Nous défendons par ce manifeste une recherche française forte en génie logiciel, essentielle pour que la France soit un acteur majeur de la société numérique.

Avertissement : ce manifeste est non-exhaustif; la richesse de la recherche en génie logiciel ne peut, en aucun cas, être limitée à ces quelques lignes.

Le contraste est saisissant entre, d'un côté, l'omniprésence de l'informatique dans notre société et la facilité avec laquelle on peut écrire un petit programme, et d'un autre côté, la difficulté extraordinaire de garantir la correction, la fiabilité, les performances ou encore l'évolutivité d'un logiciel complexe comme on en rencontre aujourd'hui dans tous les pans de notre société : télécoms, aérospatiale, automobile mais aussi finance, santé, administration, etc.

Le génie logiciel ("*software engineering*" en anglais) repose sur l'application systématique des connaissances, méthodes, compétences scientifiques et technologiques nécessaires à la modélisation, l'implémentation, les tests, la maintenance et la documentation de ces logiciels qui structurent de plus en plus notre société. La recherche en génie logiciel fédère l'ensemble des activités qui permettent de définir, formaliser et évaluer les concepts, méthodes et outils qui portent la production logicielle actuelle et future.

La recherche en génie logiciel a ouvert la voie à de grandes avancées scientifiques dans différents domaines, et ce depuis les contributions séminales de Margaret Hamilton sur le programme spatial Apollo dans les années 60. Les grands systèmes logiciels des GAFAM ou même le système d'exploitation Linux en sont les illustrations d'aujourd'hui. De fait, depuis sa naissance il y a plus d'un demi siècle, la recherche en génie logiciel résout les problèmes posés par la complexité toujours croissante des logiciels, et est le vecteur des grandes innovations de notre époque.

Nous soutenons qu'à l'instar d'autres pays, acteurs mondiaux de la recherche en génie logiciel comme les USA, le Canada, l'Allemagne, le Royaume Uni ou la Chine, la France doit défendre activement sa position sur la scène internationale dans ce domaine, à la fois pour maîtriser le monde numérique et demeurer un précurseur.

Le développement des logiciels, de par sa nature, ne permet pas toujours de distinguer aisément les éléments de recherche—lorsqu'ils existent—des éléments d'ingénierie logicielle. L'objectif de ce manifeste est donc de livrer des éléments de compréhension de ce qu'est aujourd'hui la recherche en génie logiciel et ses enjeux.

Plutôt que de tenter d'en donner une vision exhaustive, nous avons choisi d'illustrer notre propos par quelques exemples que nous jugeons représentatifs. Ces exemples démontrent notamment l'importance d'une recherche forte en génie logiciel car, à l'origine de la plupart

des grands progrès en ingénierie logicielle, se trouve une contribution issue d'un laboratoire de recherche.

"Du monolithe à la galaxie"

Les logiciels qui nous entourent sont de plus en plus complexes. Ils doivent être disponibles, performants et répondre continuellement aux nouveaux besoins de leurs utilisateurs. Pour faire face à cette complexité, des styles architecturaux ont émergé au fil des années. Ceux-ci améliorent et accélèrent les processus de développement et favorisent un partage des connaissances et des bonnes pratiques en matière de conception logicielle.

Le style architectural REST (pour *REpresentational State Transfer*) est l'un des plus connus car il est communément adopté par la plupart des applications web modernes. Ce style est le résultat de la thèse de doctorat de Roy Fielding (soutenue en 2000 à l'Université de Californie à Irvine, USA). Au-delà de leur application au serveur Apache HTTP, les grands principes formalisés par REST ont promu la notion de ressource comme une abstraction nouvelle, centrée sur les données. Cette notion de ressource REST a rendu librement accessible les données et les services.

Si le succès de ce style architectural a profondément modifié les pratiques de conception en encourageant l'intégration des données exposées par les services en ligne, il met désormais au défi la communauté scientifique quant à la gestion de l'interopérabilité des connaissances, la gouvernance et la protection des données, et la connexion d'une constellation de flux informationnels dans des environnements extrêmement pervasifs. Pour consolider la numérisation effrénée d'une galaxie de domaines, sans mettre en danger l'intimité des usagers et le développement durable de notre planète, la recherche en génie logiciel travaille à formaliser de nouvelles architectures logicielles capables de s'ajuster à une grande variabilité de besoins et de conditions d'utilisation dans un monde toujours plus connecté.

"De la confiance dans le numérique"

Démontrer la correction des programmes étant dans le cas général un problème indécidable, des méthodes et des techniques ont dû émerger pour augmenter la confiance dans le logiciel et pour convaincre à la fois qu'il réalise ce qui est attendu (validation) et qu'il le fait correctement (vérification). C'est un aspect où les champs entre théorie et pratique sont particulièrement denses et incontournables.

Par exemple, CompCert est un compilateur C optimisant prouvé qui garantit mathématiquement que le programme assembleur généré a la même sémantique—*i.e.*, aura les mêmes effets—que le programme source fourni en entrée. Même si cette réussite académique (en grande partie française due à Xavier Leroy, professeur au Collège de France et ses collaborateurs) constitue un véritable tour de force et une étape importante, il reste à garantir que le code source réponde correctement à un cahier des charges complexe, et que l'exécution du programme par un ou plusieurs processeurs et avec des

données ou dans des conditions potentiellement imprévues ne sera pas sujette à des comportements inattendus.

Pour répondre aux problèmes de la validation et de la vérification, de nombreux progrès ont été réalisés, notamment dans la gestion des tests et des méthodes formelles. Maintenir la confiance dans le logiciel reste un défi ouvert propre au génie logiciel qui est critique dans de nombreux domaines. C'est le cas en particulier de la confiance dans les systèmes intégrant du *machine learning* car on ne sait aujourd'hui ni comment les tester, ni comment mesurer la confiance qu'on peut avoir dans ces tests avec des métriques de qualité qui restent à inventer. Avancer vers une validation et une vérification des systèmes logiciels qui dépendent des impératifs du monde réel, tels que les systèmes cyber-physiques, nécessite de nouveaux paradigmes.

"Des langages pour les machines aux langages pour les humains"

Programmer consiste à rédiger des textes dans des langages respectant une grammaire analysable par des procédés automatiques et destinés, après traduction, à être exécutés par des ordinateurs. Un programme doit pouvoir être lu et manipulé à la fois par l'être humain, qui l'écrit ou l'échange avec d'autres êtres humains, et par la machine. La recherche en programmation se préoccupe de définir des abstractions et des mécanismes facilitant l'écriture et l'évolution des logiciels, notamment par une bonne séparation des préoccupations pour maîtriser la complexité et la variabilité des logiciels. Sans recherche en génie logiciel, on en serait toujours à programmer en langages d'assemblage, proches de la machine mais terriblement difficiles pour les humains.

Les langages orientés objets, massivement adoptés par l'industrie, sont par exemple nés d'une lignée de langages de programmation issus de laboratoires de recherche, dont le premier est dû à Ole-Johan Dahl et Kristen Nygaard (Simula 67, Université d'Oslo). Maîtriser ses propres langages de programmation généralistes est devenu un enjeu pour les géants du numérique (Go chez Google, Java chez Oracle, Swift chez Apple, C# chez Microsoft, etc.). Or ces langages ont des filiations directes avec leurs ancêtres issus des laboratoires de recherche.

Au-delà de ces langages de programmation généralistes disposant de concepts puissants destinés aux informaticiens, un des défis de notre époque est de permettre à un plus grand nombre l'accès à la pensée informatique en ouvrant la programmation aux experts (non informaticiens) de différents domaines—qu'ils soient scientifiques ou industriels. Aussi, les recherches sur l'ingénierie des langages visent aujourd'hui à faciliter non seulement la conception de langages dédiés mais également la génération d'environnements de développement intégrés plus naturellement adaptés aux concepts et aux contraintes d'experts de domaines très hétérogènes.

"De la fabrication ad-hoc aux procédés agiles de production"

Plus de trente ans après, le constat de F. Brooks reste vrai. « *Peu importe le génie humain, il sera toujours difficile de faire face aux quatre défis qui caractérisent un logiciel et qui rendent son développement aussi laborieux : sa complexité, sa conformité, son évolutivité constante/variabilité et son côté impalpable* ». Ces défis sont d'autant plus prégnants, qu'aujourd'hui, la plupart des entreprises tendent à devenir des « entreprises orientées logiciel », et positionnent leur système informatique au cœur de leur métier. Sa performance a un impact direct sur celle de l'entreprise.

Que ce soit les logiciels de la NASA qui ont contribué aux premiers pas de l'homme sur la lune ou aux 130 millions de lignes de code qui tournent dans un Airbus A380, cette complexité n'aurait pas été domptée sans les contributions de la recherche dans le domaine de l'ingénierie des procédés dont les approches de développement Agile, aujourd'hui très répandues dans l'industrie, sont issues. Elles tirent leur principes de l'idée de processus en spirale introduite par Barry Boehm (Université de Caroline du Sud) en 1985 et s'inspirent fortement de la méthode Scrum (présentée à la conférence OOPSLA 1995).

Les recherches dans ce domaine doivent répondre à de nombreux défis dont l'intégration de multiples expertises (sécurité, sûreté…) dans le cycle de vie du logiciel, l'optimisation des ressources (*green computing*) ou des aspects liés à l'éthique.

“Revisiter le passé pour maîtriser le futur”

La problématique de la gestion des évolutions apportées à une grande diversité d'artefacts logiciels est l'une des problématiques majeures du domaine de la maintenance logicielle. Ces évolutions sont souvent réalisées de manière collaborative par plusieurs intervenants travaillant souvent dans des environnements différents (par exemple, des logiciels libres soutenus par des équipes internationales).

Le logiciel Git, qui permet à des développeurs d'éditer et de gérer du code source de manière collaborative, et qui est issu du modèle de développement du noyau Linux, est un des exemples phares de l'aboutissement des recherches menées dans ce domaine. Si le succès de Git est désormais planétaire et commercial, via notamment la plate-forme collaborative GitHub, il est important de noter que ses fondements prennent racine dans des travaux universitaires, notamment ceux menés en France par Jacky Estublier (CNRS, Grenoble) dans les années 80.

De nombreux challenges restent néanmoins ouverts dans ce domaine. Parmi eux, citons la problématique du stockage et de l'exploitation de la masse des évolutions. La croissance phénoménale du nombre de projets logiciels développés au cours des 50 dernières années a introduit de fait une explosion du nombre d'évolutions à gérer. L'initiative Software Heritage menée par Roberto Di Cosmo (Université Paris Diderot) a pour objectif de sauvegarder le patrimoine logiciel de l'humanité. Un autre challenge réside dans la difficulté à propager et filtrer les évolutions réalisées sur les artefacts d'un logiciel (code, modèle, documentations, etc.) vers d'autres artefacts (du même logiciel ou d'autre logiciels) qui en dépendent. Ces challenges et beaucoup d'autres nécessitent de poursuivre une recherche

en génie logiciel sans laquelle la maintenance des logiciels existants sera inéluctablement mise en profonde difficulté dans un monde en perpétuel changement.

Signataires

Nous, signataires de ce texte, avons souhaité, au travers de ce manifeste, expliciter et illustrer l'importance d'une recherche forte en génie logiciel, en mettant en exergue sa portée par quelques exemples qui, par le passé, ont démontré son impact fondamental pour l'essor du numérique que nous connaissons et qui doit continuer à être soutenue pour conserver un rôle prépondérant dans un écosystème en pleine expansion. Nous remercions les relecteurs qui, par leurs retours, ont permis d'améliorer ce texte.

Nous invitons donc quiconque partageant la même intention à l'utiliser pour communiquer et défendre la place du génie logiciel dans la recherche académique française dans les années à venir.

- Mathieu Acher
- Nicolas Anquetil
- Vincent Aranega
- Jean-Christophe Bach
- Olivier Barais
- Reda Bendraou
- Antoine Beugnard
- Xavier Blanc
- Mireille Blay-Fornarino
- Simon Bliudze
- Aurélien Bourdon
- Jean-Michel Bruel
- Hugo Bruneliere
- Bernard Carré
- Christophe Casseau
- Stéphanie Challita
- François Charoy
- Benoit Combemale
- Steven Costiou
- Fabien Dagnat
- Gwendal Daniel
- Thomas Degueule
- David Delahaye
- Marcus Denker
- Christophe Dony
- Stéphane Ducasse
- Laurence Duchien
- Sophie Ebersold
- Anne Etien

- Jean-Rémy Falleri
- Marie-Pierre Gervais
- Fahad R. Golra
- Sylvain Guérin
- Lom Messan Hillah
- Marianne Huchard
- Jean-Marc Jezequel
- Daniel Le Berre
- Alexandre Le Borgne
- Olivier Le Goaër
- Érik Martin-Dorel
- Matias Martinez
- Martin Monperrus
- Floréal Morandat
- Johan Moreau
- Sébatsien Mosser
- Christophe Moustier
- Pierre-Alain Muller
- Adel Noureddine
- Quentin Perez
- Pascal Poizat
- Guillermo Polito
- Damien Pollet
- Clément Quinton
- Jean-Baptiste Raclet
- Laurent Réveillère
- Romain Rouvoy
- Lionel Seinturier
- Chantal Taconet
- Pablo Tesone
- Chouki Tibermacine
- Simon Urli
- Christelle Urtado
- Sylvain Vauttier
- Vincent Iampietro

Signez en ouvrant une *merge request* :

- Cliquez sur le lien ci-dessus.
- Connectez-vous (*e.g.*, via github).
- Il vous demande de *forker*…
- Modifiez ce texte (`index.md` , via l'éditeur en ligne) en ajoutant votre nom dans la liste.
- Enregistrez en expliquant vos modifications.
- C'est fini, enfin presque, il ne vous reste plus qu'à largement diffuser ce manifeste !

Manifeste GL

Manifeste GL
manifeste-gl@groupes.renater.fr

Manifeste pour la recherche en Génie Logiciel