Java APIs Commonly Used for API Research Evaluation

Maxime Lamothe
Department of Computer Science and Software Engineering
Concordia University
Montreal, Canada
Email: max_lam@encs.concordia.ca

***Abstract*—APIs are evermore common in software engineering. Using APIs is now a routine part of the software development lifecycle. Research into APIs has therefore sensibly increased to match the rising adoption rate of APIs and the challenges uncovered by this growth in popularity. However, our research shows that, in research, a few common APIs are frequently used to make or test most API research inquiries. These APIs are primarily Java APIs, and may not reflect the state of API evolution as a whole. This technical report was created to highlight which APIs are most commonly used in API research, in the hopes of raising awareness of current knowledge gaps in the field to improve the status quo.**

I. Introduction

In this work we present the 17 most popular systems used to evaluate software engineering research into APIs. We selected 143 published works extracted from a recent systematic review of API evolution literature and manually determined which APIs were used to either produce or test the hypotheses presented in each published work. As well as presenting the most common APIs used for research evaluation, we also highlight how each API was used to evaluate existing API research.

II. Popular APIs

Java API

With 39 independent papers within our sample dataset, the most common API used to evaluate API evolution research is the Java API either through the use of its various standard libraries, or through the JDK [1]. The Java API is widespread and has a large userbase [2]. Furthermore, the Java API benefits from a large number of open source projects available in online repositories like GitHub. For example GitHub contains 879,265 Java based projects [2]. Many papers that presents tools or approaches improve API usability [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] and help with API evolution and migration [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] make use of the Java API to evaluate the effectiveness of their approach. The Java API has been used to conduct empirical studies on API evolution [27], [2] and API usability [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [38]. The Java API has also been used to construct API quality datasets [24], and to evaluate API security tools [39].

Android API

The Android API popular for evaluating hypotheses for numerous reasons. It is a large and open source API [40], and the API benefits from a large user base through the Android ecosystem [41]. In this section, we do not distinguish between studies that make use of the Android API and Android apps to calculate the number of studies that use the Android API. We conside the Android apps presented within the context of the studies in this dataset as examples of Android API users since they are presented from the perspective of the Android API in their respective papers. A large portion of studies that use the Android API as an evaluation system conduct empirical studies on the evolution of APIs [42], [43], [44], [45], [46], [27], [47], [40], [48], [41], [49], [50], [51]. However, other studies also employ the Android ecosystem for the evaluation of various tools or approaches to help with API evolution [52], [53], [54], [55], [56], and API usability [57], [58], [59], [60], [61], [62]. Finally some papers make use of the Android API to evauate empirical studies on software usability [30], [63], [33], [64], [65], and software performance [66]

Toy systems

We consider simple systems that are produced for the sole sake of evaluating an approach presented in a paper to be toy systems. These toy systems can be used to showcase a tool. However, they are not necessarily representative of existing projects that can be found within an APIs ecosystem. 20 of the 291 publications we sampled made use of such systems. These toy systems are used for a variety of studies such as API refactoring tools [67], [68], [69], [70], [71], API documentation studies and tools [72], [73], [74], [75], [76], studies on managing API evolution [77], [78], [79], [80], [81], [82], and studies on understanding and developing better APIs [83], [84], [85], [86].

We find that 12 out of 20 studies that make use of toy systems to evaluate their findings use the Java programming language to create these toy systems. The other studies that make use of toy systems make use of varied programming languages such as BPEL [72].

Eclipse

Eclipse is an industrial yet open-source Java IDE [87]. Eclipse freely provides access to the source code to it's framework which can then be used for evaluation by researchers. Eclipse has been used as an evaluation system for API evolution

empirical studies [88], [89], [90], API usability studies [91], [92], [63], [93], an API conformance checking tool [94] API evolution mining tools and approaches [95], [96], [97], empirical studies on API refactoring [98], [99], an API migration recommendation tool [87], and API refactoring detection tools [100], [101].

JHotDraw

JHotDraw is a medium sized Java GUI framework created to demonstrate design patters [102]. JHotDraw has been used to evaluate API recommendation tools [103], API usage mining tools [104], API refactoring detection tools [100], [101], API migration tools and approaches [105], [106], [107], API change rules evolution in empirical studies [102], [108], [96], [89], and refactorings in an API upgrade case study [71].

Log4j

Log4j is a Java library that provides application logging functionality [109]. Various studies that present API tooling such as, API usage extraction tools [110], [26], API recommendation tools [111], API refactoring detection tool [100], and API migration tools [109] make use of the Log4j API to test the effectiveness of their tools. However, papers that present empirical studies such as API evolution studies [27], [112], [89], [99], studies on API documentation evolution [113], and studies that observe API compatibility [114] also use the Log4j API as a benchmark.

Struts

Apache Struts is a Java MVC framework for creating Java web applications [53]. In this section we do not distinguish between Struts and Struts 2. Struts is mainly used to test API tools and approaches such as detection of refactoring in APIs [101], [100], mining framework changes [96], [115], [53], API recommendation tools [5], and tools to detect dynamic API specifications [116]. However, in two cases empirical studies use struts to validate API evolution hypotheses [112], [89].

Guava

Guava is a Java library of collection utilities that were not originally provided as part of the Java SDK [110]. Over 3,000 Guava clients exist on GitHub [110]. Guava has been used to test a variety of hypotheses, ranging from API usage analysis [110], [117] API deprecation [2], [118], API documentation analysis [30], the impact of refactoring on API clients [119], and the impact of unbundling APIs [120].

Hibernate

Hibernate is a framework for mapping an object oriented domain to a relational database [37]. Hibernate has over 1000 deprecated APIs over it's history, making it a prime candidate to test API deprecation hypotheses [37]. Hibernate has several user projects available on GitHub [61] and many questions on online forums, also making it a good candidate for approaches that learn API characteristics from online forums [61], [90], or studies that observe API usability [91], [117], [121]. It has also been used as a test subject for hypotheses about API documentation [122].

JUnit

JUnit is a popular open source Java testing framework [91]. It is used as a test subject for studies about API documentation [30], [122], API usage patterns [26], [110], [91], API evolution problems [123], and API migration [124].

JFreeChart

JFreeChart is a Java chart library with over 54 releases that contain many API changes with similar names [102]. The change history of JFreeChart makes it a good API to test change rules in APIs [102], understanding unfamiliar APIs [125], and testing API recommendation [126] and migration [105], [121], [106], [127] tools.

Proprietary systems

Not all systems used for API research are open-source systems. Six of the papers in our sample test or build their hypotheses upon proprietary closed source systems from various companies. These studies are nevertheless varied in scope, and do not appear to be limited by the closed nature of their source code. The studies range from API usability and design [128], [129], [130], extracting API usage patterns [131], and understanding API evolution [132], [133].

Spring

Spring is a framework that provides access to Java objects through reflection [110]. It is a popular project, that has at least 150 classes, and has at least 10 commits per week over its lifetime [110]. It is employed as a test subject by six of the papers in our sample dataset. It is used a a test subject for studies in API recommendation [111], improving API documentation [90], [118], understanding developer reaction to deprecation [2], and for approaches to understand API usage [110], [117].

Hadoop

Hadoop is one of the most popular Java libraries developed under the Apache foundation [5]. Hadoop is used as a test subject by a variety of studies in our dataset. Most of the works that employ Hadoop as a test system concentrate on API documentation, either by detecting documentation errors [134], recommending or searching for API documentation [122], [30], or exploring API documentation quality [118]. However, Hadoop is also used to test API recommendation tools [5], and as a test subject to keep track of API popularity [52].

Lucene

Lucene is a free and open-source search engine Java library [135]. Six of the papers in our dataset use Lucene as an API to test their hypotheses, however none of these studies highlight why Lucene is a prime candidate as a test API. In all six instance, Lucene is selected as one of several test APIs, and Lucene never appears as a singular test API without our dataset. Lucene is used in a variety of studies, from API migration studies [136], API deprecation studies [118], API documentation evolution and error detection studies [113], [134], API specification checking studies [94], and API refactoring detection studies [100].

Pharo

Pharo is a dynamically typed programming language, with over 3600 distinct systems and over 6 years of evolution [137]. Therefore the API is seen as a good candidate for ecosystem studies. In particular studies that use the Pharo ecosystem concentrate on the ripple effects of API changes on an ecosys-

tem [138], how developers react to API deprecation [139], and how developers react to the evolution of an API [140], [141], [137]. One of the studies in our dataset also used the Pharo API as a test subject to benchmark a tool that extracts API changes that occur during API evolution [142].

.Net API

Within our dataset, the .Net API as a test API is always coupled with the Java API. The APIs can be couple as a comparison since both APIs present similar features, and a large number of client programs [34]. However, in one case both APIs are required to test the hypothesis since the goal of the study is to build a migration mapping between two APIs [20], [21]. In the majority of cases however, the APIs are chosen to provide results that are valid across languages, either to uncover patterns of knowledge [143], or as responses to user surveys [144].

III. Conclusion

In this paper we present the 17 most popular APIs used to evaluate software engineering research into APIs. We find that most of the popular APIs used for evaluation are Java APIs, with a few outliers such as the Pharo API and the .Net API. We hope that by highlighting the most common APIs used to evaluate past API research, the information presented in this technical report can be used to foster future API research by facilitating the replication of existing work, as well as highlighting the lack of programming language variety in existing API research.

References

[1] "what is an interface? (the java™ tutorials learning the java language object-oriented programming concepts," 2019.

[2] A. A. Sawant, R. Robbes, and A. Bacchelli, "On the reaction to deprecation of clients of 4 + 1 popular Java APIs and the JDK," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2158–2197, aug 2018.

[3] C. McMillan, D. Poshyvanyk, and M. Grechanik, "Recommending source code examples via API call usages and documentation," *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering - RSSE '10*, pp. 21–25, 2010.

[4] J. Gao, P. Kong, L. Li, T. F. Bissyande, and J. Klein, "Negative Results on Mining Crypto-API Usage Rules in Android Apps," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, may 2019, pp. 388–398.

[5] F. Thung, S. Wang, D. Lo, and J. Lawall, "Automatic recommendation of API methods from feature requests," *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 290–300, nov 2013.

[6] H. A. Nguyen, R. Dyer, T. N. Nguyen, and H. Rajan, "Mining preconditions of APIs in large-scale code corpus," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*. New York, New York, USA: ACM Press, 2014, pp. 166–177.

[7] M. M. Rahman, C. K. Roy, and D. Lo, "RACK: Automatic API Recommendation Using Crowdsourced Knowledge," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, mar 2016, pp. 349–359.

[8] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from stack overflow," in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, vol. 14-22-May-. New York, New York, USA: ACM Press, 2016, pp. 392–403.

[9] A. Reinhardt, T. Zhang, M. Mathur, and M. Kim, "Augmenting stack overflow with API usage patterns mined from GitHub," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. New York, New York, USA: ACM Press, 2018, pp. 880–883.

[10] T. Nguyen, N. Tran, H. Phan, T. Nguyen, L. Truong, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Complementing global and local contexts in representing API descriptions to improve API retrieval tasks," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. New York, New York, USA: ACM Press, 2018, pp. 551–562.

[11] M. Liu, X. Peng, A. Marcus, Z. Xing, W. Xie, S. Xing, and Y. Liu, "Generating query-specific class API summaries," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2019*, ser. ESEC/FSE 2019. New York, New York, USA: ACM Press, 2019, pp. 120–130.

[12] M. Wen, Y. Liu, R. Wu, X. Xie, S.-C. Cheung, and Z. Su, "Exposing Library API Misuses Via Mutation Analysis," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, may 2019, pp. 866–877.

[13] G. Antoniol, M. Di Penta, and E. Merlo, "An automatic approach to identify class evolution discontinuities," in *Proceedings. 7th International Workshop on Principles of Software Evolution, 2004.* IEEE, sep 2004, pp. 31–40.

[14] B. Gharaibeh, T. N. Nguyen, and J. M. Chang, "Coping with API Evolution for Running, Mission-Critical Applications Using Virtual Execution Environment," in *Seventh International Conference on Quality Software (QSIC 2007)*, no. Qsic. IEEE, 2007, pp. 171–180.

[15] H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei, "MAPO: Mining and Recommending API Usage Patterns," in *Proceedings of the 23rd European Conference on ECOOP 2009 — Object-Oriented Programming*, ser. Genoa. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 318–343.

[16] G. Uddin, B. Dagenais, and M. P. Robillard, "Analyzing temporal API usage patterns," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, nov 2011, pp. 456–459.

[17] W. Zheng, Q. Zhang, and M. Lyu, "Cross-library API recommendation using web search engines," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering - SIGSOFT/FSE '11*. New York, New York, USA: ACM Press, 2011, p. 480.

[18] S. Bouktif, H. Sahraoui, and F. Ahmed, "Predicting Stability of Open-Source Software Systems Using Combination of Bayesian Classifiers," *ACM Transactions on Management Information Systems*, vol. 5, no. 1, pp. 1–26, apr 2014.

[19] A. T. Nguyen, M. Hilton, M. Codoban, H. A. Nguyen, L. Mast, E. Rademacher, T. N. Nguyen, and D. Dig, "API code recommendation using statistical learning from fine-grained changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*. New York, New York, USA: ACM Press, 2016, pp. 511–522.

[20] T. D. Nguyen, A. T. Nguyen, H. D. Phan, and T. N. Nguyen, "Exploring API Embedding for API Usages and Applications," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, may 2017, pp. 438–449.

[21] N. D. Q. Bui, Y. Yu, and L. Jiang, "SAR: learning cross-language API mappings with little knowledge," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2019*. New York, New York, USA: ACM Press, 2019, pp. 796–806.

[22] M. Kim and D. Notkin, "Discovering and representing systematic code changes," in *2009 IEEE 31st International Conference on Software Engineering*. IEEE, 2009, pp. 309–319.

[23] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, pp. 1053–1058, Dec. 1972.

[24] S. Amani, S. Nadi, H. A. Nguyen, T. N. Nguyen, and M. Mezini, "MUBench," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*. New York, New York, USA: ACM Press, may 2016, pp. 464–467.

[25] H. Ma, R. Amor, and E. Tempero, "Indexing the Java API Using Source

Code," in *19th Australian Conference on Software Engineering (aswec 2008)*. IEEE, mar 2008, pp. 451–460.

[26] S. Thummalapenta and T. Xie, "SpotWeb: Detecting Framework Hotspots and Coldspots via Mining Open Source Code on the Web," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, sep 2008, pp. 327–336.

[27] S. Mostafa, R. Rodriguez, and X. Wang, "Experience paper: a study on behavioral backward incompatibilities of Java software libraries," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2017*. New York, New York, USA: ACM Press, 2017, pp. 215–225.

[28] U. Dekel and J. D. Herbsleb, "Reading the documentation of invoked API functions in program comprehension," in *2009 IEEE 17th International Conference on Program Comprehension*. IEEE, may 2009, pp. 168–177.

[29] D. Hou and L. Li, "Obstacles in Using Frameworks and APIs: An Exploratory Study of Programmers' Newsgroup Discussions," in *2011 IEEE 19th International Conference on Program Comprehension*. IEEE, jun 2011, pp. 91–100.

[30] C. Treude and M. Aniche, "Where does Google find API documentation?" *Proceedings of the 2nd International Workshop on API Usage and Evolution - WAPI '18*, pp. 19–22, 2018.

[31] D. Ancona, F. Dagnino, and L. Franceschini, "A formalism for specification of Java API interfaces," in *Companion Proceedings for the ISSTA/ECOOP 2018 Workshops on - ISSTA '18*, vol. 2. New York, New York, USA: ACM Press, 2018, pp. 24–26.

[32] M. A. Saied, H. Sahraoui, and B. Dufour, "An observational study on API usage constraints and their documentation," *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 33–42, mar 2015.

[33] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," *Georgia Tech Technical Report*, pp. 1–11, 2012.

[34] D. Fucci, A. Mollaalizadehbahnemiri, and W. Maalej, "On using machine learning to identify knowledge in API reference documentation," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2019*. New York, New York, USA: ACM Press, 2019, pp. 109–119.

[35] F. Pontes, R. Gheyi, S. Souto, A. Garcia, and M. Ribeiro, "Java reflection API: revealing the dark side of the mirror," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2019*. New York, New York, USA: ACM Press, 2019, pp. 636–646.

[36] T. V. Nguyen and T. N. Nguyen, "Inferring API elements relevant to an english query," in *Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18*, vol. Part F1373. New York, New York, USA: ACM Press, 2018, pp. 167–168.

[37] A. A. Sawant, M. Aniche, A. van Deursen, and A. Bacchelli, "Understanding developers' needs on deprecation as a language feature," in *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*, vol. 11. New York, New York, USA: ACM Press, 2018, pp. 561–571.

[38] D. Kawrykow and M. P. Robillard, "Improving API Usage through Automatic Detection of Redundant Code," in *2009 IEEE/ACM International Conference on Automated Software Engineering*. IEEE, nov 2009, pp. 111–122.

[39] S. Huang, J. Guo, S. Li, X. Li, Y. Qi, K. Chow, and J. Huang, "SafeCheck: Safety Enhancement of Java Unsafe API," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, may 2019, pp. 889–899.

[40] M. Lamothe and W. Shang, "Exploring the use of automated API migrating techniques in practice," *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*, pp. 503–514, 2018.

[41] L. Li, J. Gao, T. F. Bissyandé, L. Ma, X. Xia, and J. Klein, "Characterising deprecated Android APIs," in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*. New York, New York, USA: ACM Press, 2018, pp. 254–264.

[42] T. Espinha, A. Zaidman, and H.-G. Gross, "Web API Fragility: How Robust is Your Mobile Application?" *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, pp. 12–21, may 2015.

[43] T. McDonnell, B. Ray, and M. Kim, "An Empirical Study of API Stability and Adoption in the Android Ecosystem," in *2013 IEEE International Conference on Software Maintenance*. IEEE, sep 2013, pp. 70–79.

[44] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "API change and fault proneness: a threat to the success of Android apps," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*. New York, New York, USA: ACM Press, 2013, p. 477.

[45] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "How do API changes trigger stack overflow discussions? a study on the Android SDK," in *Proceedings of the 22nd International Conference on Program Comprehension - ICPC 2014*. New York, New York, USA: ACM Press, 2014, pp. 83–94.

[46] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. D. Penta, R. Oliveto, and D. Poshyvanyk, "The Impact of API Change- and Fault-Proneness on the User Ratings of Android Apps," *IEEE Transactions on Software Engineering*, vol. 41, no. 4, pp. 384–407, apr 2015.

[47] G. Grano, A. Di Sorbo, F. Mercaldo, C. A. Visaggio, G. Canfora, and S. Panichella, "Android apps and user feedback: a dataset for software evolution and quality improvement," in *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics - WAMA 2017*, no. ii. New York, New York, USA: ACM Press, 2017, pp. 8–11.

[48] D. He, L. Li, L. Wang, H. Zheng, G. Li, and J. Xue, "Understanding and detecting evolution-induced compatibility issues in Android apps," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018*. New York, New York, USA: ACM Press, 2018, pp. 167–177.

[49] P. Calciati, K. Kuznetsov, X. Bai, and A. Gorla, "What did really change with the new release of the app?" in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*. New York, New York, USA: ACM Press, 2018, pp. 142–152.

[50] S. Um, "The evolution of a digital ecosystem," in *Companion to the Proceedings of the 11th International Symposium on Open Collaboration - OpenSym '15*. New York, New York, USA: ACM Press, 2015, pp. 1–1.

[51] M. Linares-Vásquez, "Supporting evolution and maintenance of Android apps," in *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*. New York, New York, USA: ACM Press, 2014, pp. 714–717.

[52] A. Hora and M. T. Valente, "Apiwave: Keeping track of API popularity and migration," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, sep 2015, pp. 321–323.

[53] P. Yu, F. Yang, C. Cao, H. Hu, and X. Ma, "API Usage Change Rules Mining based on Fine-grained Call Dependency Analysis," in *Proceedings of the 9th Asia-Pacific Symposium on Internetware - Internetware'17*, vol. Part F1309. New York, New York, USA: ACM Press, 2017, pp. 1–9.

[54] G. Yang, J. Jones, A. Moninger, and M. Che, "How do Android operating system updates impact apps?" in *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems - MOBILESoft '18*. New York, New York, USA: ACM Press, 2018, pp. 156–160.

[55] A. Brito, L. Xavier, A. Hora, and M. T. Valente, "APIDiff: Detecting API breaking changes," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, vol. 2018-March, no. Dcc. IEEE, mar 2018, pp. 507–511.

[56] S. Scalabrino, G. Bavota, M. Linares-Vasquez, M. Lanza, and R. Oliveto, "Data-Driven Solutions to Detect API Compatibility Issues in Android: An Empirical Study," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, vol. 2019-May. IEEE, may 2019, pp. 288–298.

[57] J. E. Montandon, H. Borges, D. Felix, and M. T. Valente, "Documenting APIs with examples: Lessons learned with the APIMiner platform," in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, oct 2013, pp. 401–408.

[58] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live API documentation," in *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. New York, New York, USA: ACM Press, 2014, pp. 643–652.

[59] S. Azad, P. C. Rigby, and L. Guerrouj, "Generating API Call Rules from Version History and Stack Overflow Posts," *ACM Transactions on Software Engineering and Methodology*, vol. 25, no. 4, pp. 1–22, jan 2017.

[60] W. Yuan, H. H. Nguyen, L. Jiang, and Y. Chen, "LibraryGuru," in *Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18*. New York, New York, USA: ACM Press, 2018, pp. 364–365.

[61] H. Phan, H. A. Nguyen, N. M. Tran, L. H. Truong, A. T. Nguyen, and T. N. Nguyen, "Statistical learning of API fully qualified names in code snippets of online forums," in *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*, vol. 11, no. 18. New York, New York, USA: ACM Press, 2018, pp. 632–642.

[62] L. Wei, Y. Liu, and S.-C. Cheung, "PIVOT: Learning API-Device Correlations to Facilitate Android Compatibility Issue Detection," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, may 2019, pp. 878–888.

[63] M. F. Zibran, F. Z. Eishita, and C. K. Roy, "Useful, But Usable? Factors Affecting the Usability of APIs," in *2011 18th Working Conference on Reverse Engineering*, no. Table II. IEEE, oct 2011, pp. 151–155.

[64] W. Wang and M. W. Godfrey, "Detecting API usage obstacles: A study of iOS and Android developer questions," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, may 2013, pp. 61–64.

[65] R. Heumüller, S. Nielebock, and F. Ortmeier, "Who plays with whom? ... and how? mining API interaction patterns from source code," in *Proceedings of the 7th International Workshop on Software Mining - SoftwareMining 2018*. New York, New York, USA: ACM Press, 2018, pp. 8–11.

[66] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Mining energy-greedy API usage patterns in Android apps: an empirical study," *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, pp. 2–11, 2014.

[67] T. Freese, "Inline Method Considered Helpful: An Approach to Interface Evolution," 2003, pp. 271–278.

[68] Z. Troni, "API Evolution with RefactoringNG," in *2010 Second World Congress on Software Engineering*, vol. 2. IEEE, dec 2010, pp. 293–297.

[69] Z. Troníček, "RefactoringNG," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*. New York, New York, USA: ACM Press, 2012, p. 1165.

[70] J. Henkel and A. Diwan, "Catchup! capturing and replaying refactorings to support API evolution," in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* IEEe, 2005, pp. 274–283.

[71] I. Şavga, M. Rudolf, S. Götz, and U. Aßmann, "Practical refactoring-based framework upgrade," in *Proceedings of the 7th international conference on Generative programming and component engineering - GPCE '08*. New York, New York, USA: ACM Press, 2008, p. 171.

[72] B. Benatallah, F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani, "Developing Adapters for Web Services Integration," in *Advanced Information Systems Engineering*, O. Pastor and J. e Cunha, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 415–429.

[73] J. Jiang, J. Koskinen, A. Ruokonen, and T. Systa, "Constructing Usage Scenarios for API Redocumentation," in *15th IEEE International Conference on Program Comprehension (ICPC '07)*. IEEE, jun 2007, pp. 259–264.

[74] C. Chen and K. Zhang, "Who asked what: integrating crowdsourced FAQs into API documentation," in *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, ser. ICSE Companion 2014. New York, New York, USA: ACM Press, 2014, pp. 456–459.

[75] J. Stylos, A. Faulring, Z. Yang, and B. A. Myers, "Improving API documentation using API usage information," in *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, sep 2009, pp. 119–126.

[76] S. Endrikat, S. Hanenberg, R. Robbes, and A. Stefik, "How do API documentation and static typing affect API usability?" in *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*, no. 1. New York, New York, USA: ACM Press, 2014, pp. 632–642.

[77] R. Elio, E. Stroulia, and W. Blanchet, "Using Interaction Models to Detect and Resolve Inconsistencies in Evolving Service Compositions," *Web Intelli. and Agent Sys.*, vol. 7, no. 2, pp. 139–160, apr 2009.

[78] D. Rose, S. Stegmaier, G. Reina, D. Weiskopf, and T. Ertl, "Non-invasive Adaptation of Black-box User Interfaces," *Reproduction*, 2002.

[79] Zhenchang Xing and E. Stroulia, "Understanding class evolution in object-oriented software," in *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.* IEEE, jun 2004, pp. 34–43.

[80] S. R. Ponnekanti and A. Fox, "Interoperability Among Independently Evolving Web Services," in *5th ACM/IFIP/USENIX International Conference on Middleware, MIDDLEWARE 2004*, 2004, vol. LNCS 3231, pp. 331–351.

[81] T. Nguyen, E. Munson, and C. Thao, "Managing the evolution of Web-based applications with WebSCM," in *21st IEEE International Conference on Software Maintenance (ICSM'05)*. IEEE, 2005, pp. 577–586.

[82] B. Ellis, J. Stylos, and B. Myers, "The Factory Pattern in API Design: A Usability Evaluation," *29th International Conference on Software Engineering (ICSE'07)*, pp. 302–312, may 2007.

[83] I. Porres and I. Rauf, "Modeling behavioral RESTful web service interfaces in UML," in *Proceedings of the 2011 ACM Symposium on Applied Computing - SAC '11*, ser. SAC '11. New York, New York, USA: ACM Press, 2011, p. 1598.

[84] R. Tsouroplis, M. Petychakis, I. Alvertis, E. Biliri, F. Lampathaki, and D. Askounis, "Internet-Based Enterprise Innovation Through a Community-Based API Builder to Manage APIs," 2015, pp. 65–76.

[85] Andrew J. Ko, B. Myers, and H. Aung, "Six Learning Barriers in End-User Programming Systems," in *2004 IEEE Symposium on Visual Languages - Human Centric Computing*. IEEE, 2004, pp. 199–206.

[86] G. Mesfin, T.-M. Grønli, D. Midekso, and G. Ghinea, "Towards end-user development of REST client applications on smartphones," *Computer Standards & Interfaces*, vol. 44, pp. 205–219, feb 2016.

[87] B. Dagenais and M. P. Robillard, "SemDiff: Analysis and recommendation support for API evolution," in *2009 IEEE 31st International Conference on Software Engineering*. IEEE, 2009, pp. 599–602.

[88] J. Kim and E. Lee, "The effect of IMPORT change in software change history," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14*. New York, New York, USA: ACM Press, 2014, pp. 1753–1754.

[89] D. Dig and R. Johnson, "How do APIs evolve? A story of refactoring," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 18, no. 2, pp. 83–107, mar 2006.

[90] B. Dagenais and M. P. Robillard, "Creating and evolving developer documentation," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering - FSE '10*, ser. FSE '10. New York, New York, USA: ACM Press, 2010, p. 127.

[91] A. Hora, M. T. Valente, R. Robbes, and N. Anquetil, "When should internal interfaces be promoted to public?" in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, vol. 13-18-Nove. New York, New York, USA: ACM Press, 2016, pp. 278–289.

[92] J. Businge, A. Serebrenik, and M. van den Brand, "Analyzing the Eclipse API Usage: Putting the Developer in the Loop," in *2013 17th European Conference on Software Maintenance and Reengineering*. IEEE, mar 2013, pp. 37–46.

[93] R. Holmes and R. J. Walker, "A newbie's guide to eclipse APIs," in *Proceedings of the 2008 international workshop on Mining software repositories - MSR '08*. New York, New York, USA: ACM Press, 2008, p. 149.

[94] M. Pradel, C. Jaspan, J. Aldrich, and T. R. Gross, "Statically checking API protocol conformance with mined multi-object specifications," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, jun 2012, pp. 925–935.

[95] Y. M. Mileva, A. Wasylkowski, and A. Zeller, "Mining Evolution of Object Usage," 2011, pp. 105–129.

[96] T. Schäfer, J. Jonas, and M. Mezini, "Mining framework usage changes from instantiation code," in *Proceedings of the 13th international conference on Software engineering - ICSE '08*, ser. ICSE '08. New York, New York, USA: ACM Press, 2008, p. 471.

[97] D. Mandelin, L. Xu, R. Bodík, and D. Kimelman, "Jungloid mining," *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation - PLDI '05*, p. 48, 2005.

[98] M. Kim, D. Cai, and S. Kim, "An empirical investigation into the role of API-level refactorings during software evolution," *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, p. 151, 2011.

[99] D. Dig and R. Johnson, "The role of refactorings in API evolution," *21st IEEE International Conference on Software Maintenance (ICSM'05)*, vol. 2005, pp. 389–398, 2005.

[100] K. Taneja, D. Dig, and T. Xie, "Automated detection of api refactorings in libraries," in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*. New York, New York, USA: ACM Press, 2007, p. 377.

[101] D. Dig, C. Comertoglu, D. Marinov, and R. Johnson, "Automated Detection of Refactorings in Evolving Components," in *Proceedings of the 20th European Conference on Object-Oriented Programming*, 2006, no. July, pp. 404–428.

[102] W. Wu, A. Serveaux, Y.-G. Guéhéneuc, and G. Antoniol, "The impact of imperfect change rules on framework API evolution identification: an empirical study," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1126–1158, aug 2015.

[103] M. Asaduzzaman, C. K. Roy, S. Monir, and K. A. Schneider, "Exploring API method parameter recommendations," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, sep 2015, pp. 271–280.

[104] C. De Roover, R. Lammel, and E. Pek, "Multi-dimensional exploration of API usage," in *2013 21st International Conference on Program Comprehension (ICPC)*. IEEE, may 2013, pp. 152–161.

[105] W. Wu, Y.-G. Guéhéneuc, G. Antoniol, and M. Kim, "AURA," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, vol. 1. New York, New York, USA: ACM Press, 2010, p. 325.

[106] M. Kim, D. Notkin, and D. Grossman, "Automatic Inference of Structural Changes for Matching across Program Versions," in *29th International Conference on Software Engineering (ICSE'07)*, ser. ICSE '07. Washington, DC, USA: IEEE, may 2007, pp. 333–343.

[107] T. Tourwe and T. Mens, "Automated support for framework-based software," in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.* IEEE Comput. Soc, 2003, pp. 148–157.

[108] R. Yokomori, H. Siy, M. Noro, and K. Inoue, "Assessing the impact of framework changes using component ranking," in *2009 IEEE International Conference on Software Maintenance*. IEEE, sep 2009, pp. 189–198.

[109] P. Kapur, B. Cossette, and R. J. Walker, "Refactoring references for library migration," *ACM SIGPLAN Notices*, vol. 45, no. 10, p. 726, oct 2010.

[110] A. A. Sawant and A. Bacchelli, "fine-GRAPE: fine-grained APi usage extractor – an approach and dataset to investigate API usage," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1348–1371, jun 2017.

[111] X. Liu, L. Huang, and V. Ng, "Effective API recommendation without historical software repositories," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018*. New York, New York, USA: ACM Press, 2018, pp. 282–292.

[112] B. E. Cossette and R. J. Walker, "Seeking the ground truth," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12*, ser. FSE '12. New York, New York, USA: ACM Press, 2012, p. 1.

[113] L. Shi, H. Zhong, T. Xie, and M. Li, "An Empirical Study on Evolution of API Documentation," 2011, pp. 416–431.

[114] J. Hýbl and Z. Troníček, "On testing the source compatibility in Java," in *Proceedings of the 2013 companion publication for conference on Systems, programming, & applications: software for humanity - SPLASH '13*. New York, New York, USA: ACM Press, 2013, pp. 87–88.

[115] S. Meng, X. Wang, L. Zhang, and H. Mei, "A history-based matching approach to identification of framework evolution," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, jun 2012, pp. 353–363.

[116] Z. Alsaeed and M. Young, "Extending existing inference tools to mine dynamic APIs," in *Proceedings of the 2nd International Workshop on API Usage and Evolution - WAPI '18*. New York, New York, USA: ACM Press, 2018, pp. 23–26.

[117] A. A. Sawant and A. Bacchelli, "A Dataset for API Usage," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, vol. 2015-Augus. IEEE, may 2015, pp. 506–509.

[118] D. Ko, K. Ma, S. Park, S. Kim, D. Kim, and Y. L. Traon, "API Document Quality for Resolving Deprecated APIs," in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 2. IEEE, dec 2014, pp. 27–30.

[119] R. G. Kula, A. Ouni, D. M. German, and K. Inoue, "An empirical study on the impact of refactoring activities on evolving client-used APIs," *Information and Software Technology*, vol. 93, no. July 2016, pp. 186–199, jan 2018.

[120] A. S. Matos, J. B. Ferreira Filho, and L. S. Rocha, "Splitting APIs: An Exploratory Study of Software Unbundling," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, may 2019, pp. 360–370.

[121] H. A. Nguyen, T. T. Nguyen, G. Wilson, A. T. Nguyen, M. Kim, and T. N. Nguyen, "A graph-based approach to API usage adaptation," *ACM SIGPLAN Notices*, vol. 45, no. 10, p. 302, oct 2010.

[122] M. P. Robillard and Y. B. Chhetri, "Recommending reference API documentation," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1558–1586, dec 2015.

[123] J. Dietrich, K. Jezek, and P. Brada, "Broken promises: An empirical study into evolution problems in Java programs caused by library upgrades," *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pp. 64–73, feb 2014.

[124] H. Zhong, S. Thummalapenta, T. Xie, L. Zhang, and Q. Wang, "Mining API mapping for language migration," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, vol. 1. New York, New York, USA: ACM Press, 2010, p. 195.

[125] E. Duala-Ekoko and M. P. Robillard, "Asking and answering questions about unfamiliar APIs: An exploratory study," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, jun 2012, pp. 266–276.

[126] ——, "Using Structure-Based Recommendations to Facilitate Discoverability in APIs," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6813 LNCS, pp. 79–104.

[127] Zhenchang Xing and E. Stroulia, "API-Evolution Support with Diff-CatchUp," *IEEE Transactions on Software Engineering*, vol. 33, no. 12, pp. 818–836, dec 2007.

[128] J. Stylos, B. Graf, D. K. Busse, C. Ziegler, R. Ehret, and J. Karstens, "A case study of API redesign for improved usability," in *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, sep 2008, pp. 189–192.

[129] E. Mosqueira-Rey, D. Alonso-Ríos, V. Moret-Bonillo, I. Fernández-Varela, and D. Álvarez-Estévez, "A systematic approach to API usability: Taxonomy-derived criteria and a case study," *Information and Software Technology*, vol. 97, no. December 2017, pp. 46–63, may 2018.

[130] A. Zghidi, I. Hammouda, B. Hnich, and E. Knauss, "On the Role of Fitness Dimensions in API Design Assessment - An Empirical Investigation," in *2017 IEEE/ACM 1st International Workshop on API Usage and Evolution (WAPI)*. IEEE, may 2017, pp. 19–22.

[131] J. Wang, Y. Dang, H. Zhang, K. Chen, T. Xie, and D. Zhang, "Mining succinct and high-coverage API usage patterns from source code," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, may 2013, pp. 319–328.

[132] W. Granli, J. Burchell, I. Hammouda, and E. Knauss, "The driving forces of API evolution," in *Proceedings of the 14th International Workshop on Principles of Software Evolution - IWPSE 2015*, vol. 30-Aug-201. New York, New York, USA: ACM Press, 2015, pp. 28–37.

[133] W. Zhang, Z. Lin, G. Xiao, J. Chen, J. Wang, and Y. Jiang, "LanguageTool proofreading rules evolution and update," in *Proceedings of 2018 International Conference on Big Data Technologies - ICBDT '18*, no. 18. New York, New York, USA: ACM Press, 2018, pp. 95–100.

[134] H. Zhong and Z. Su, "Detecting API documentation errors," *ACM SIGPLAN Notices*, vol. 48, no. 10, pp. 803–816, nov 2013.

[135] "Apache lucene," 2019.

[136] A. T. Nguyen, H. A. Nguyen, T. T. Nguyen, and T. N. Nguyen, "Statistical learning approach for mining API usage mappings for code migration," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering - ASE '14*. New York, New York, USA: ACM Press, 2014, pp. 457–468.

[137] A. Hora, R. Robbes, M. T. Valente, N. Anquetil, A. Etien, and S. Ducasse, "How do developers react to API evolution? A large-scale empirical study," *Software Quality Journal*, vol. 26, no. 1, pp. 161–191, mar 2018.

[138] R. Robbes and M. Lungu, "A study of ripple effects in software ecosystems," in *Proceeding of the 33rd international conference on*

Software engineering - ICSE '11, ser. ICSE '11. New York, New York, USA: ACM Press, 2011, p. 904.

[139] R. Robbes, M. Lungu, and D. Röthlisberger, "How do developers react to API deprecation?" in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12*. New York, New York, USA: ACM Press, 2012, p. 1.

[140] A. Hora, R. Robbes, N. Anquetil, A. Etien, S. Ducasse, and M. T. Valente, "How do developers react to API evolution? The Pharo ecosystem case," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, vol. 3, no. Dcc. IEEE, sep 2015, pp. 251–260.

[141] A. Hora, N. Anquetil, A. Etien, S. Ducasse, and M. T. Valente, "Automatic detection of system-specific conventions unknown to developers," *Journal of Systems and Software*, vol. 109, pp. 192–204, nov 2015.

[142] A. Hora, A. Etien, N. Anquetil, S. Ducasse, and M. T. Valente, "APIEvolutionMiner: Keeping API evolution under control," in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, feb 2014, pp. 420–424.

[143] W. Maalej and M. P. Robillard, "Patterns of Knowledge in API Reference Documentation," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1264–1282, sep 2013.

[144] M. P. Robillard, "What Makes APIs Hard to Learn? Answers from Developers," *IEEE Software*, vol. 26, no. 6, pp. 27–34, nov 2009.