

Vers des Logiciels Éco-responsables

Olivier Le Goaer, Adel Noureddine, Franck Barbier, Romain Rouvoy, Florence Maraninchi

▶ **To cite this version:**

Olivier Le Goaer, Adel Noureddine, Franck Barbier, Romain Rouvoy, Florence Maraninchi. Vers des Logiciels Éco-responsables : Le génie logiciel au défi de la sobriété écologique. 2021. hal-03264970v1

HAL Id: hal-03264970

https://hal.science/hal-03264970v1

Preprint submitted on 18 Jun 2021 (v1), last revised 13 Sep 2021 (v2)

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.

Vers des Logiciels Éco-responsables
Le génie logiciel au défi de la sobriété écologique

Olivier Le Goaër, Adel Noureddine, Franck Barbier*
Romain Rouvoy†
Florence Maraninchi‡

Depuis son origine, le génie logiciel sest attaché à élaborer des méthodes, concepts et outils pour réduire significativement les coûts liés au développement (maintenance, réutilisation...) des logiciels, garantir leur intégrité, fiabilité... Cependant, pour relever le défi de la transition écologique, une nouvelle préoccupation doit être intégrée dans tous ces artefacts : le coût environnemental. Ainsi, le génie logiciel ń éco-responsable ż est la branche qui sintéresse à lefficience énergétique et à la durabilité des logiciels. Nous dégageons notamment 6 grandes pistes de recherche qui, combinées, peuvent apporter des réponses à la hauteur des enjeux.

Mots-clés : énergie, environnement, éco-conception, génie logiciel

1 Introduction

Selon la dernière étude Green IT [Green IT, 2019], à léchelle planétaire, lempreinte environnementale du numérique équivaut à un continent de 2 à 3 fois la taille de la France, et à 5 fois le poids du parc automobile français. Les fabricants matériels optimisent lefficience de leurs équipements depuis quarante ans tandis que les éditeurs de logiciels font le chemin inverse en entassant des couches applicatives les unes sur les autres. Peut-être que laspect purement immatériel du logiciel rend difficilement perceptibles les émissions carbone en bout de chaîne, mais cest un fait. ń Lorsque la mémoire était comptée, les développeurs informatiques avaient lhabitude décrire du code synthétique et efficace. Aujourdhui, ces préoccupations ont disparu et lon assiste à une véritable inflation des lignes de code, ce qui signifie des calculs plus longs et plus gourmands en électricité ż, raconte Anne-Cécile Orgerie dans [CNRS, 2010]. Mais, devant la menace écologique planétaire, la tendance commence à se renverser et lon se pose à nouveau la question de la frugalité, dans un contexte où les usages du numérique ont littéralement explosés. Toute la production logicielle est concernée par léco-conception, mais ce sont les types dapplications ayant la plus forte base dutilisateurs qui produiront les effets les plus visibles, comme le web, le mobile et le cloud, sans même parler des chiffres vertigineux de lIoT. Cest en ce sens que de nouveaux défis se posent à la communauté du GDR GPL. Toutefois, il y a lieu de bien les caractériser et de les éclairer en termes de démarches de recherche à amplifier ou démarrer.

Lidée déco-conception logicielle a fait du chemin puisque lappel à défis pour les journées du GDR GPL à Pau en Mars 2010 comportait déjà un défi lié à ń *la réification de lénergie dans le domaine du logiciel* ż [Menaud et al., 2010]. Neuf ans plus tard, la conférence douverture du GDR GPL 2019 est ń *Quels défis pour le développement durable des logiciels ?* ż [Romain Rouvoy, 2019]. Peut-être que

*Université de Pau et des Pays de l'Adour - E2S
†Université de Lille / Inria / IUF
‡Université Grenoble Alpes / Verimag

le temps est venu de créer un groupe de travail du GDR dédié à cette thématique afin de promouvoir ces activités plus vertueuses en matière de développement logiciel éco-responsable.

2 Pistes de recherches

Nous dégageons ci-dessous 6 pistes de recherche visant à produire des logiciels plus efficients en énergie et plus durables (ou soutenables), ces deux aspects étant primordiaux dans le domaine de léco-conception.

2.1 Savoir mesurer, comprendre et prédire la consommation énergétique

La première chose à faire pour maitriser la problématique de la consommation dénergie par les couches logicielles est de disposer des outils (e.g., *energy profilers*) pour la mesurer au runtime. Pas si simple lorsque linformatique est répartie, que les couches sentremêlent, sont masquées (que fait lOS par exemple des services requis par une machine virtuelle ?) et que les couches matérielles entrent inévitablement en jeu. Sans compter que loutillage de mesure engendre lui-même une consommation quil faut être capable de retrancher au bilan énergétique. Dautre part, si mesurer un logiciel bien défini semble aisé, mesurer à grande échelle des millions de logiciels nécessite de faire de grands progrès du côté de la génération automatique de tests afin de se baser sur des scénarios dusage réalistes lors des mesures.

Au-delà des mesures et des observations empiriques, il faut comprendre la chaîne dinstructions amenant à cette consommation. Lautomatisation de ce processus doit favoriser la récolte de données de consommation et ainsi aider à comprendre plus précisément les facteurs impactant cette consommation au niveau de larchitecture logicielle. Cette compréhension est une étape cruciale pour construire des logiciels éco-responsables dès la phase de conception. Ainsi, la prédiction de la consommation énergétique logicielle dès les phases de conception pose un défi majeur pour la communauté du génie logiciel. Léco-conception logicielle doit notamment pouvoir se projeter sur les contextes dexécution ciblés, et proposer des modèles de prédictions réalistes des usages finaux.

2.2 Améliorer la qualité des productions logicielles

Une autre piste intéressante est de considérer que lefficacité énergétique est un attribut de qualité (comme la sécurité ou laccessibilité, par exemple) des logiciels modernes. Le corollaire est que les logiciels qui contiennent des défauts de conception provoquent *de facto* des surconsommations ; il faut être capable de les trouver et idéalement de les corriger ou de les isoler. Sur les équipements fonctionnant sur batterie (e.g., smartphone), la surconsommation a même un autre effet pervers puisquelle peut user prématurément la batterie dont le nombre de cycles de charge est limité. Cette usure engendre donc irrémédiablement une pollution dans le monde réel.

On trouve dans ce champ dapplication les travaux sur lanalyse statique ou dynamique de code, les environnements dexécutions, ainsi que les tests logiciels en général. On peut même imaginer de nouveaux langages de programmation et de nouveaux outils formels qui garantissent, par construction, lefficience énergétique du code écrit par les développeurs. Enfin, des outils CASE dédiés et la génération automatique de code ont le potentiel daugmenter lefficience énergétique globale des lignes de production logicielle.

2.3 Dégraisser les logiciels

Une autre approche consiste à sattaquer au surplus logiciel des ń obésiciels ż, ces logiciels en surpoids par rapport à la fonction quil doivent remplir. Lingénierie des exigences est ici concernée, pour intégrer ń juste ce quil faut ż de fonctionnalités dans les productions logicielles : la ń wise-tech ż. La variante extrémiste de cette approche étant la ń *low-tech* ż, une sorte de décroissance logicielle choisie. Des travaux empiriques peuvent mettre en évidence cette dérive, en observant par exemple lévolution de *frameworks* populaires (de la version 1.0 à nos jours) ou simplement de lévolution du poids moyen dune application au cours de la dernière décennie. La gestion des dépendances entre modules est aussi un sujet majeur puisque lon se retrouve rapidement à incorporer beaucoup alors même que le logiciel fait peu.

Dapparence plus anecdotique, cette piste a en réalité un potentiel énorme puisque que lon touche ici au fléau de renouvellement prématuré des équipements électroniques qui doivent rester capables dexécuter les obésiciels, et donc à lépuisement des ressources naturelles (métaux, terres rares, eau) pour en fabriquer sans cesse de nouveaux toujours plus avides dénergie.

2.4 Sensibiliser les utilisateurs, engager les développeurs

Le génie logiciel éco-responsable peut et doit aussi attaquer le problème du côté des utilisateurs finaux et de celui des développeurs. Il est en théorie possible de modifier le comportement des utilisateurs en introduisant une forme de label/score écologique dans lindustrie du logiciel, comme cela se fait réglementairement dans de nombreux autres domaines. Lutilisateur doit ń consommer ż du logiciel de manière éclairée.

Quant aux développeurs, ils sont à la fois une partie de la solution et une partie du problème. Les études récentes auprès des développeurs indiquent un intérêt croissant vers léco-conception logicielle, mais ces derniers manquent dinformations et doutils nécessaires à cette réalisation [Manotas et al., 2016]. Dis autrement, si le GL leur fournit les bons outils, alors ils sen empareront. Mais dans le même temps, la problématique du développement et de la maintenance logicielle dont les chaines dintégration continue et les nombreux postes mis en jeu peuvent avoir un impact non négligeable pour la reproduction et léventuelle correction du moindre bug. Autre exemple de la dualité de la problématique : les *repair bots*, dont les nombreux essais pour corriger un bug logiciel peuvent engendrer des consommations déraisonnables, sous couvert dautomatisation et déconomies humaines.

Enfin et surtout ; ce travail de fond ne peut pas être mené sans mettre les étudiant.e.s de nos écoles et universités dans la boucle, car ils sont à la fois des utilisateurs et les développeurs de demain. Léco-conception logicielle doit être intégrée dans loffre de formation dès maintenant.

2.5 Avoir une vision holistique incluant les acteurs humains

La complexité de la dimension énergétique dans le contexte logiciel (comparée aux autres critères non fonctionnels, tels que la fiabilité) vient aussi du rôle prépondérant de lenvironnement dexécution proche et lointain du logiciel. Les différentes couches logicielles doivent ainsi intégrer la dimension énergétique : au niveau intergiciel (middleware), lors de la programmation orientée composant, au niveau des conteneurs logiciels (tels que Docker, Kubernetes), ou pour les services logiciels. La montée en popularité du logiciel en tant que service (*Software-as-a-Service*, SaaS), tend à réduire le logiciel client à une interface dun service logiciel complexe installé sur le cloud, avec limplication dune multitude de couches consommatrices dénergie.

Lexplosion ces dernières années des objets connectés amène aussi son lot de complexité : réseau, puissances faibles et distribuées, sécurité et vie privée, et surtout le retour de lhumain (et du vivant

en général) comme un acteur majeur des systèmes logiciels. Le défi est ainsi de sortir de lapproche technique ou architecturale ń par silo ż, c.-à-d. optimiser des systèmes logiciels par couche, ou par domaine dapplication, et détudier et optimiser le système dans son ensemble, y compris la boucle dinteraction avec lhumain.

Le logiciel éco-responsable doit donc être en mesure de sautooptimiser, mais aussi doptimiser son environnement (objets connectés, maison/industrie du futur), et de pousser aux changements comportementaux nécessaires à une transition énergétique et écologique plus large.

2.6 Créer un nouveau paradigme du logiciel durable

Et si le numérique actuel était une voie sans issue ? Cette dernière piste de recherche, la plus disruptive il faut bien le dire, consiste à imaginer un futur en faisant table rase du passé, dans lequel les logiciels seraient durables, centrés sur des objectifs de frugalité et résilience. Il ne sagit pas daméliorer lexistant, mais bien de repenser totalement une infrastructure numérique utilisant au mieux les ressources matérielles existantes afin datteindre la frugalité et la résilience. Cela revient donc à appliquer au logiciel, et pour les propriétés de frugalité et de résilience, une approche qui a déjà été appliquée au matériel pour la propriété de prédictibilité temporelle dans le projet CompSOC [CompSOC].

Le but est aussi de repenser les solutions techniques dans une réflexion plus large sur le rôle du numérique dans nos systèmes socio-techniques et sur les usages et les besoins. Cette piste sinscrit dans les démarches *low-tech* plus ou moins globales et de déconnexion voire de dénumérisation (*Collapse Informatics* [Tomlinson et al. 2013], *CollapseOS* [CollapseOS]). Elle peut paraître idéaliste et éloignée des contraintes économiques, mais reste cependant scientifiquement intéressante ne serait-ce que pour permettre de mesurer la distance qui existe entre la complexité de certains de nos systèmes actuels et des solutions en quelque sorte minimales, libérées du poids de la compatibilité ascendante et de lhistorique.

Une piste possible dans ce thème serait danalyser les systèmes actuels pour comprendre quelle dégradation de leurs fonctionnalités serait observée sils nétaient pas alimentés en énergie et connectés 24h sur 24 et 7 jours sur 7

3 La recherche en GL éco-responsable : exemples de projets

La recherche en génie logiciel éco-responsable est une réalité, en témoigne ces quelques projets en cours :

- La société GreenSpector fournit un outil de mesure de lénergie pour mobile (Android et iOS). Elle a collaboré avec lUniversité de Lille et Inria, sur des projets et études tels que Web Energy Archive du Green Code Lab ;
- LUniversité de Lille et Inria ont développé des modèles et outils de mesure de lénergie logicielle dont notamment PowerAPI [Colmant et al., 2018]. Ces outils sont en passe dêtre adoptés par la communauté industrielle, comme Orange ou Davidson consulting ;
- Le projet Creedengo, né des efforts conjoints de la société Snapp et de lUniversité de Pau et des Pays de lAdour concerne la création dun pool de règles orientées énergie dédiées aux applications mobiles et implantées dans un ń Linter Vert ż [Olivier Le Goaer, 2019] ;
- Lappel à projets PERFECTO 2018 de lADEME a financé 5 projets en éco-conception logicielle, ciblant essentiellement le Cloud (Data Centers) et lIoT. Des équipes de recherche de Mines ParisTech et de lUniversité de Nantes font partie des lauréats.

— Le projet ANR HELP visait à fournir aux développeurs dapplications des simulateurs de plates-formes matérielles capables de mettre en évidence limpact du logiciel sur les états de consommation du matériel [HELP16].

4 Bibliographie

1. [CNRS, 2018] Numérique : le grand gâchis énergétique, Le Journal du CNRS, 2018, https://lejournal.cnrs.fr/articles/numerique-le-grand-gachis-energetique
2. [Menaud et al. 2010] Jean-Marc Menaud, Adrien Lèbre, Thomas Ledoux, Jacques Noyé Pierre Cointe, Rémi Douence et Mario Südholt (Ecole des Mines de Nantes / INRIA / LINA) Vers une réification de l'énergie dans le domaine du logiciel : L'énergie comme ressource de première classe, Session défis, Journées du GDR GPL, Pau 2010.
3. [Romain Rouvoy, 2019] Quels défis pour le développement durable des logiciels ? Conférence invitée, Journées GDR-GPL, Juin 2019
4. [Green IT, 2019] Empreinte environnementale du numérique mondial, https://www.greenit.fr/etude-empreinte-environnementale-du-numerique-mondial/
5. [Manotas et al., 2016] An Empirical Study of Practitioners' Perspectives on Green Software Engineering, 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, 2016, pp. 237-248.
6. [Olivier Le Goaer, 2019] Linter vert pour l'écoconception logicielle, Green Days, Juin 2019
7. [Colmant et al., 2018] Maxime Colmant, Romain Rouvoy, Mascha Kurpicz, Anita Sobe, Pascal Felber, Lionel Seinturier : The next 700 CPU power models. Journal of Systems and Software 144 : 382-396 (2018).
8. [Perfecto, 2018] Lauréats de lappel à projets de recherche ADEME PERFECTO 2018. https://presse.ademe.fr/wp-content/uploads/2018/10/Laureats-AAP-Recherche-PERFECTO-2018-ADEME.pdf
9. [CompSOC] http://compsoc.eu/
10. [Tomlinson et.al, 2013] Bill Tomlinson, Eli Blevis, Bonnie Nardi, Donal Patterson, M. Six Silberman, Yue Pan : Collapse Informatics and Practice : Theory, Method. ACM Transactions on Computer-Human Interaction (2013).
11. [CollapseOS] https://collapseos.org/
12. [HELP16] Modeling Power Consumption and Temperature in TLM Models – Matthieu Moy, Claude Helmstetter, Tayeb Bouhadiba, Florence Maraninchi https://ojs.dagstuhl.de/index.php/lites/article/view/LITES-v003-i001-a003