Analyse de cycle de vie adapté au logiciel : analyses, outillages et recommandations

Henri-Pierre Charles Henri-Pierre.Charles@cea.fr[1], Xavier Zeitoun Xavier.Zeitoun@cea.fr[2], Jean-François Berrée Jean-Francois.Berree@cea.fr[2], Maxime Péralta Maxime.Peralta@cea.fr[1], and Thibault, Makan Halter Thibault.Halter@cea.fr'[1]

[1]Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
[2]Univ. Paris-Saclay, CEA, List F-91120, Palaiseau, France

12 décembre 2024

Abstract : Les impacts économiques et environnementaux des logiciels sont plus complexes à évaluer que celui des équipements informatiques, car les logiciels sont immatériels. Ce sont néanmoins ces derniers qui, en demandant toujours plus de capacités de calcul et de mémoire, sont responsables de l'obsolescence du matériel.
La loi de Wirth [5] indique que les logiciels évoluent plus rapidement que le matériel, mais cette loi est empirique et les méthodologies pour la démontrer sont difficiles à mettre en œuvre.
Dans cet article, nous présentons une partie d'un projet de recherche dont l'objet est de fournir (1) une méthodologie de mesure de l'évolution de la complexité d'un logiciel d'usage commun puis (2) des outils permettant de vérifier que les nouvelles versions d'un logiciel permettent son fonctionnement sur les machines anciennes.
L'objet du projet englobant a pour objectif de développer des moyens de mesures que nous structurons autour du terme "prescriptions du logiciel sur le matériel".
Keywords : Développement logiciel, sobriété numérique, numérique frugal, obsolescence matérielle, loi de Wirth

I Introduction

L'évaluation d'un système informatique, comme un *smartphone* ou un ordinateur portable, s'effectue principalement par l'analyse de la chaîne de la valeur de ses composants. Cela permet de connaître les acteurs économiques, la compréhension des coûts et de faire une analyse macro-économique. Plus récemment, l'analyse du cycle de vie (ACV) permet d'avoir une vision plus précise sur les aspects soutenabilité environnementale, mais également d'avoir une méthodologie pour sa réalisation.

Ces approches sont satisfaisantes pour l'analyse de la partie matérielle d'un système informatique. Mais si l'on pousse l'analyse sur sa durée de vie, on constate que c'est la partie logicielle qui est principalement responsable de l'obsolescence.

Certains logiciels ont une durée de vie largement supérieure à la durée des systèmes informatiques. Par exemple le logiciel LibreOffice [3] a été publié en 2011, le logiciel LaTeXen 1983 et fonctionnent toujours sur tous les ordinateurs personnels depuis cette date.

La loi de Moore [2] a guidé l'évolution des performances des ordinateurs depuis les années 50. La "loi de Wirth" [5], qui explique que l'évolution des logiciels est plus rapide que celle du matériel, a été énoncé en 1995. Elle est plus considérée comme une attaque contre les modèles de développement logiciels que comme une loi reposant sur des unités quantifiables.

C'est pourtant cette évolution logicielle qui rend obsolète les composants électroniques. Cette évolution a été mesurée pour certaines applications. Par exemple cet article [4] évalue pour l'IA, les quantités de mémoire et de FLOPS (Floating Point Operation per Second) nécessaire pour l'apprentissage.

Dans cet article, nous décrivons un projet d'étude sur l'évolution logicielle. L'idée principale est d'évaluer l'évolution de logiciels dans le temps et d'évaluer la croissance de leur complexité pour réaliser des tâches simples sur des ordinateurs classiques. Ces résultats seront mis en parallèles avec l'évolution des processeurs dans le même temps.

Il s'agit donc de trouver des métriques pertinentes pour évaluer l'évolution logicielle, les mesurer sur un grand nombre de versions d'un même logiciel et de comparer cette évolution à celle des ordinateurs.

On pourrait dire différemment qu'il s'agit de mettre en évidence la loi de Wirth sur des scénarios d'usage courants.

Des études ont déjà été réalisés, par exemple pour l'IA [4], ou pour la bureautique [1]. La première utilise la métrique du nombre d'opérations en nombre flottant nécessaire sur des applications différentes en IA, la seconde la quantité de mémoire nécessaire.

II Scénarios envisagés

Identifier des métriques est indispensable afin d'utiliser une approche scientifique et également dans le but de pouvoir chiffrer des objectifs à atteindre.

Dans cette étude, nous souhaitons étudier les métriques que l'on peut mesurer lors de l'exécution des différentes versions d'un même logiciel. Une liste non exhaustive serait de compter : le nombre d'instructions pour un scénario, le nombre d'accès mémoire, le nombre d'appels à des librairies, le nombre d'entrées / sorties, etc.

Dans cette étude, nous proposons de travailler sur trois scénarios d'usage courant. :

Traitement de texte : il s'agira d'évaluer la saisie d'un texte simple, un document d'environ une page sur les différentes versions d'un outil de traitement de texte : `libreoffice`.

Rendu de page web : il s'agira d'évaluer le rendu d'une page web dans un navigateur internet : `firefox`

Génération de page web : il s'agira d'évaluer la génération d'une page web par un CMS mettant en œuvre, un serveur web, un interpréteur PHP utilisé par un CSM simple, une base de donnée.

Ces scénarios sont relativement simples, mais servirons à bâtir une méthodologie qu'il sera ensuite possible d'intégrer dans des outils logiciels qui aideront à respecter des recommandations.

III Scénarios

Pour chacun des scénarios envisagés, nous allons faire fonctionner les différentes versions de ce logiciel pour la réalisation des différents scénarios. Nous collecterons des statistiques sur l'exécution (nombre d'instructions utilisées, nombre d'accès à la mémoire, etc). Les différentes versions étudiées s'étaleront sur toute la durée de vie des logiciels (en général sur une durée supérieure à 10 ans).

Ces statistiques permettront de montrer l'évolution des logiciels dans le temps.

Nous collecterons également les caractéristiques des machines qui ont existé sur la période équivalente. Nous relierons ensuite les caractéristiques de ces machines à des impacts environnementaux, principalement pour la fabrication et l'utilisation du matériel, en utilisant la méthode ACV.

Nous pourrons ainsi vérifier par la pratique la véracité de la loi Wirth et, le cas échéant, son implication en terme d'impact.

Si elle est vérifiée, nos résultats permettrons à un développeur de logiciel de vérifier si la nouvelle version d'un logiciel peut fonctionner sur des machines plus anciennes.

IV Méthodologie

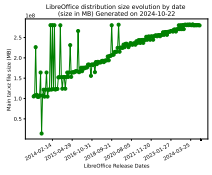
L'environnement expérimental sera le suivant :

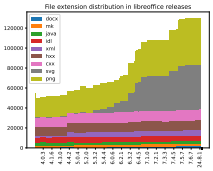
- Environnement : linux `Ubuntu`
- Émulation : `QEMU` qui est un émulateur de système qui permet
  - de mesurer le nombre d'instruction exécutées et de les catégoriser (via un plugin).
  - de quantifier les appels systèmes
- Automatisation : l'outil `xdotool` permet d'automatiser les scénarios en simulant des interactions utilisation,

Il est également possible d'intégrer d'autres métriques mesurées lors de la construction du programme final par un compilateur:

- compter le nombre de librairies nécessaires
- réaliser des métriques sur l'évolution du code

Par exemple les deux graphes suivant indique (1) l'évolution de la taille du code des différentes distributions et (2) l'évolution de la distribution des types de fichiers dans les distributions. On peut constater que ce ne sont pas les fichiers programme (C++) qui évoluent le plus.





V Résultats attendus

On peut imaginer que les résultats auront une forme ressemblant à la figure 1. L'axe vertical correspond à l'évolution en complexité, l'axe vertical la période de temps étudiée pour les différentes versions d'un logiciel.

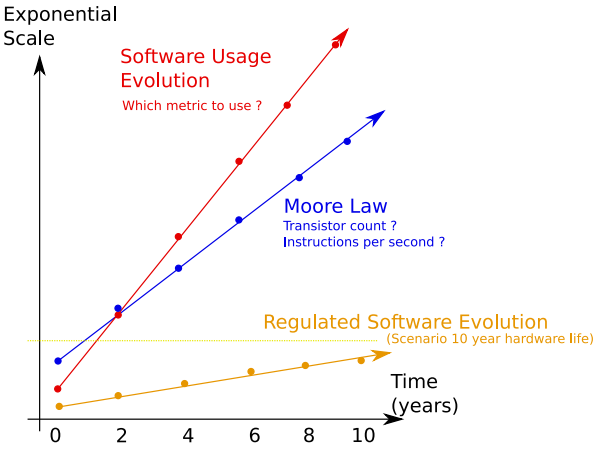


Figure 1: Résultats possibles

- La ligne rouge correspondant à l'évolution de la complexité du logiciel sur plusieurs années. L'étude permettra de définir quelle est la métrique la plus pertinente à utiliser. Cela pourrait être le nombre d'instructions utilisées pour effectuer un scénario.

- La ligne bleue correspondant à l'évolution du matériel sur la même durée. L'unité utilisée pourrait être le nombre maximum d'instructions exécutées par seconde.

- La ligne orange représente l'évolution d'un logiciel dont les concepteurs accepteraient de limiter les développements dans les nouvelles versions par des recommandations chiffrées.

  Par exemple des développeurs pourraient mentionner dans leur logiciel : "le nombre d'instructions nécessaire à la réalisation d'un scénario utilisateur par la nouvelle version de notre logiciel respecte une évolution frugale. Cette évolution permet d'utiliser notre logiciel sur des machines d'il y a 10 ans".

Les résultats de cette étude pourraient permettre de définir des outils indiquant au programmeur quelle attitude adopter pour les nouvelles versions de son logiciel.

Si l'objectif est de pouvoir faire tourner les nouvelles versions du même logiciel sur des machines vieilles de 10 ans, (scénario orange) il devra limiter les développements

VI Conclusion

Cet article décrit une partie des tâches d'un projet plus large dont l'objectif est de développer des moyens de mesures que nous structurons autour du terme "prescriptions du logiciel sur le matériel".

Nous chercherons à optimiser la gestion de l'obsolescence en enrichissant nos méthodes de conception avec une solution d'aide à la décision permettant de cibler un logiciel optimisé répondant au besoin fonctionnel d'un système et prévenant un type d'obsolescence.

References

[1] Frédéric Bordage. x171 : la croissance du poids de nos logiciels, Aout 2020. URL: `https://www.greenit.fr/2020/08/18/x171-la-croissance-du-poids-de-nos-logiciels/`.

[2] Gordon E Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff. *IEEE solid-state circuits society newsletter*, 11(3):33–35, 2006.

[3] The Document Foundation. Libreoffice, 2020. Version 7.0. URL: https://www.libreoffice.org/download/download/.

[4] Gaël Varoquaux, Alexandra Sasha Luccioni, and Meredith Whittaker. Hype, Sustainability, and the Price of the Bigger-is-Better Paradigm in AI, September 2024. arXiv:2409.14160 [cs]. URL: http://arxiv.org/abs/2409.14160, doi:10.48550/arXiv.2409.14160.

[5] N. Wirth. A plea for lean software. *Computer*, 28(2):64–68, February 1995. Conference Name: Computer. URL: https://ieeexplore.ieee.org/document/348001/?arnumber=348001, doi:10.1109/2.348001.