INTÉGRATION DE L'OUTIL D'ANALYSE *Mopsa* DANS *Visual Studio Code*

Gabriel Jeantet
n° étudiant: 28605630
Parcours STL, Master Informatique
Sorbonne Université
gabriel.jeantetpro@gmail.com

Mohamed Hernouf
n° étudiant: 3872505
Parcours STL, Master Informatique
Sorbonne Université
hernouf@yandex.ru

Claire Bardoux
n° étudiant: 28609617
Parcours STL, Master Informatique
Sorbonne Université
cbardoux26@gmail.com

Ce document présente la démarche suivie en cours afin d'effectuer nos recherches pour le projet d'*intégration de l'outil d'analyse Mopsa dans Visual Studio Code*.

1 Introduction

Des chercheurs du laboratoire *LIP6* developpent un analyseur statique et sémantique de programmes. Cet outil affiche ses résultats et s'utilise à partir d'un terminal. Il génére également un fichier résultat JSON. Visual Studio Code (VSCode) est un éditeur de code extensible. Il permet de s'intégrer facilement à des outils de build, d'analyse de code et de déboguage. Notre but est d'écrire une extension VSCode capable d'integrer Mopsa pour naviguer les résultats de l'analyse.

Il y a deux aspects principaux : l'exploitation du fichier JSON résultat et l'implantation du *Debug Adapter Protocol*. Dans la première partie, nous utilisons la liste et position des erreurs découvertes pour décorer le code sur lequel l'outil a été lancé. Ainsi l'utilisateur voit où il y a des erreurs et peut cliquer sur une variable et retrouver toutes les informations inférées par l'analyseur sur cette variable à ce point. Néanmoins le but est de mettre à jour l'interface au fur et à mesure de l'analyse et d'exécuter l'analyse pas à pas comme dans un débogueur. Pour cela nous devons dévéloper le *Debug Adapter Protocol* dans notre extension VSCode ainsi que dans leur outil.

2 Les mots clés retenus

Nous avons listé les mots utilisés lors de nos recherches bibliographiques et les avons organisées sous forme de carte heuristique.

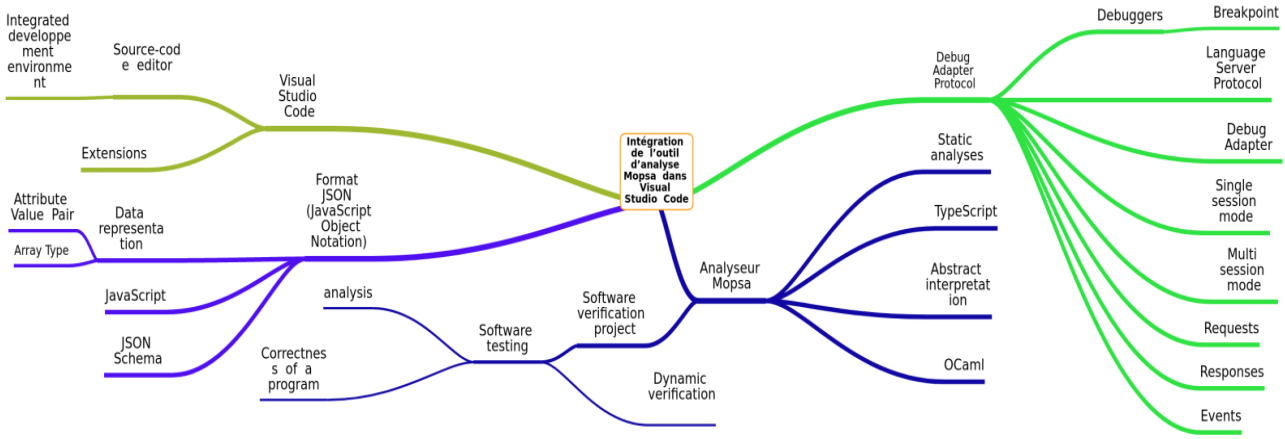

FIGURE 1 – Sample figure caption.

Voici la bibliographie :

— Debug Adapter Protocol
— Analyseur Mopsa [5][3][1][4]
— Format JSON [6],[2]
— Visual Studio Code

A noter que pour Debug Adapter Protocol et Visual Studio Code les informations sont extraite de sites web officiels mais pas sans être des documents, donc nous ne pouvons les citer dans le bibiliographie.

3 Descriptif de la recherche documentaire

Pour trouver des termes nous avons utilisé *le grand dictionnaire terminologique* et *termsciences*, qui permettent de nous guider sur les termes à privilégier et ceux à eviter. On a également utilisé la page de Wikipedia pour trouver des mots clés réliés ainsi que de l'information en faisant attention à ce qu'elle soit fiable.

Ensuite lorsqu'on avait un mot clé précis on effectuait des recherches d'articles scientifiques dans et les bases de données comme Google Scolar ou Web of Science. Google Scholar trouve beaucoup de résultats mais a très peu de possibilités de tri, tandis que Web of Science c'est tout le contraire. On utilisait également des bases de données spécialisés en informatique tels que ACM ou ArXiv. ArXiv a une publication très rapide et donc des données récentes, néanmoins ils sont si récents que certains documents ont pas eu le temps d'être relu et peuvent être faux. ACM, propose beaucoup d'outils de tri des articles et contrairement à ArXiv, les articles sont relu avant d'être plubliées, donc plus sûre.

Nous avons également lu beaucoup de documentation et d'API pour savoir comment écrire notre extension.

Finalement, grâce au catalogue de la bibliothèque universitaire nous avons pu rechercher les livres qui nous seraient utiles pour apprendre le langage de programmation dans lequel nous developpons.

4 Bibliographie produite dans le cadre du projet

Références

[1] David Delmas and Antoine Miné. Analysis of program differences with numerical abstract interpretation. page 6.

[2] David Flanagan. *JavaScript : The Definitive Guide*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 6th edition, 2011.

[3] Matthieu Journault and Antoine Miné. Static analysis by abstract interpretation of the functional correctness of matrix manipulating programs. In Xavier Rival, editor, *Static Analysis*, volume 9837, pages 257–277. Springer Berlin Heidelberg.

[4] Matthieu Journault, Antoine Miné, Raphaël Monat, and Abdelraouf Ouadjaout. Combinations of reusable abstract domains for a multilingual static analyzer. In Supratik Chakraborty and Jorge A. Navas, editors, *Verified Software. Theories, Tools, and Experiments*, volume 12031, pages 1–18. Springer International Publishing.

[5] Matthieu Journault, Antoine Miné, and Abdelraouf Ouadjaout. An abstract domain for trees with numeric relations. In Luís Caires, editor, *Programming Languages and Systems*, volume 11423, pages 724–751. Springer International Publishing.

[6] Geoff Langdale and Daniel Lemire. Parsing gigabytes of JSON per second. 28(6) :941–960.

5 Evaluation des sources

Fiabilité des sources		
Ressource	Manière dont on l'a trouvé	Evaluation critique
[6]	Via la base de données arXiv `https://arxiv.org/abs/1902.08318`	— Date/Fraicheur : L'article a été publié le 22 Février 2019 et jusqu'au deux janvier elle a eu des modification, ce qui prouve de sa fraicheur et véridicité. — Pertinence : Cette information nous a été utile afin de comprendre comment fonctionne le format JSON qui est décrite dans les premières pages. — Provenance : Bien que les articles de arXiv ne sont pas toujours vérifiés, nous pouvons étre sûr que les auteurs sont bien des personnes compétentes dans leurs domaines car nous pouvons retrouver leurs profils et d'autres articles dans le réseau web *ResearchGate* — Rigeur du contenu : La qualité de l'information était largement suffisante car nous avons acquis les compétences nécessaires afin d'exploiter sur les fichiers de format JSON. — Objectif : L'information a été écrite afin d'informer d'autres personnes une façon de parser un fichier JSON plus rapidement.
[5]	Via le site officiel de l'outil Mopsa `http://mopsa.lip6.fr/`	— Date/Fraicheur : Avril 2019, c'est la première publication de l'article. — Pertinence : L'information nous a été utile car elle nous a permis de comprendre mieux l'outil pour lequel nous dévélopons une extension. — Provenance : La provenance est vérifié avec sûreté étant donné que les auteurs sont nos tuteurs de projets et les dévéloppeurs de l'outil mopsa. — Rigeur du contenu : Les informations sont tout à fait vérifiable nottament car ils illustrent divers exemples de leur expériences dans leur article. — Objectif : Ce document a pour but d'informer les personnes sur leur outil.
[2]	via le catalogue de la bibliothèque	— Date/Fraicheur : Le livre a été publié en Mars 2011. — Pertinence : Cette information était très importante dans notre projet car nous avions peu de connaissance dans ce langage de programmation. — Provenance : L'auteur *David Flanagan* est un développeur et enseignant à *Massachusetts Institute of Technology*, donc la personne est bien qualifié pour parler de ce domaine. — Rigeur du contenu : Plus que suffisante. Les données sont sûres car c'est la sixième edition de ce livre et il est vendu partout dans le monde, et publié par une maison d'édition de renom. — Objectif : C'est un livre à caractère éducatif.