Classification Systématique sur l'Utilisabilité des APIs

Selsabil Dbouba, Khaled Fayala, Constantin Masson, Maxime Gallais-Jimenez
Département IRO
Université de Montréal, Montréal, Canada
{selsabil.dbouba, khaled.fayala, constantin.masson, maxime.gallais-jimenez}@umontreal.ca

***Résumé*—Les APIs (*Application Programing Interface*) sont considérées comme un des concepts les plus importants dans l'ingénierie logicielle moderne. Elles permettent aux développeurs de logiciels de minimiser l'impact causé par les changements dans la mise en œuvre des services logiciels et de travailler notamment d'une manière indépendante. Leur utilisabilité est un facteur important et a un impact prépondérant sur la qualité des logiciels. Dans cet article, nous avons réalisé une étude systématique de l'utilisabilité des APIs. Nous considérons 794 articles provenant de sources multiples, et nous nous concentrons plus précisément sur 57 d'entre eux. Cette étude nous aide à comprendre les tendances et les types d'artefacts qui ont été majoritairement utilisés pour l'amélioration de l'utilisabilité des APIs. Nos résultats incluent également une caractérisation et une analyse des propositions publiées suivant un schéma de classification rigoureux pour les articles retenus.**

***Mots clés*— *Application programming interface, API usability, API documentation, API usage, Systematic mapping*.**

I. INTRODUCTION

Une API est un ensemble de composants réutilisables tels que des objets, des routines ou des variables qui fournissent des fonctionnalités spécifiques [1], [2], [3]. Les APIs sont extrêmement populaires car elles favorisent la réutilisation des systèmes logiciels existants [4]. Les interfaces de programmation sont largement utilisées dans le développement de logiciels pour réutiliser les bibliothèques et les frameworks, et ainsi créer des applications riches avec diverses fonctionnalités [1], [5], [6]. Il existe plusieurs définitions pour le terme utilisabilité. La première définition [7] traite de l'utilisabilité des composants logiciels et correspond à la définition donnée par la norme ISO/CEI 9126 [8]. Elle définit l'utilisabilité comme : « *La capacité du composant logiciel à être compris, lorsqu'il est utilisé dans des conditions spécifiées* ». La seconde partie de la définition est particulièrement importante : un produit logiciel n'a pas de fonctionnalité intrinsèque, mais seulement une capacité à être utilisée dans un contexte donné (« contexte d'utilisation ») [9].

Dans cet article, nous présentons une étude systématique sur l'utilisabilité des APIs en suivant les étapes du processus proposé par Petersen et al. [10] qui définit la carte systématique comme une « *méthode pour construire un schéma de classification et structurer un domaine d'intérêt en ingénierie logicielle. Différentes facettes du schéma peuvent également être combinées pour répondre à des questions de recherche plus spécifiques.* »

Une cartographie systématique se concentre principalement sur la classification, l'analyse thématique et l'identification des domaines de publication [11]. Notre cartographie systématique se concentrera sur l'utilisabilité, afin d'identifier les tendances dans ce domaine.

La suite de cet article est organisée comme suit. La section II décrit la procédure adoptée pour notre étude. La section III présente les résultats de l'étape de sélection et la section IV les résultats de l'étude systématique, suivis par leur discussion à la section V. La section VI traite des limites de l'étude. La dernière section contient une conclusion avec un résumé des principales recommandations pour la recherche future sur l'utilisabilité des APIs.

II. PROCÉDURE

Dans cette section, nous discutons les principales étapes du processus de notre étude systématique, telle que définie par Petersen et al. [10]. Celles-ci incluent la définition de questions de recherche, la recherche de documents pertinents, le screening de documents, le mot-clé de résumés et l'extraction de données (voir la figure 1).

A. *Définition des questions de recherche*

Nous formulons donc nos objectifs de recherche avec les deux questions de recherche suivantes :

- **RQ1** : Quelles sont les tendances en termes d'utilisabilité d'API ?

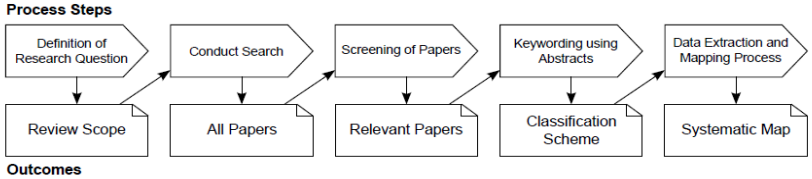

Figure 1 : Processus d'une étude systématique [10].

- **RQ2** : Quel type de processus et à quel moment est-il mis en place pour améliorer l'utilisabilité ?

B. Sélection de la source

À partir des questions de recherche, nous avons extrait des mots clés pour interroger Scopus[1]. Les principaux mots-clés qui ont été initialement identifiés sont les variations et les synonymes de deux mots « ***API*** » et « ***usability*** » qui sont : « *Application Programming Interface* », « *Software Library* » pour le terme API et « *ease-of-use* » pour l'utilisabilité. Enfin, pour construire la requête de recherche, nous avons essayé différentes combinaisons de mots clés, de variations et de synonymes avec des opérateurs logiques. La requête (sans filtre) utilisée est la suivante :

```
TITLE-ABS-KEY((((''API'')
    OR (''Software library'')
    OR (''Application Programming Interface'))
AND ((''usability'')
    OR (''ease?of?use''))
OR (''API us*'')
```

Avec la requête précédente, nous avons trouvé 794 documents.

C. Procédure de Screening

Le screening est la phase la plus cruciale d'une étude systématique [10]. Nous avons suivi la procédure de sélection suivante.

Les résultats de la recherche ont d'abord été filtrés pour éliminer automatiquement les enregistrements qui n'entraient pas dans le cadre de cette étude. Les documents qui ne sont pas dans le domaine du génie logiciel, non écrit en anglais ou en français. Le filtrage automatique, à travers l'ajout des filtres à la requête dans scopus, donne la requête suivante :

```
TITLE-ABS-KEY((((''API'')
    OR (''Software library'')
    OR (''Application Programming Interface'))
AND ((''usability'')
    OR (''ease?of?use''))
OR (''API us*'')
AND (LIMIT-TO(SUBJAREA, ''COMP''))
AND (LIMIT-TO(LANGUAGE, ''English'')
OR LIMIT-TO (LANGUAGE, ''French'')
```

Ce filtre automatique nous permet de réduire le nombre de documents trouvés à 516.

Afin d'éviter l'exclusion des documents qui devraient faire partie du corpus final, nous avons suivi une procédure de screening stricte. Avec quatre examinateurs à notre disposition (co-auteurs de ce document), chaque article est examiné par deux examinateurs indépendamment. Lorsque les deux examinateurs d'un article ne sont pas d'accord sur l'inclusion ou l'exclusion d'un document, une discussion est nécessaire par tous les membres jusqu'à ce qu'un consensus soit atteint. Pour déterminer un processus d'inclusion/exclusion équitable, chaque évaluateur examine un échantillon d'au moins 50% des articles étudiés, pour s'assurer qu'aucun document ne soit manqué.

Critères d'Inclusion

Un article est inclus si son titre ou son résumé mentionne explicitement l'utilisabilité des APIs dans son contexte particulier. De plus cet article doit montrer un potentiel pour répondre à au moins une question de recherche.

Critères d'Exclusion

Comme mentionné précédemment, chaque document a été examiné par deux examinateurs pour décider de l'inclure ou de l'exclure en utilisant seulement des informations contenant son titre et son résumé. Nous avons aussi utilisé les quatre critères d'exclusion suivants :

1. Le papier n'est pas dans le domaine du génie logiciel.
2. Le document ne traite pas l'utilisabilité des APIs.
3. Ce n'est pas un article complet d'une conférence ou d'un journal.
4. La contribution principale du document n'est pas sur l'utilisabilité de l'API.

D. Schéma de classification

Dans notre étude, nous avons utilisé une approche hybride afin de déterminer le schéma de classification retenu pour l'analyse. Pour commencer, nous avons appliqué la stratégie de haut vers le bas en proposant une première version de notre schéma sur la base des titres, les résumés des articles ainsi que nos connaissances générales du domaine. Ensuite, nous avons amélioré notre schéma en suivant la stratégie de bas vers le haut qui consiste à extraire le schéma de classification en lisant les papiers conservés et en déterminant les propriétés importantes de classification. Nous avons conservé quatre propriétés de classification :

- **Type d'étude**

Cela détermine le type d'étude utilisée : *revue de littérature*, *étude de cas*, *étude utilisateur*, ou *autre*.

- **Contribution**

Cela spécifie le type de contribution utilisé dans l'article : *détection d'artefact*, *mesure*, *développement d'outil* ou *autre*.

Ainsi, nous pouvons spécifier le type de **détection d'artefact** étudié dans l'article : *API usage*, *API documentation*, *Test cases*, *code completion*, etc.

[1] Base de données scientifique interdisciplinaire contenant plus de 60 millions enregistrements, www.elsevier.com/online-tools/scopus.

- **Source de données**

Cela caractérise la source des données visualisées : *données industrielles, open source, données hybrides, réponse d'une enquête* ou *absence de données*.

- **Stade de contribution**

Avant développement, *pendant développement* ou *après développement*.

E. Extraction de données et carte systématique

Au cours de cette phase est réalisé le classement réel des articles pertinents par des catégories du schéma de classification. Les examinateurs effectuent une analyse détaillée de chaque article retenu pour l'étude et déterminent les classes appropriées pour chacun d'entre eux. La section suivante présente les résultats concernant les publications examinées dans cette étude.

III. RÉSULTATS DE PROCESSUS DE SÉLECTION

Dans cette section, nous décrivons les étapes réalisées lors de la sélection des articles et les résultats de sélection.

A. Outil de sélection

Pour réaliser efficacement l'examen de centaines d'articles, nous avons utilisé un outil fourni par l'université. Cet outil offre trois fonctionnalités : *mise en place*, *filtrage* et *finalisation*.

- ***Fonctionnalité de mise en place***

Cette fonctionnalité permet l'analyse des informations fournies par le moteur de recherche scopus qui exporte les données en format .csv. L'outil affecte automatiquement les articles aux examinateurs en produisant, pour chacun d'eux, un fichier ayant l'extension .todo. Notons que chaque article doit être examiné par deux examinateurs. Il y a donc une répartition aléatoire et équitable des papiers.

- ***Fonctionnalité de filtrage***.

À l'aide de cette fonction, chaque examinateur exécute son fichier todo. L'article est présenté dans une simple fenêtre contenant les auteurs, le titre, le résumé dans une fenêtre de texte défilant (nommée tk), une liste de contrôles des raisons de rejet et des boutons pour passer à l'article suivant ou pour enregistrer une session, comme le montre la Figure 2.

Après le passage en revue de chaque article, la décision est enregistrée dans un fichier de sortie ayant l'extension .out et l'article suivant est affiché. Une session de présélection pourrait être interrompue (option de sauvegarde) et poursuivie une autre fois. Lorsque tous les fichiers todo sont sélectionnés, nous obtenons un fichier de décision par examinateur.

- ***Fonctionnalité de finalisation*.**

Quatre examinateurs ont été impliqués dans le processus de sélection, et chacun des documents a été doublement évalué par deux examinateurs.

Dans cette fonctionnalité, les fichiers de décision des quatre examinateurs sont fusionnés. Les articles sont classés dans trois catégories :

- ✓ **Accepté**

Lorsque les deux examinateurs s'entendent sur l'acceptation.

- ✓ **Rejeté**

Lorsque les deux sont d'accord sur le rejet de l'article.

- ✓ **Conflit**

Quand l'un accepte et l'autre rejette l'article. Afin de résoudre les conflits rencontrés lors de l'intégration, des rencontres en face à face ont été organisés entre les quatre examinateurs pour parvenir à un accord.

B. Résultats de sélection

Dans cette partie, nous présentons le flux d'informations à travers les différentes phases de l'étude [12]. A chaque phase, le nombre d'articles est réduit en éliminant ceux qui ne correspondent pas à notre étude systématique.

Dans les paragraphes qui suivent, nous expliquons les passages de flux d'informations :

- **Identification**

Au total, la recherche a produit 794 documents candidats. Tous les documents retournés proviennent de la base de données scientifique Scopus. Une grande partie de ces articles sont publiés dans des conférences et revues sponsorisées par IEEE.

- **Screening**

Dans cette phase, nous avons examiné 516 articles au lieu 794 puisque nous avons appliqué des filtres automatiques dans la requête (voir *procédure de screening* dans la section II). Les résultats que nous avons obtenus dans cette phase sont les suivants :

- ✓ Articles **rejetés** : 369
- ✓ Articles **ambigus** : 91
- ✓ Articles **acceptés** : 56

- **Admissibilité**

Après la phase de Screening et donc le traitement des articles ambigus, 73 articles ont été acceptés pour l'analyse. Ce qui représente 14% des documents. C'est le taux d'acceptation habituel dans les études systématiques de littérature et de cartographie [13].

Les 721 autres documents ont été rejetés pour l'un des quatre critères d'exclusions indiqués précédemment. La principale cause de rejet au cours du Screening était que les titres et/ou les résumés qui contenaient les mots clés recherchés ne proposaient pas de contribution spécifique pour notre étude.

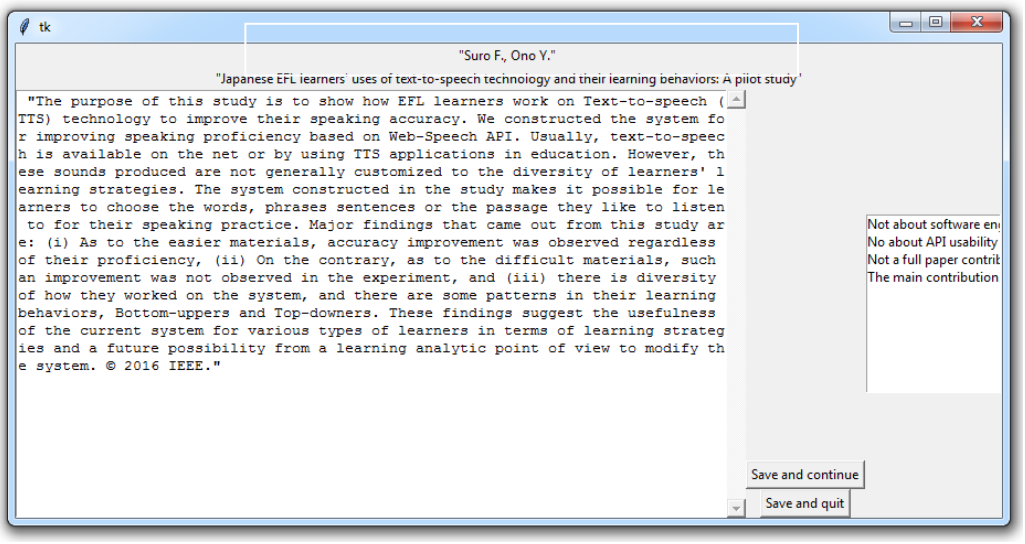

Figure 2. Interface de l'outil de screening

Dans la phase d'admissibilité, nous avons aussi rejeté 16 autres documents après la lecture complète pour une des raisons d'exclusion décrite précédemment, ou parce que l'article contient moins de 4 pages au total ou ne contribue pas à notre étude. Cette étude porte finalement sur 57 articles.

- **Validation**

Nous avons considéré à ce niveau 57 articles au total qui proposent des contributions en rapport avec l'utilisabilité des APIs.

Donc grâce aux différentes étapes de processus de sélection expliquées dans cette section et à l'utilisation des critères d'Inclusion/Exclusion, nous sommes passés de 794 articles identifiés au départ à seulement 57 articles correspondant à notre étude.

Dans la section suivante, nous analysons les documents conservés selon le schéma de classification décrit précédemment afin de répondre aux questions de recherche énoncées à la section II.

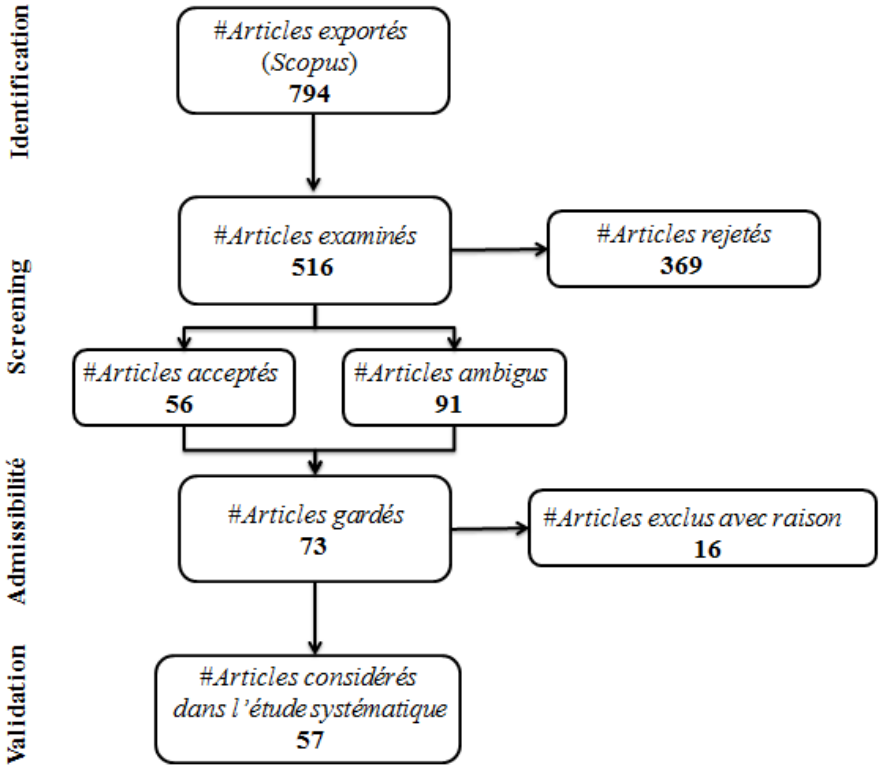

Figure 3. Flux d'informations grâce à notre processus de cartographie systématique **[12]**.

IV. RESULTATS DE L'ETUDE

Comme précisé plus tôt notre étude porte sur 57 articles car 16 ont été exclus après lecture en raison de leur manque de pages ou de leur éloignement total du sujet. Les articles ont été classés selon le schéma de classification présenté précédemment (Section II D). Cette section a pour objectif de présenter les résultats découlant de cette classification afin de pouvoir répondre aux questions de recherche posées en préambule de cet article (Section II).

A. *Type d'étude*

Nous avons considéré 4 types d'étude.

1) L'étude de cas : Ce type d'étude a pour but d'étudier un cas précis afin de récolter des détails précis. Dans notre cas une grande partie des papiers étudiés sont des études de cas (42% Figure 4). Ceci peut s'expliquer d'une part par le fait qu'il y a une grande diversité d'API, ce qui oblige à se concentrer sur un seul cas et d'autre part que l'utilisabilité est un critère plutôt spécifique au cas étudié. Le type de l'API va jouer sur les critères d'utilisabilité. Par exemple l'article de Sohan et al. *[14]* s'attache spécifiquement à l'étude des APIs web afin d'y suivre leur évolution au cours du temps et plus précisément celle de leur documentation. Le but étant de voir si celle-ci est mise à jour en même temps que sa documentation. De plus ils inspectent la manière dont les informations relatives aux mises à jour sont transmises au client. La spécificité de l'utilisabilité empêche l'étude d'un grand nombre d'API simulatanément. Une étude de cas typique est celle menée par Gerken et al. *[15]* qui s'intéresse à l'utilité d'avoir des feedbacks le plus tôt possible lors du développement d'une API afin de détecter des erreurs de conception avant la phase de réalisation. On remarque donc l'importance de s'attacher à un cas particulier pour obtenir des résultats exploitables et généralisables. Cela est signe d'un domaine en développement comme nous le montrerons plus tard (Section V).

2) L'étude utilisateur : Comme son nom l'indique elle est centrée sur l'utilisateur, il est l'objet de l'étude, tout tourne autour de lui. Dans le contexte de l'utilisabilité ceci est primordial. C'est pour cela que nous avons 37% (Figure 4) des études de notre classification systématique qui se consacrent à l'utilisateur. Tout cela sans compter les études de cas où la validation est effectuée par une étude utilisateur. La particularité de l'utilisabilité est que c'est un facteur assez subjectif. Cependant lorsque l'on mène ce type d'étude on peut y apercevoir des tendances. Comme le montre Stylos et al. *[16]* où sur l'étude d'une API donnée ils ont demandé aux utilisateurs de réaliser une tâche avec celle-ci et de réflechir à haute voix. De cette expérience ils se sont aperçus que les utilisateurs pensaient avec un meilleur niveau que les méthodes proposée par l'API et ils ont détecté une faille d'utilisabilité. Piccioni et al. *[17]* s'attachent à la dimension cognitive de l'utilisabilité. Cette dimension cognitive ne peut être perçue que par le biais d'une étude utilisateur car nous ne sommes pas capable de simuler le ressenti humain. Ils ont suivi le guide de la dimension cognitive de Clarke qui fait référence sur l'utilisabilité des APIs *[18]*. L'étude utilisateur est aussi importante car elle permet de mettre en évidence la partie compréhension de l'API. Ko et al. *[19]* expriment le besoin de l'utilisateur d'avoir des connaissances en photographie pour pouvor utiliser une API de caméra. Tous ces critères ne sont constatables que par le biais de l'étude utilisateur.

3) Revue de littérature : Ce type d'étude permet de passer en revue tout ce qui peut s'écrire dans le domaine. Ce sont des sources rares, 5% de nos articles passés en revue (Figure 4), mais pas pour le moins importantes. Stylos et al. *[20]* passent en revue les différents attributs de qualité d'une API afin proposer une méthode d'amélioration de l'utilisabilité des APIs. Il ressort de cet article que l'utilisabilité se définit bien comme la facilité d'utilisation et conseille d'adopter un point de vu plus humain en fournissant une documentation adaptée. L'autre revue de Stylos et al. *[21]* s'attache plus au côté facilité d'apprentissage de l'API et l'effort à fournir pour le déboggage du code. Il résulte de cet article qu'il faut améliorer différents aspects de l'API qui sont la documentation, fournir plus d'exemples et voir même changer d'outil de développement.

4) Autre : Ce sont tous les cas qui n'entrent pas les types d'études précédemment évoqués. Par exemple nous avons Xie et al. *[22]* ont mis au point un framework d'exploitation de l'utilisation de l'API et son outil de support MAPO. Le but de cet outil étant d'extraire les différents types d'utilisation d'API afin que les développeurs puissent les analyser. Autre exemple de cette pluralité en termes de type d'étude, l'article de Lee et al. *[23]* qui présente un processus entier de développement d'API. Ces articles permettent d'apporter une approche différente sur l'utilisabilité.

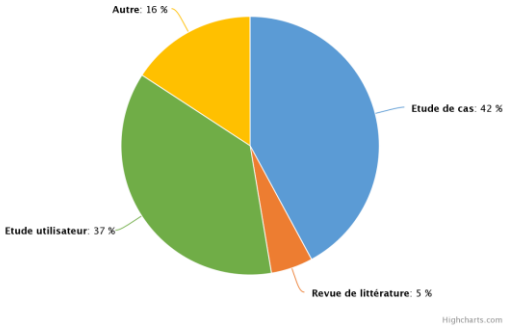

Figure 4 : Types d'études

B. *Contribution*

Nous avons étudié 4 types de contribution différents.

1) Détection d'artéfacts : Celle-ci consiste à analyser et comprendre un élément précis de l'utilisabilité. Il y a différentts types d'artéfacts qui seront discutés plus tard (Section IV D). Ce type de contribution est la plus importante (61%, Figure 5). Ceci s'explique par le fait qu'il est plus simple et plus concret d'étudier un seul aspect de l'utilisabilité. Ceci permet d'avoir des résultats plus précis qui auront pour but d'être généralisés.

2) Mesure d'API : Cette contribution a pour but de mesurer de façon qualitative et/ou quantitative l''utilisabilité d'une API. Rama et al. *[24]* donnent de nouvelles mesures d'utilisabilité à

partir de plusieurs éléments constituants des problèmes d'utilisabilité d'API. Comme par exemple le trop grand nombre de paramètres donnés à une méthode. Stylos et al. *[25]* ont développé une nouvelle méthode afin de mesurer, durant le dévelopement, l'utilisabilité d'une API. La mesure peut aussi s'exprimer par le prisme de la description d'un processus de développement d'API comme le montre Lee et al. *[23]*. Les auteurs recommendent un processus entier, en 4 étapes, de développement, qui leur permet d'obtenir des résultats en terme d'utilisabilité légèrement supérieurs à la moyenne. Mais la mesure peut aussi s'exprimer grâce à l'étude utilisateur exprimée précédemment (Section IV A), Piccioni et al. *[17]* utilisent leur approche cognitive afin de mesurer différents critères d'utilisabilité. Donc la mesure d'API est une contribution importante demandant un travail complet. Ce qui explique qu'elle représente 30% des articles étudiés (Figure 5).

3) Développement d'outil : Les outils sont un bon moyen d'appuyer une approche. Leur développement est plus long mais permet d'avoir un résultat plus parlant. C'est pour cette raison qu'ils ne représentent que 5% des contributions récoltées (Figure 5). Wu et al. *[26]* ont par exemple présenté ACUA qui est un outil permettant de suivre les changements apportés à une API, et notifier quels impacts ces changements pourraient avoir sur les clients de l'API. ACUA permet aussi de prévenir le client des changements intervenus sur l'API afin qu'il puisse adapter son programme en conséquence. Le développement d'outil permet de mettre en pratique les différents sujets d'études et ajoute une vision pratique à toutes ces démarches.

4) Autre : Dans cette section ont été classés les articles n'ayant pas de réelle contribution pour l'étude mais qui comportaient des détails intéressants pouvant faire référence. Avec dans un premier temps l'article de McLellan et al. *[27]* qui observe que le code bien commenté et fondé pourrait aider l'apprentissage des entrées et des sorties de nouveaux appels API et leurs relations dans le contexte. Secondement Suchine et al. *[28]* montrent que les développeurs de tout niveau rencontrent des problèmes similaires concernant les protocoles des APIs.

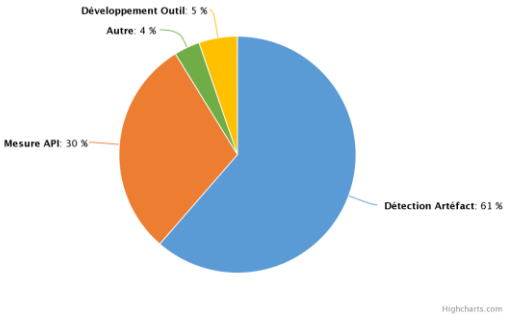

Figure 5 : Types de contribution

C. *Source de données*

Nous nous sommes intéressés aux différents types de données utilisées. Parmi nos résultats ces sont les données qui ont le moins contribué à notre étude. Nous avons classifié nos articles selon 5 types de source : *données open source, données industrielles, données hybrides, données d'enquête et aucune donnée.* Nous avons pu constater que la majeure partie des données d'étude provenaient de domaines open source ce qui s'explique par leur facilité d'accès. Une autre partie importante des données provient des enquêtes menées. Ceci correspond à la grande proportion d'étude utilisateur. Les autres types de données ont été beaucoup moins observés.

D. *Stade de contribution*

Nous avons aussi regardé à quel moment du développement intervenait chaque article.

1) Avant le développement : Comme indiqué il y a des contributions qui portent à anticiper les critères d'utilisabilité avant même de commencer le développement de l'API. Ces critères sont conjoints avec ceux pendant le développement. Le but est de penser prématurément aux erreurs qui pourraient survenir. Cependant les exemples de contributions avant le développement restent minoritaires à seulement 12% (Figure 6)

2) Pendant le développement : D'autres contributions vont vouloir aider le développeur lors de la mise en place de son API. Lorsqu'il est dans le processus de développement, avec toutes les fonctionnalités possibles en tête et tous les détails de conception. Il peut se servir de ce moment pour améliorer l'utilisabilité. Stylos et Clarke ont montré dans leur étude que lors du développement de l'API, proposer des constructeurs avec des paramètres ne diminue pas les erreurs d'utilisation *[29]*. Ces études sont elles aussi minoritaires à seulement 17% (Figure 6).

3) Après le développement : La majorité (71%, Figure 6) des contributions sont applicables après le développement. Ceci est expliqué par le fait que la majeure partie des APIs sont déjà en place et qu'on ne va pas proposer de solution qui consiste à tout développer à nouveau. De plus le but est d'améliorer l'utilisabilité. Il est difficile de deviner si elle sera bonne ou non avant le développement. Une grande partie des approches consiste à regarder ce qu'en pense les utilisateurs. Par exemple Wang et al. *[30]* recommande de prétaiter attentions à la rétrocompatibilit des changement apportés au cours de la maintenance de l'API.

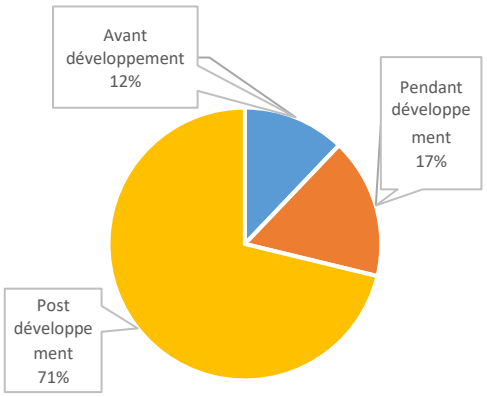

Figure 6 : Stade de contribution

Il existe aussi des contributions qui prennent effet à plusieurs stades du développement. Comme par exemple les conseils de Lee et al. [23] avec leurs 4 étapes de développement qui s'appliquent donc avant, pendant et après le développement. Un exemple de conseil avancé est de commencer la documentation avant le développement. On remarque que l'utilisabilité peut donc être améliorée tout au long du processus de développement, mais que la majeure partie des solutions sont proposées après le développement.

E. Les artéfacts

De nombreux artéfacts ont été détectés lors de cette étude (Figure 7). La multitude d'artéfacts est dû au fait que certains d'entre eux se recoupent les uns les autres. Il n'y a pas la présence d'un artéfact qui domine tous les autres, cependant on peut noter qu'il y en a 3 majoritaires qui sont la documentation (22%), les exemples (17%) et l'utilisation d'API (17%).

Premièrement, la documentation est fortement présente car c'est le premier endroit où le développeur se rend. Il y a accès lorsqu'il a besoin de détails sur l'utilisation de l'API. L'API atteignant l'utilisabilité ultime n'aurait pas besoin de documentation, mais nous y sommes encore loin. D'après Zibran et al. [31], c'est même le premier facteur d'utilisabilité. Alors que Petersen et al. [32] considèrent l'hypothèse comme quoi elle est loin d'être indispensable. Cela reste une pensée à part. La documentation permet de faire le lien entre le développeur et l'utilisateur. Elle permet la compréhension de l'API, c'est un mode d'emploi, on comprend donc l'importance de son étude.

Ensuite viennent les exemples d'utilisation. Ils sont très souvent liés à la documentation. L'exemple permet de mettre en contexte l'explication fournie pour la documentation. Ils sont très appréciés des utilisateurs mais il est difficile pour les développeurs de générer des exemples répondant à tous les contextes d'utilisation. C'est pour cela que Bue et al. [33] ont mis au point un algorithme pour générer automatiquement des exemples à partir de code source. Méthode plus inattendue proposée par Nasehi et al. [34] qui ont généré des exemples d'utilisation grâce à des tests unitaires. Les exemples sont eux aussi très importants et les utilisateurs apprécient les exemples commentés avec une description en langage naturel, c'est ce qu'a révélé l'étude de Wang et al. [35].

Dans notre étude nous retrouvons une bonne proportion de recherche d'utilisation d'API. L'objectif de cet artéfact est de mieux comprendre l'utilisation faite des APIs. Comprendre l'utilisation permet de mieux étudier l'utilisabilité. Etudier l'utilisation permet aussi de détecter les utilisations suspectes [36]. L'étude de l'utilisation est donc intéressante elle permet la bonne évolution de l'API.

Pour poursuivre nous constatons un attrait non négligeable à la complétion du code. Celle-ci peut permettre d'améliorer la rapidité de développement en rappelant les paramètres ou le nom des méthodes. Elle peut aussi aider à apprendre à utiliser l'API. Car en plus des exemples cités précédemment elle peut fournir des suggestions de méthodes pertinentes ou juste nous relayer la documentation [37]. C'est un secteur étudié ou de nouvelles fonctionnalités de regroupement, tri et filtrages voient le jour [5]. Une technique basée sur l'exemple, utilisant les informations contextuelles a été mise au point pour de meilleur résultats [36]. C'est donc un outil qui avec le temps peut s'avérer indispensable.

Comme évoqué précédemment, les différents artéfacts s'entrecoupent, avec par exemple une étude des contraintes d'utilisation et leur documentation menée par Sahraoui et al. [38]. On peut noter que le type d'artéfact permet d'axer la recherche sur un point particulier mais ils ont tendance à avoir des points communs et apportent chacun de l'aide pour améliorer l'utilisabilité.

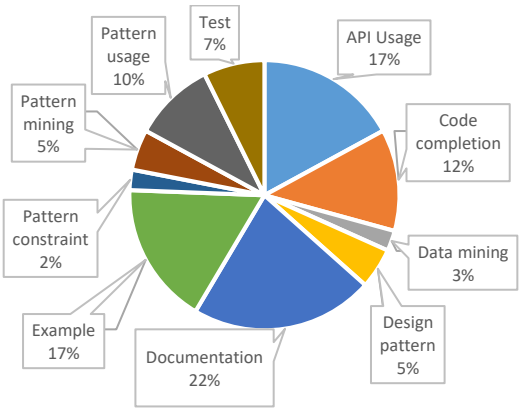

Figure 7 : Types de d'artéfacts

V. DISCUSSION

Dans cette section nous allons tacher de répondre à nos deux questions de recherche énoncées au début de l'article en utilisant les résultats obtenus de notre schéma de classification et du traitement des données concernant les articles que nous avons grâce à scopus.

Dans un premier temps rappelons nos questions de recherches :

- **RQ1** : Quelles sont les tendances en termes d'utilisabilité d'API ?
- **RQ2** : Quel type de processus et à quel moment est-il mis en place pour améliorer l'utilisabilité ?

A. Les tendances

Nous avons pu observer que tous nos articles à l'exception d'un seul sont datés du XXI^{ème} siècle et qu'il y a une tendance de plus en plus forte concernant l'utilisabilité (Figure 8). Ce phénomène s'explique par l'inexorable croissance du nombre d'APIs, mais aussi par la mise à jour des APIs existantes et par l'expansion du nombre de sites permettant de répondre aux questions en tout genre sur les APIs. C'est pour cela que la question de l'utilisabilité des APIs se pose de plus en plus. Car les développeurs veulent avoir une API populaire et agréable à l'utilisation.

Ensuite nous avons pu observer que les études réalisées étaient majoritairement des études de cas, et des études utilisateur. Ceci s'explique par le fait que l'utilisabilité a des caractéristiques précises dépendantes grandement du ressenti des utilisateurs. De plus les critères d'utilisabilité diffèrent en fonction du type d'API, ce qui explique la grande proportion d'étude de cas. Ce qui a poussé les études à se concentrer sur

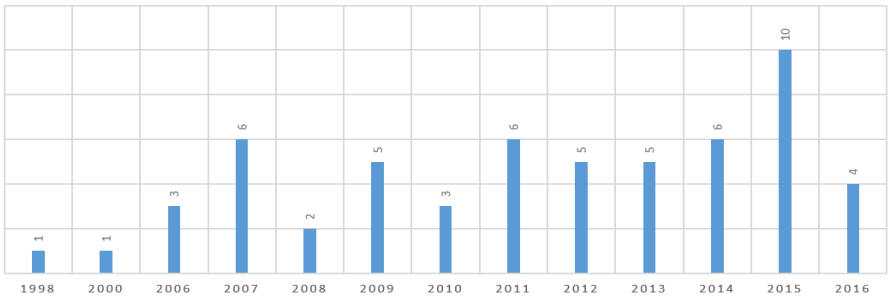

Figure 7 : Nombre d'articles par année

un seul cas. De plus les études de cas sont en bonne partie validées par des études utilisateurs. Ce qui montre le lien fort entre les deux et la tendance actuelle d'appliquer un principe sur une API particulière ou un type d'API et de vérifier les résultats auprès des utilisateurs. Pour apporter une pierre à l'édifice de cette recherche détaillée nous avons constaté qu'une part prépondérante des contributions concernaient la détection d'artéfact. Donc en plus de limiter le champ de l'API dans l'étude on se concentre sur un point particulier de celle-ci pour tenter d'améliorer les travaux actuels.

Pour conclure sur les tendances concernant l'utilisabilité. Nous avons pu constater l'intérêt grandissant dans ce domaine et que des études de grande envergure ne devraient pas tarder à être lancées. Pour le moment le domaine se limite à des études de cas qui se restreignent à un petit ensemble d'APIs. Dans une majeure partie des cas elles se concentrent sur un artéfact, et comportent la majorité du temps une étude utilisateur. Les mots d'ordres sont donc précision et utilisateur.

B. Les processus d'amélioration

En ce qui concerne l'amélioration nous pouvons nous tourner dans un premier temps vers les différents artéfacts étudiés. Pour débuter il faut se tourner vers la documentation qui doit être réalisée le plus tôt possible [31]. Lors de la réalisation de cette dernière il faut faire attention de préciser les contraintes des paramètres de chaque méthode [38]. Ensuite lors de l'implémentation il est important de privilégier des constructeurs sans paramètre (par défaut) [5], [29], et des méthodes considérées haut niveau (qui permettent à l'utilisateur de réaliser une action avec le moins d'effort d'écriture et réflexion possible) [16]. Ces méthodes doivent avoir des préfixes cohérents et des noms explicites [27]. De plus elles doivent au maximum éviter de se surcharger [27]. Concernant le code en lui-même il ne doit pas comporter plus de classe que nécessaire et les packages doivent être de taille raisonnable pour que l'utilisateur puisse s'y retrouver [39]. Par la suite tout un processus de tests est à réaliser. Ce qui permettra de commencer à réaliser des exemples qui pourront être intégrés à la documentation [34]. Les développeurs doivent restés alertes aux sites de question et réponses (Q&A) qui seront leur moyen de dialogue avec les utilisateurs et une façon de récupérer des avis sur le travail effectué [35]. De plus ils doivent mettre à jour la documentation au fur et mesure tout en alertant les utilisateurs, et continuer par conséquent le travail d'illustration par l'exemple. Comme à dit Einstein : « L'exemple n'est pas un autre moyen d'enseigner, c'est le seul. » [35]. C'est pour cela que les exemples se devront d'être commentés et comporter une description en langage naturel, très appréciée des utilisateurs [35]. Après le lancement de l'API il est important de recueillir les feedbacks et le type d'utilisation qui est fait de l'API afin de pouvoir étudier le comportement des utilisateurs et à terme combler les lacunes et améliorer les forces de celle-ci. Autre tâche qu'il est possible de réaliser après le développement de l'API est une étude empirique de l'utilisabilité mais ceci est couteux en termes de moyens et surtout de temps. Une des alternatives proposée est « l'API Peer Review » de Farooq et al. [40] qui permet soit un gain de temps par rapport l'étude soit de compléter l'étude empirique et par conséquent de détecter toutes les faiblesses d'utilisabilité. Toutes ces études poussent le développeur à faire des changements. Il doit donc être vigilant à la rétrocompatibilité de ceux-ci [30].

Comme nous pouvons le voir l'amélioration de l'utilisabilité se fait tout au long du processus de développement. De nombreux moyens sont à la disposition de l'utilisateur afin d'aider les utilisateurs à apprendre, comprendre et utiliser son travail. Nous voyons aussi qu'il existe une partie maintenabilité à ne pas négliger.

VI. MENACES A LA VALIDITE

Plusieurs menaces à la validité sont possibles dans ce rapport. La première est que chaque article a été lu en détail par un seul examinateur. Ce qui confère des avantages et des inconvénients lors de la mise en commun du travail. Chacun à des références différentes, ce qui est intéressant mais il n'est pas chose aisée de toutes les intégrer au bon moment. Ce problème se retrouve aussi lors de la classification où chaque examinateur a classifié les articles qu'il a lu, ce qui est une très bonne façon de faire. Cependant, malgré l'explication du schéma de classification, chacun à son interprétation de celui-ci. Un travail de regroupement des catégories a été fait sur les articles après la classification, et le résultat des statistiques s'en ressent car n'ayant pas les mêmes mots nous avions les mêmes idées et les résultats des études s'en ressentent.

La plupart de nos menaces viennent de la contrainte de temps pour réaliser l'exercice. Il a été demandé de réaliser cette classification systématique en quelques semaines et cela dans le cadre d'un cours. De plus nous réalisons ici, notre première classification systématique. C'est aussi la première collaboration entre tous les membres de l'article. Tout cela ne nous a pas empêché d'être le plus rigoureux et enthousiaste possible dans notre travail afin de produire les meilleurs résultats possibles.

Une autre menace que nous pourrions soulever est à propos du sujet. D'une part le terme API qui de nos jours est très vaste. Cela a pour avantage qu'il y a une richesse dans la littérature à étudier mais en contrepartie il peut y avoir des articles très éloignés de nos attentes. Cependant nous avons essayé d'atténuer cette contrainte au maximum lors de l'étape de screening réalisée plus tôt (Section III B). D'autre part nous avons le terme utilisabilité qui d'après nos recherches désigne la facilité d'utilisation. Mais ce terme est aussi très proche de l'utilisation de l'API comme nous avons pu le constater lors de cette étude. De surcroit les articles étudiés sont récents. Ce domaine est de nos jours de plus en plus étudié. Par conséquent notre revue systématique pourrait être prématurément obsolète et avoir besoin d'une mise à jour.

Une légère menace que nous pourrions ajouter serait à propos du regroupement des artéfacts. Plusieurs types d'artéfacts se recoupent et nous disposons d'un grand nombre de types. C'est pour cela que nous aurions pu les regrouper par catégories afin d'avoir une meilleure généralisation.

La dernière menace éventuelle pourrait être le fait que nous sommes des étudiants en maîtrise et ne sommes pas spécialistes des APIs et c'est pour cela que nous nous sommes concentrés sur les articles recensés lors notre requête sur scopus et n'avons pu apporter plus de références qui sont déjà nombreuses et d'après nos lectures complètes. Nous pouvons ajouter à tout cela la source d'extraction de nos articles. Il s'agit uniquement de scopus. Bien que cette base de données fasse référence par sa richesse il est possible que nous ayons manqué des articles.

Nous retiendrons des menaces présentes qu'elles sont pour la plupart liées à la contrainte de temps que nous avions pour réaliser l'exercice de cette classification systématique et dans un second temps à notre manque d'expérience en la matière mais ceci est compensé par notre envie de bien faire.

VII. Conclusion

Dans cette revue systématique nous avons montré que la tendance à étudier l'utilisabilité était en progression au cours des dernières années. La majorité des études sont des études de cas se faisant sur un artéfact particulier de l'API. Et la validation ou l'étude met à contribution l'utilisateur afin de recueillir son ressenti. Nous avons pu considérer plusieurs méthodes afin d'améliorer l'utilisabilité. Ces méthodes sont applicables avant, pendant et après le développement. En s'attachant à différents artéfacts lors du développement l'utilisabilité peut être accrue. Les principaux critères retenus sont la documentation, les exemples et l'utilisation qui est faite de l'API. En combinant les différentes contributions on peut surement améliorer l'utilisabilité, mais ce point est à étudier. Le domaine manque aussi d'études de très grandes envergures. Les contributions étant de plus en plus nombreuses il sera intéressant de combiner un maximum d'artéfacts et toujours en validant les résultats sur un ensemble d'utilisateur.

VIII. References

[1] B. Foote et R. E. Johnson, «Designing reusable classes,» *Journal of object-oriented programming*, pp. 22-35, 1988.

[2] Y. B. Chhetri et M. P. Robillard, «Recommending reference API documentation,» p. 1–29, 2014.

[3] J. Stylos et B. Myers, «The implications of method placement on API learnability,» *Proceedings of the 16th ACM SIGSOFT international symposium on the foundations of software engineering*, p. 105–112, 2008.

[4] T. Grill, O. Polacek et M. Tscheligi, «Methods towards API usability: A structural analysis of usability problem categories,» *Proceedings of the Fourth International Conference on HumanCentered Software Engineering*, p. 164–180, 2012.

[5] C. Zhang, J. Yang, Y. Zhang, J. Fan, X. Zhang, J. Zhao et P. Ou, «Automatic parameter recommendation for practical API usage,» *ICSE*, p. 826–836, 2012.

[6] A. Bacchelli et A. Sawant, «A Dataset for API Usage,» *Proc. 12th Working Conference on Mining Software Repositories (MSR)*, pp. 506-509, 2015.

[7] M. F. Bertoa, J. M. Troya et A. Vallecillo, «Measuring the usability of software components,» vol. 79, pp. 427-439, 2006.

[8] ISO/IEC, Software engineering - Product quality, 2001.

[9] E. Kühn et T. Scheller, «Automated measurement of API usability: The API Concepts Framework,» *Information and Software Technology*, vol. 61, pp. 145-162, 2015.

[10] K. Petersen, R. Feldt, S. Mujtaba et M. Mattsson, «Systematic Mapping Studies in Software Engineering,» *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, p. 68–77, 2008.

[11] B. Edouard, S. Houari, S. Eugène, M. Paul et S. Wael, «Systematic mapping study of model transformations for concrete problems,» *Model-driven engineering and software development*, pp. 176-183, 2007.

[12] D. Moher, A. Liberati, J. Tetzlaff et D. Altman, «Preferred reportingitems for systematic reviews and meta-analyses: the prisma statement,» *Annals of Internal Medicine*, vol. 151, n° %15, p. 264–269, 2009.

[13] A. Seriai, O. Benomar, B. Cerat et H. Sahraoui, «Validation of Software Visualization Tools: A Systematic Mapping Study,» *Working Conference on Software Visualization*, pp. 60-69, 2014.

[14] S. M. Sohan, C. Anslow et F. Maurer, «A Case Study of Web API Evolution,» *IEEE World Congress on Services, SERVICES 2015*, pp. 245-252, 2015.

[15] J. Gerken, H.-C. Jetter, M. Zöllner, M. Mader et H. Reiterer, «The concept maps method as a tool to evaluate the usability of APIs,» *29th Annual CHI Conference on Human Factors in Computing Systems*, pp. 3373-3382, 2011.

[16] J. Stylos, B. Graf, D. Busse, C. Ziegler, R. Ehret et J. Karstens, «A case study of API redesign for improved usability,» 2008.

[17] M. Piccioni, C. Furia et B. Meyer, «An empirical study of API usability,» *IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 5-14, 2013.

[18] S. Clarke, «Measuring API usability,» 2004.

[19] A. Ko et Y. Riche, «The role of conceptual knowledge in API usability,» *IEEE Symposium on Visual Languages and Human Centric Computing*, pp. 173-176, 2011.

[20] J. Stylos et B. Myers, «Improving API usability,» *Communications of the ACM*, vol. 59, pp. 62-69, 2016.

[21] J. Stylos et B. Myers, «Mapping the space of API design decisions,» *IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 50-57, 2007.

[22] X. Tao et P. J, «MAPO: Mining API usages from open source repositories,» *International Workshop on Mining Software Repositories*, pp. 54-57, 2006.

[23] S. Lee, S. Lee, S. Lim, J. Jung, S. Choi, N. Kim et J.-B. Lee, «An API design process in terms of usability: A case study on building more usable apis for smart TV platform,» 2014.

[24] G. Rama et A. Kak, «Some structural measures of API usability,» *Software - Practice and Experience*, vol. 45, pp. 75-110, 2015.

[25] B. Ellis, J. Stylos et B. Myers, «The factory pattern in API design: A usability evaluation,» *29th International Conference on Software Engineering*, pp. 302-311, 2007.

[26] W. Wu, B. Adams, Y.-G. Guéhéneuc et G. Antoniol, «ACUA: API change and usage auditor,» 2014.

[27] S. G. McLellan, A. W. Roesler, J. T. Tempest et C. I. Spinuzzi, «Building More Usable APIs,» *IEEE Software*, pp. 78-86, 1998.

[28] J. Sushine, J. Herbsleb et J. Aldrich, «Searching the State Space: A Qualitative Study of API Protocol Usability,» *23rd IEEE International Conference on Program Comprehension*, pp. 82-93, 2015.

[29] J. Stylos et S. Clarke, «Usability implications of requiring parameters in objects' constructors,» *29th International Conference on Software Engineering*, n°14222614, pp. 529-538, 2007.

[30] W. Wang, H. Malik et M. Godfrey, «Recommending posts concerning API issues in developer Q&A sites,» *12th Working Conference on Mining Software Repositories*, n°17180082, pp. 224-234, 2015.

[31] M. Zibran, F. Eishita et C. Roy, «Useful, but usable? Factors affecting the usability of APIs,» *18th Working Conference on Reverse Engineering*, n°16079520, pp. 151-155, 2011.

[32] P. Petersen, S. Hanenberg et R. Robbes, «An empirical comparison of static and dynamic type systems on API usage in the presence of an IDE: Java vs. Groovy with eclipse,» 2014.

[33] R. Buse et W. Weimer, «Synthesizing API usage examples,» *34th International Conference on Software Engineering*, n°16227140, pp. 782-792, 2012.

[34] S. Nasehi et F. Maurer, «Unit tests as API usage examples,» *IEEE International Conference on Software Maintenance*, n°15609553, 2010.

[35] L. Wang, Z. Y., F. L., X. B. et Y. F., «An exploratory study of API usage examples on the web,» *Asia-Pacific Software Engineering Conference, APSEC*, vol. 1, n°16462686, pp. 396-405, 2012.

[36] M. Asaduzzaman, R. C.K, K. Schneider et D. Hou, «Context-sensitive code completion tool for better API usability,» *30th International Conference on Software Maintenance and Evolution*, n°16976154, pp. 621-624, 2014.

[37] M. Bruch, M. Monperrus et M. Mezini, «Learning from Examples to Improve Code Completion,» *ESEC-FSE'09*, pp. 213-222, 2009.

[38] M. Saied, H. Sahraoui et B. Dufour, «An observational study on API usage constraints and their documentation,» *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering*, n°17081813, pp. 33-42, 2015.

[39] J. Juanjuan, J. Koskinen, A. Ruokonen et T. Systä, «Constructing usage scenarios for API redocumentation,» *15th IEEE International Conference on Program Comprehension*, n°14268260, pp. 259-264, 2007.

[40] U. Farooq, L. Welicki et D. Zirkler, «API usability peer reviews: A method for evaluating the usability of application programming interfaces,» *28th Annual CHI Conference on Human Factors in Computing Systems*, vol. 4, pp. 2327-2336, 2010.