quoi s'agit-il ? 5

Chapitre 2
L'informatique embarquée critique 8

Chapitre 3
L'informatique en milieu industriel non critique 10

Les enjeux croissants de la qualité logicielle 11

L'importance du processus de développement 13

Chapitre 4
Le logiciel libre et la qualité logicielle 15

Préface

Un logiciel, ça bogue ; un projet informatique, ça dérive. Ce sont des évidences présentes à l'esprit tant du grand public que des professionnels. La problématique de la qualité logicielle est perçue intuitivement – intuition alimentée par plusieurs décennies d'écrans bleus, de pertes de données, de failles de sécurité exploitées, de projets en retard ou abandonnés à grands fracas – et souvent sans grand recul sur ses véritables enjeux.

Dans ce contexte, le logiciel libre, dont les modes de fonctionnement et les modèles économiques restent encore mystérieux pour les non-spécialistes, est encore trop rarement associé à l'idée de qualité. Si les géants américains, qui exhibent des milliards en budgets de R&D, n'y arrivent pas, que peut-on attendre de ces communautés et écosystèmes aux moyens si restreints en comparaison ?

Pourtant, les analyses le montrent depuis 15 ans, dans la plupart des domaines où une offre professionnelle de logiciel libre s'est développée ou se développe activement (outils de développement, infrastructure, cloud computing, big data, after PC…), les logiciels libres rivalisent et souvent surpassent en qualité les logiciels propriétaires qui leurs sont comparables.

Il y a là un paradoxe qui mérite d'être expliqué. Ce que nous ferons tout d'abord, en appréhendant ces questions sous l'angle de l'ingénierie, et non pas seulement du folklore, puis en passant en revue les outils et les méthodes qui permettent de répondre aux défis – économiques et sociétaux – que pose la faillibilité du logiciel dans une société en pleine transition numérique.

Dans ce domaine, les approches dont le logiciel libre a été le pionnier – innovation ouverte, développement distribué, développement dirigé par les tests, architectures modulaires et extensibles, résilience intégrée sciemment dans les protocoles du Web et de l'Internet, etc. – constituent autant de pistes pour améliorer la qualité de la production de l'ensemble de l'industrie logicielle.

La qualité logicielle est, avec le Big Data et l'After PC, l'un des trois grands axes de R&D du groupe thématique Logiciel Libre. Nos membres investissent largement dans ce domaine pour améliorer la compétitivité de notre écosystème, au niveau régional mais aussi national et mondial. Cet ouvrage expose de façon synthétique ses enjeux et éclaire les réponses que nous y apportons.

Stéfane Fermigier,
président du groupe thématique Logiciel Libre

Avec le triple respect des coûts, des délais et des fonctionnalités, la qualité logicielle est le quatrième facteur caractérisant «l'art – ou la science – de la production du logiciel». Or, étant plus complexe à mesurer, la qualité logicielle en vient trop souvent à constituer un facteur d'ajustement.

Pendant longtemps, on a assisté à des approches différentes, presque étanches, de la qualité logicielle, étroitement liées aux caractéristiques et aux contraintes de l'environnement dans lequel on se trouvait.

Chapitre 1

Qualité Logicielle, de quoi s'agit-il ?

Le problème de garantir qu'un logiciel réalise bien les fonctions pour lesquelles il a été conçu est très ancien, et a intéressé des grands noms de la recherche en Informatique, comme C.A.R. Hoare et David Gries, qui défendaient déjà dans les années 1970 l'idée d'écrire du code exclusivement à partir de spécifications, et d'en prouver la *correction* avec l'aide d'instruments issus de la Logique Mathématique. Malgré des résultats théoriques limitatifs bien connus, cette idée fondatrice a donné lieu à un foisonnement d'activités de recherche qui visent à employer des méthodes formelles pour s'approcher de cet objectif, en allant de la vérification automatique à la preuve formelle, à l'interprétation abstraite, sans parler des diverses techniques utilisées pour la conception de logiciels, tels les microprocesseurs.

Il s'agit d'un domaine de l'Informatique qui montre clairement le temps long nécessaire à la recherche sur des sujets complexes : **il a fallu plusieurs dizaines d'années, et d'énormes progrès techniques et scientifiques, pour passer des premières approches proposées dans les années 1960 aux exploits d'aujourd'hui**

que sont la production d'une chaîne de compilation entièrement certifiée grâce à un assistant à la demonstration (projet CompCert, Inria), la vérification du code de contrôle-commande de l'Airbus A380 (projet Astrée, ENS Ulm, et Inria), la certification d'un noyau de système d'exploitation au niveau seL4 (Nicta).

Une large palette de techniques et d'outils existe enfin pour certifier du code, ou en prouver des propriétés (typage fort, vérification, interprétation abstraite, preuve formelle…).

> *Beware of bugs in the above code;*
> *I have only proved it correct, not tried it.*
> *Donald Knuth, 1977*

Mais la qualité logicielle ne se limite pas à la question importante de l'établissement de la correction du code par des méthodes formelles : la dimension et la complexité du code dévéloppé aujourd'hui, et la vitesse avec laquelle il change, rendent indispensable la mise en place de *tests* systématiques, qui doivent être proprement conçus, structurés et maintenus.

Et même lorsqu'on arrive à obtenir un code prouvé correct, avec une assurance totale sur la fonctionnalité qu'il réalise, on reste confronté à la problématique de la lisibilité, de la maintenabilité et de l'évolutivité de ce code, qui dépendent profondément de la façon donc le

code est *structuré et écrit*. Ces questions importantes font l'objet de recherches dans le domaine du génie logiciel, qui étudie *l'architecture* des logiciels, leur *complexité*, et le *processus* de développement et de maintenance, avec le besoin fort de fournir des *mesures* exploitables.

Dans ce contexte, **le logiciel libre est le moyen qui s'est généralisé parmi les chercheurs** pour partager leurs résultats et faciliter les contributions entre eux – quoique on peut observer que le développement même des outils qui réalisent ces méthodes ne suit que très exceptionnellement ces mêmes méthodes.

Plusieurs projets de recherche exploratoires financés dans ce domaine ont été labellisés par le pôle Systematic et nous considérons que cet axe de recherche à long terme est stratégique. Ainsi le groupe thématique Logiciel Libre a-t-il labellisé les projets de recherche (ANR) ASOPT, Codex, et le projet de R&D collaborative FUI 14 SafePython.

http://systematic-paris-region.org/fr/projets/asopt
http://systematic-paris-region.org/fr/projets/codex
http://systematic-paris-region.org/fr/projets/safepython

Chapitre 2

L'informatique embarquée critique

Malgré des exploits certains, l'utilisation de ces méthodes reste anecdotique dans le monde industriel, qui commence à n'y avoir timidement recours que dans quelques environnements hautement critiques et intrinsèquement coûteux qui justifient de recourir aux preuves formelles.

Dans ces domaines, des normes strictes obligent les industriels à un effort très conséquent de qualification de leur code, pour des exigences de fiabilité (transport, avionique) ou de sécurité (cartes à puces, télécommunications) ; l'activité de programmation y est soumise à des règles métier très contraignantes.

Le coût grandissant du développement des couches logicielles nécessaires à la construction des applications

critiques, et le besoin fréquent d'en assurer la maintenance sur des durées pouvant dépasser largement la décennie, ont conduit à l'utilisation de logiciels libres dans ces domaines, et le phénomène prend de plus en plus d'ampleur.

Le besoin de s'appuyer sur des standards pour la certification, et la possibilité d'avoir recours à des preuves formelles posent les bases pour des avancées significatives. Des projets dans cette direction ont déjà été financés après labellisation par Systematic (projets HILITE (FUI 9), projet Couverture (FUI 5), etc.), et cette voie de recherche reste très prometteuse.

http://systematic-paris-region.org/fr/projets/hilite
http://systematic-paris-region.org/fr/projets/couverture

Chapitre 3

L'informatique en milieu industriel non critique

Les entreprises savent désormais que **la qualité de leurs logiciels est une caractéristique de la valeur de leur Système d'Information, et de leur patrimoine applicatif.** Dans ce contexte non critique qui représente l'essentiel du code utilisé aujourd'hui, on trouve des niveaux d'exigence et de criticité divers : systèmes d'information d'entreprise et d'administration, informatique transactionnelle sensible (par exemple dans le secteur bancaire), informatique embarquée non critique.

Tous ces cas sont caractérisés par la sédimentation de grandes quantités de code, l'interconnexion croissante entre systèmes, et un large recours au test, avec cependant une combinatoire impossible à couvrir de manière exhaustive, que ce soit par des méthodes manuelles ou automatisées.

L'enjeu de la qualité logicielle devient alors d'optimiser les moyens de tests disponibles limités pour minimiser les risques (réglementaires, financiers, métier…).

En production et qualification de code, on a recours, dans le meilleur des cas, à une grande variété de méthodes

approchées, qui visent non pas à *prouver* la correction du code dans l'absolu, mais plutôt qu'il a un niveau de *qualité* suffisant pour répondre de manière *satisfaisante* aux besoins pour lesquels il a été conçu. On cherche aussi à garantir que le *processus* de son développement suit des bonnes pratiques laissant espérer par exemple que les erreurs découvertes seront bien corrigées et tracées pour éviter des régressions. On cherche enfin à améliorer directement la qualité du produit logiciel en industrialisant les différents types de tests, avec une démarche qui reste le plus souvent volontariste.

On observe ainsi une tendance forte au déploiement d'outils qui visent à améliorer le contrôle du code développé, en mêlant des métriques de différentes natures qui essayent de couvrir à la fois le code lui-même et le processus de développement. Le socle de ces outils est très souvent en logiciel libre, et Systematic a déjà contribué à financer un certain nombre de projets, allant des forges logicielles (Helios et Coclico, FUI 5 et FUI 7) à la qualimétrie (Squale et Squash, FUI 5 et FUI 10).

http://systematic-paris-region.org/fr/projets/helios
http://systematic-paris-region.org/fr/projets/coclico
http://systematic-paris-region.org/fr/projets/squale
http://systematic-paris-region.org/fr/projets/squash

Les enjeux croissants de la qualité logicielle

Depuis quelques années, on assiste à un essor de la qualité logicielle dans les domaines de l'informatique non critique sous l'effet de différents facteurs, parmi lesquels on peut mentionner :

- **L'augmentation des coûts de qualification**, qui prennent une part de plus en plus importante dans le coût des projets en raison de la sédimentation du patrimoine applicatif et de l'interconnexion croissante entre systèmes ;
- **La complexification de la qualification**, qui engendre le besoin d'industrialiser les tests, d'évoluer vers des centres de qualification transverses, et de professionnaliser la filière des métiers du test ;
- **L'externalisation des développements** (éventuellement à l'étranger), qui induit la nécessité d'un contrôle accru pour le donneur d'ordre. La mesure de la qualité logicielle devient alors un moyen de pilotage de la sous-traitance et de maîtrise de son patrimoine applicatif.

Le secteur de la qualité logicielle dans l'informatique non critique est en croissance et en mutation, et **touche à l'organisation même de l'entreprise. Il comporte encore une part d'empirisme et nécessite une standardisation.** La valorisation du retour sur investissement

de la qualité logicielle y est également complexe (il faut distinguer la « bonne » recette d'une application, de la recette d'une « bonne » application).

À ce titre, **c'est un terreau favorable à l'ingénierie collaborative caractéristique du monde du logiciel libre.** De nombreuses technologies s'y déploient : forges logicielles, démarches agiles, cloud, intégration continue… Là encore, la préférence va aux solutions libres, qui ne requièrent pas d'avoir à quantifier a priori de retour sur investissement, et qui ne nécessitent pas d'investissements ni coûts de licences.

D'autre part, le besoin de formation est réel sur la filière des métiers du test, ce qui favorise également le libre.

L'importance du processus de développement

La maintenabilité d'une application, c'est-à-dire sa facilité à évoluer au cours du temps, est étroitement liée à sa qualité logicielle. Il est intéressant de voir l'évolution des processus de développement, en particulier l'émergence des processus dits agiles et leur impact sur la qualité logicielle.

Apparus au début des années 1990, **les processus agiles ont été fortement popularisés par les grands acteurs du web** : Google, Facebook, Amazon, etc. Au contraire

des méthodes traditionnelles, qui visent à anticiper et découper au maximum les différentes étapes du processus de développement, les méthodes agiles misent sur la communication tout au long du projet entre les différents intervenants. La qualité logicielle est aussi traitée de façon très différente : là où les processus traditionnels vont tenter de la mesurer et de la valider en aval, les méthodes agiles en font un pré-requis permanent à la réussite du projet – l'hypothèse étant qu'un projet informatique complexe, tel un jardin, est l'objet permanent d'imprévu, la prise en compte efficace de l'imprévu étant partie intégrante de la qualité logicielle.

Parmi les nombreuses pratiques popularisées par les développeurs agiles, la principale est le développement piloté par les tests (TDD) : le test automatique est écrit avant même la réalisation de la fonctionnalité par le développeur. **En plus de fournir automatiquement un jeu de tests à rejouer plus tard pour éviter les régressions, le développeur est amené à réfléchir en permanence à la qualité de son code.**

Les plates-formes d'intégration continue (constituées de briques libres) s'enrichissent d'outils libres d'analyse de code, de rejeu de tests unitaires et de non-régression automatiques pour constituer de véritables chaînes de qualification continue (par exemple Git, Jenkins, Sonarqube, Squash TA, Selenium, Soap UI…).

Citons aussi la pratique du pair-programming, ou développement “à quatre yeux” sur le même ordinateur, prônée par le mouvement agile eXtreme Programming comme la façon la plus efficace de garantir la qualité du code produit. Cette pratique reste assez rare, alors que sa variante, le *pair-reviewing*, est très largement répandue dès que les enjeux de qualité sont importants.

Si toutes ces pratiques se sont rapidement imposées, c’est que **les outils nécessaires à leur mise en place ont été d’emblée mis à disposition de la communauté des développeurs sous la forme de logiciels libres** : Git, Jenkins, xUnit, etc.

Le logiciel libre, étroitement lié au développement des méthodes agiles, est indissociable du contrôle de la qualité logicielle.

Chapitre 4

Le logiciel libre et la qualité logicielle

Le logiciel libre a changé la nature même du développement logiciel : plutôt que de grands projets centralisés, dont le code source est gardé secret et dont l'évolution est donc, par force, balisée et encadrée, nous retrouvons dans le monde du logiciel libre un bouillonnement de projets, au cycle de développement souvent très court, alimenté de contributions venant de toutes parts, et dont le code source est accessible à quiconque.

L'essor du logiciel libre est fortement lié à la généralisation des formes d'interactions complexes entre utilisateurs, développeurs, chercheurs et autres acteurs. Ces interactions complexes ont été rendues possibles par le développement des réseaux à haut débit mais aussi par toute une batterie de technologies et d'applications rendant possibles ces interactions.

C'est ainsi tout un ensemble de problèmes nouveaux, différents et plus riches, que le logiciel libre pose.

Ainsi, dans le cas des applications de la recherche fondamentale portant sur la vérification et la preuve de programmes, il est clair qu'un cycle de développement

très serré pousse à explorer des techniques de preuves rapides et incrémentales.

Dans d'autres cas, tel celui des éditeurs de distributions GNU/Linux, il faut vérifier que les composants logiciels d'une distribution (plusieurs milliers) sont compatibles, et le restent au cours de son évolution. Cela nécessite des instruments d'analyse sophistiqués et nouveaux qui s'attaquent au problème réel de la maintenance d'un système complexe constitué de centaines voire de milliers de paquetages logiciels installés sur chacune des machines. Rien moins que la plateforme informatique d'aujourd'hui, allant des terminaux mobiles aux serveurs dans le cloud.

Plus généralement, **la construction d'applications fiables et certifiées basées sur du code évoluant rapidement et dont les nombreux contributeurs sont disséminés partout sur la planète pose des défis majeurs :**

- Comment trouver rapidement les erreurs dans les briques logicielles ?
- Comment tracer ces erreurs dans un historique de développement aussi complexe ?
- Comment améliorer la qualité de ces composants ?
- Comment tester des architectures logicielles de plus en plus complexes ?
- Comment en prouver la correction ?

En parallèle, **il faut également comprendre, améliorer et aider l'évolution des mécanismes de structuration des communautés d'acteurs autour des logiciels libres :**

- Quelles sont les propriétés des graphes formés par les réseaux collaboratifs ?
- Quels formats de données, quels types d'outils de recherche, quels genres d'instruments avancés intégrant des idées anciennes ou nouvelles venant des domaines du génie logiciel, des bases de données, des systèmes ou des interfaces homme-machine, sont les plus aptes à accompagner le mouvement ?
- Quels concepts de gestion de configurations logicielles sont adaptés au développement en logiciel libre ?

À terme, c'est bien le défi posé par la multiplication des projets en logiciel libre, et la complexification des réseaux d'interaction qui la sous-tendent, qu'il faudra relever : nous avons besoin de nouveaux outils théoriques pour apporter des réponses qui passent à l'échelle, et de nouveaux outillages solides qui puissent être incorporés dans la pratique industrielle.

À PROPOS DU GROUPE THÉMATIQUE LOGICIEL LIBRE DE SYSTEMATIC (GTLL)

Créé en 2007, le groupe thématique Logiciel Libre du pôle Systematic Paris-Region forme l'un des principaux viviers de l'Open Source en France.

Avec pour mission de « développer l'écosystème du libre en Île-de-France », le GTLL regroupe plus d'une centaine d'acteurs de l'innovation ouverte (PME, ETI, grands groupes et académiques). Il vise à favoriser la coopération, l'innovation et l'emploi, autour de projets de R&D collaborative et grâce à des actions de soutien au développement des entreprises innovantes (promotion, marketing, stratégie, aide à la recherche de financements...), dans le cadre des principes et des valeurs de l'open source. Il est à ce jour le plus important cluster au monde à focaliser ses activités de R&D collaborative sur les logiciels libres et les défis spécifiques à l'open source, comme l'after PC (l'ère informatique du Cloud, des mobiles et des objets connectés), la qualité logicielle, et le déluge des données. Après 7 ans d'existence, 40 projets de R&D collaborative consacrés au logiciel libre représentant un effort de R&D de près de 160 M€ ont déjà été financés grâce à l'aide du GTLL.

La qualité logicielle sous tous ses aspects est fondamentale pour l'évolution de notre société qui dépend toujours plus étroitement du logiciel, en particulier embarqué.

Si l'accessibilité des sources des logiciels libres est un facteur naturel de qualité logicielle, la croissance explosive du nombre de composants, et leur évolution rapide, posent des défis nouveaux. Pour les relever, il est indispensable de continuer l'effort de recherche, à moyen et long terme, pour concevoir des approches innovantes et ambitieuses sur les différentes axes évoqués.

À plus court terme, l'émergence et la maturité des outils libres d'amélioration de la qualité logicielle bénéficie à toute l'industrie du logiciel. Ces outils fournissent un levier de maîtrise, de contrôle et de valorisation du patrimoine logiciel, qui est partie intégrante des actifs de l'entreprise, et souvent au cœur de son efficience.

Après la phase d'adoption des technologies libres dans les couches basses des Systèmes d'Information d'Entreprise et d'Administration, l'étape suivante est celle de la généralisation d'un outillage libre pour garantir la qualité, l'agilité et la pérennité du SI.

Le GT Logiciel Libre de Systematic est décidé à accompagner ce mouvement, et a fait de la Qualité logicielle l'un des trois axes majeurs de sa feuille de route.

Le **groupe thématique Logiciel Libre** forme l'un des principaux viviers de l'open source en France. Il rassemble startups, PME, grands groupes, universités et centres de recherche autour d'une même vision des défis technologiques de demain et d'un engagement profond pour la compétitivité de notre économie.

Systematic
Paris Region Systems & ICT Cluster

Conception : Nord Compo