Um Estudo em Larga Escala sobre o Uso de APIs Internas

Aline Brito, André Hora, Marco Tulio Valente

[1]ASERG Group – Departamento de Ciência da Computação (DCC)
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – Minas Gerais – Brasil.

alinebrito@ufmg.br, {hora,mtov}@dcc.ufmg.br

Abstract. *Libraries and frameworks are increasingly used when building software systems, and hence their APIs. It is common to divide the APIs into two types, public and internal. While public APIs are stable, internal APIs handle implementation details, are unsupported and unstable. Therefore, their use is not recommended. However, some systems use internal APIs, which is a bad practice in software development. In this paper, we study a set of questions related to the use of internal APIs. Our database is composed of around 260K Java projects and 131M APIs. This paper also includes comments from clients and providers of internal APIs.*

Resumo. *Bibliotecas e frameworks são cada vez mais usados no desenvolvimento de sistemas, e consequentemente suas APIs. É comum a divisão dessas APIs em dois tipos, públicas e internas. Enquanto as APIs públicas são estáveis, as APIs internas envolvem detalhes de implementação, são instáveis e sem suporte. Consequentemente, os provedores não recomendam o uso. Entretanto, verifica-se que alguns sistemas clientes usam APIs internas, uma má prática no desenvolvimento de software. Nesse contexto, estuda-se nesse artigo um conjunto de questões relativas a frequência de utilização de APIs internas. Utiliza-se um dataset com cerca de 260 mil projetos Java e 131 milhões de APIs. O artigo também inclui uma consulta aos clientes e provedores de APIs internas.*

1. Introdução

O uso de bibliotecas e as suas APIs (*Application Programming Interfaces*) no desenvolvimento e manutenção de sistemas de *software* é cada vez mais popular, já que aumentam a produtividade [Mileva et al. 2010, Hora and Valente 2015]. É comum os provedores dessas bibliotecas dividirem suas APIs em dois tipos, internas e públicas [Hora et al. 2016, Mastrangelo et al. 2015]. As APIs públicas são estáveis e tendem a manter a compatibilidade com versões anteriores. Já as APIs internas referem-se a detalhes de implementação, e não são recomendadas para uso externo ao projeto. Além disso, a atualização de uma API interna não garante a compatibilidade com outras versões.

Entretanto, alguns clientes fazem referências a APIs internas nos seus sistemas, isto é, essa má prática de programação é adotada apesar das recomendações contrárias [Businge et al. 2015, Hora et al. 2016, Mastrangelo et al. 2015]. Essas referências a interfaces internas podem envolver um grande impacto no projeto. Pode-se ter comportamentos inesperados no *software* e até mesmo a interrupção do seu funcionamento.

Para distinguir APIs internas e públicas, alguns provedores incluem pacotes específicos. Por exemplo, na IDE Eclipse, APIs internas são implementadas em pacotes que

possuem o nome "internal" [Businge et al. 2015, Businge et al. 2013]. Já o JDK utiliza o prefixo "sun" [Mastrangelo et al. 2015]. O texto a seguir reproduz parte das diretivas do Eclipse e da Oracle relacionadas ao uso de interfaces internas.

Eclipse. "*Packages containing only implementation details have "internal" in the package name. Legitimate client code must never reference the names of internal elements. Client code that oversteps the above rules might fail on diferent versions and patch levels of the platform.*[1]"

JDK. "*The sun.* packages are not part of the supported, public interface. A Java program that directly calls into sun.* packages is not guaranteed to work on all Java-compatible platforms. In fact, such a program is not guaranteed to work even in future versions on the same platform.*[2]"

Para melhor quantificar o cenário que motiva esse trabalho, realizou-se um estudo preliminar onde minerou-se a quantidade de projetos que utilizam pelo menos uma API com as nomenclaturas citadas. Através da infraestrutura Boa analisou-se 263.425 projetos Java GitHub [Dyer et al. 2013]. *Foram encontrados 17.910 projetos (6.8%) usando APIs internas*. Assim, estuda-se neste artigo duas questões de pesquisa centrais relativas ao uso de APIs internas, descritas a seguir. Adicionalmente o estudo inclui o *feedback* de provedores e clientes de interfaces internas.

QP #1. Qual a frequência de utilização de interfaces internas?

QP #2. Qual a distribuição do uso das interfaces internas de uma biblioteca?

São analisadas 17 bibliotecas e suas APIs internas usadas por clientes, em um *dataset* com aproximadamente 260 mil projetos Java GitHub e 131 milhões de APIs. No melhor do nosso conhecimento, esse é o maior estudo sobre o uso de APIs internas Java. Estudos relacionados concentraram em bibliotecas específicas e analisaram menos clientes [Businge et al. 2015, Businge et al. 2013, Mastrangelo et al. 2015]. Um estudo relacionado ao uso de APIs internas apenas no contexto de *plug-ins* Eclipse, por exemplo, analisa 512 projetos [Businge et al. 2015]. Para melhor entendimento, o estudo proposto nesse artigo concentra-se em dois níveis de granularidade: APIs (por exemplo, quantos clientes estão usando org.junit.internal.AssumptionViolatedException?) e bibliotecas[3] (por exemplo, quantos clientes JUnit estão usando interfaces internas?).

O restante deste artigo está organizado conforme descrito a seguir. A Seção 2 apresenta a metodologia proposta; a Seção 3 os resultados obtidos, e a Seção 4 inclui os comentários de clientes e provedores de APIs internas. Na Seção 5 os riscos à validade são apresentados. A Seção 6 discute trabalhos relacionados e a Seção 7 conclui o estudo.

2. Metodologia

Neste estudo analisam-se 17 bibliotecas Java e o uso das suas interfaces internas em sistemas clientes. Para verificar o uso dessas interfaces, utiliza-se um *dataset* composto por 263.425 projetos e 16.386.193 arquivos Java, minerados através da infraestrutura Boa [Dyer et al. 2013]. Essa infraestrutura inclui diversos *datasets* do GitHub, assim como uma DSL (*Domain Specific Language*) para minerar repositórios de *software* e uma interface Web para executar os scripts de mineração.

Criou-se um *script* Boa para minerar projetos que fazem uso de pelo menos uma interface/biblioteca em um arquivo Java válido. Os metadados obtidos, foram então, in-

[1] www.eclipse.org/articles/article.php?file=Article-API-Use/index.html

[2] www.oracle.com/technetwork/java/faq-sun-packages-142232.html

[3] Neste trabalho, o termo "biblioteca" é utilizado para designar tanto *frameworks* quanto bibliotecas.

seridos em um banco de dados onde criou-se um *script* para agrupar as interfaces encontradas e contabilizar seus sistemas clientes. Em seguida minerou-se as interfaces internas. Para a biblioteca JDK, avalia-se interfaces que possuem o prefixo sun.* . Para as demais bibliotecas considera-se o respectivo prefixo e o pacote internal. A Tabela 1 apresenta a descrição das bibliotecas analisadas e o total de sistemas clientes encontrados.

3. Resultados

QP #1: Qual a frequência de utilização de interfaces internas?

Nessa primeira questão de pesquisa estuda-se a frequência com que clientes usam interfaces internas. A Tabela 1 apresenta o total de clientes de cada biblioteca que usam pelo menos uma interface interna. As bibliotecas Action Bar Sherlock e Eclipse apresentam cerca de 20% dos clientes usando interfaces internas. Entre 15% e 5%, encontram-se as bibliotecas Facebook, OpenQA, Twitter4j, Apache Maven e Google Inject. As demais bibliotecas possuem menos de 5% dos clientes usando interfaces internas.

Tabela 1. Clientes Usando Interfaces Internas

Bibliotecas	**Descrição**	**Total Clientes**	**Interfaces Internas**	
			Total	**%**
Action Bar Sherlock	Criação de barra de ação em aplicativos Android	2.729	810	29.68
Eclipse	Integração e extensão com a plataforma Eclipse	9.774	2.304	23.57
Facebook	Integração de aplicativos com o Facebook	1.253	184	14.68
OpenQA	Desenvolvimento de testes	2.497	206	8.25
Twitter4j	Integração com serviços do Twitter	1.583	106	6.70
Apache Maven	Gerenciamento de projetos e compressão de software	2.304	145	6.29
Google Inject	Injeção de dependências	3.808	216	5.67
Mockito	Desenvolvimento de testes	7.393	331	4.48
Easy Mock	Desenvolvimento de testes	2.260	92	4.07
Jboss	Integração com servidores de aplicação JBoss	4.729	190	4.02
Hibernate	Persistência de dados	11.993	331	2.76
Android	Desenvolvimento de aplicativos Android	68.116	1.816	2.67
JDK	Desenvolvimento de aplicativos e componentes Java	237.898	5.838	2.45
Google Gson	Conversão de objetos JSON	7.906	147	1.86
Apache Cordova	Desenvolvimento de aplicativos móveis	3.006	39	1.30
JUnit	Desenvolvimento de testes	67.677	774	1.14
Hamcrest	Desenvolvimento de expressões em testes	8.219	50	0.61

Para algumas bibliotecas o baixo percentual de uso das interfaces internas não implica necessariamente em menos projetos. A Figura 1 apresenta as 5 bibliotecas com o maior número de clientes usando interfaces internas. A biblioteca JUnit, por exemplo, ocupa a penúltima posição considerando o percentual de clientes (1.14%), entretanto é a 5ª biblioteca (774 projetos) com o maior número de clientes usando interfaces internas.

QP #2: Qual a distribuição do uso das interfaces internas de uma biblioteca?

Nessa segunda questão de pesquisa analisa-se até a 50ª interface interna mais usada por biblioteca. A Figura 2 apresenta a distribuição das interfaces mais usadas, para as bibliotecas com o maior percentual de clientes usando interfaces internas. Já a Figura 3 apresenta a distribuição considerando todos os projetos. É mostrado um *box plot* e um *violin plot* (em amarelo), a fim de melhor apresentar a distribuição. O número em

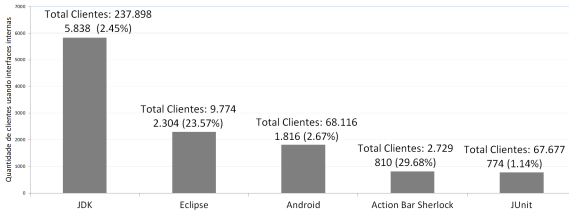
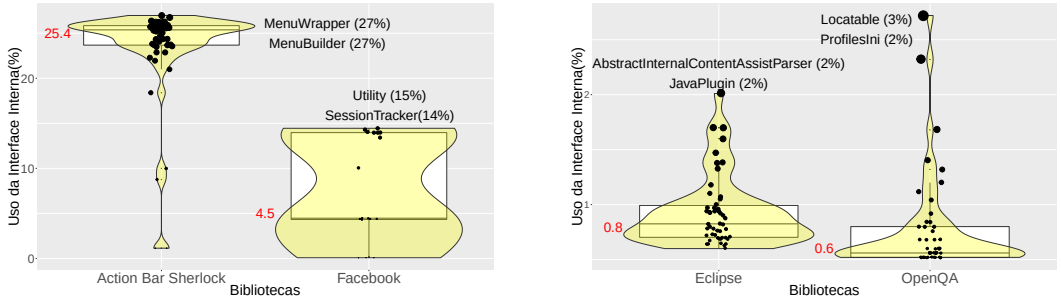


Figura 1. Bibliotecas com mais Clientes Usando Interfaces Internas

vermelho corresponde à mediana; cada ponto traçado corresponde a uma interface, e o seu tamanho é proporcional ao seu uso (quanto maior o ponto, mais usada é a interface).



(a) Action Bar Sherlock e Facebook

(b) Eclipse e OpenQA

Figura 2. Distribuição das Interfaces Internas por Biblioteca

A biblioteca Action Bar Sherlock possui 10 interfaces usadas por mais de 26% dos seus clientes, sendo que com.actionbarsherlock.internal.view.menu.MenuWrapper e com.actionbarsherlock.internal.view.menu.MenuBuilder são usadas por 27% dos clientes. A biblioteca Facebook possui 18 interfaces internas usadas em sistemas clientes; acima de 14% encontram-se três interfaces, com.facebook.internal.Utility (15%, 181 projetos), com.facebook.internal.SessionTracker (14%, 179 projetos), com.facebook.internal.SessionAuthorizationType (14%, 176 projetos). A biblioteca Eclipse não possui interfaces internas com uso acima de 3%. Sua interface interna mais popular é org.eclipse.xtext.ui.editor.contentassist.antlr.internal.AbstractInternalContentAssistParser (196 projetos, 2%). A biblioteca OpenQA também não possui interfaces internas com uso acima de 3%, suas interfaces internas mais usadas pertencem ao pacote selenium (org.openqa.selenium.internal.Locatable (3%) e org.openqa.selenium.firefox.internal.ProfilesIni (2%)). As bibliotecas Google Gson, JDK, Mockito, Hibernate, e Hamcrest não possuem interfaces internas com uso acima de 1%. As bibliotecas Apache Cordova, Google Inject, Android, e JBoss não possuem interfaces internas usadas por mais de 2% dos seus clientes. Por fim, as bibliotecas Apache Maven, Twitter4j, e Easy Mock não tem interfaces internas usadas por mais de 5% dos clientes.

As 50 interfaces internas mais usadas (Figura 3) pertencem às bibliotecas JDK, Action Bar Sherlock e Android. As duas mais usadas pertencem ao JDK, sun.misc.BASE64Encoder (1.502 projetos, 0.57%) e sun.misc.BASE64Decoder (798 projetos, 0.3%). De fato, o uso de interfaces internas do JDK é bem crítico e perigoso para clientes, e tem sido objeto de estudo da literatura recente [Mastrangelo et al. 2015]. A terceira interface é com.actionbarsherlock.internal.view.menu.MenuWrapper (736 projetos, 0.28%). As interfaces Android ocupam a 27ª posição, com.android.internal.telephony.Phone (694 projetos, 0.26%) e a 49ª posição, com.android.internal.telephony.TelephonyIntents (586 projetos, 0.22%).

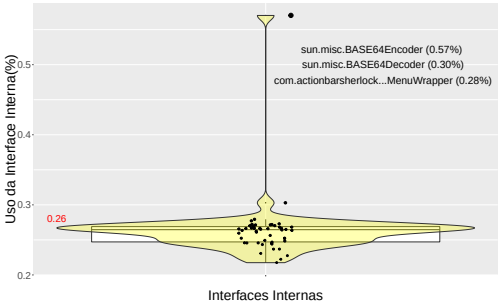


Figura 3. Distribuição das Interfaces Internas

O uso de interfaces internas é um indicativo para os provedores sobre quais elementos internos estão oferecendo recursos necessários para os seus clientes e que provavelmente não são disponibilizados por meio das interfaces públicas.

4. Comentários de Provedores e Clientes de Interfaces Internas

4.1. Consulta aos Clientes de Interfaces Internas

Nessa consulta, verifica-se o conhecimento dos clientes sobre as interfaces internas usadas em seus sistemas. Para tanto, foram consultados 10 desenvolvedores ativos de sistemas clientes cadastrados no GitHub. Para cada desenvolvedor criou-se uma *issue* com as perguntas descritas a seguir.

1. *You used an internal interface in "*name[1]*" class of your project. Do you know that it is an internal interface of the "*name[2]*" library?*
2. *Are you aware that internal interfaces are unstable and may change without backward-compatibility?*

Obteve-se 4 respostas (40%). O primeiro desenvolvedor usou a interface com.android.internal.telephony.Phone em seu projeto, e alegou que era um código-fonte de terceiros. Provavelmente esse desenvolvedor copiou o código-fonte da biblioteca para o seu projeto. O segundo desenvolvedor utilizou a interface interna org.hamcrest.internal.ArrayIterator, e justificou o seu uso devido a uma sugestão da IDE utilizada. O terceiro desenvolvedor utilizou a interface sun.misc.BASE64Encoder. Em sua resposta, ele informou que não sabia que tratava-se de uma interface interna do JDK. O texto a seguir apresenta um trecho da sua resposta.

"I was unaware of how unstable internal interfaces were. From looking into this I have noticed that Oracle have released a Base64 encoder/decoder in the java.util package, so I am now using that instead."

Por fim, o quarto desenvolvedor utilizou a interface com.google.inject.internal.Lists. Em sua resposta ele alegou que tinha conhecimento sobre interfaces internas mas não percebeu que estava utilizando. O texto a seguir apresenta um trecho da suas resposta.

"1. I had not noticed it is an internal. 2. Yes. Now that you have pointed it out, I will probably change it."

Assim, observa-se que o uso de interfaces internas está relacionado com o pouco conhecimento da biblioteca utilizada.

[1]Nome da classe onde a interface interna encontra-se.
[2]Nome da biblioteca. Exemplo: Google Gson.

4.2. Consulta aos Provedores de Interfaces Internas

Nessa consulta, tem-se por objetivo verificar o conhecimento dos provedores sobre os clientes que estão usando suas interfaces internas. Foram criadas *issues* em repositórios GitHub de 4 bibliotecas e obteve-se 3 respostas (75%). Para cada provedor foram realizadas as duas perguntas descritas a seguir. A *issue* também incluiu a lista de interfaces internas mineradas nesse estudo e a quantidade de usuários utilizando as mesmas.

1. *Did you know that internal interfaces are used by clients?*
2. *From the presented interfaces, there is some interface that is a candidate to be promoted to the public one?*

JUnit.[1] Quatro desenvolvedores da biblioteca JUnit responderam a *issue* que foi aberta, sendo um deles um membro *core* do projeto. Obteve-se seis respostas. Dentre os principais pontos discutidos, destaca-se a atualização da biblioteca JUnit a partir da quinta versão, onde o pacote internal será substituído pela notação @API(Internal) e possíveis intervenções para inibir o uso de interfaces internas. Um dos desenvolvedores informou que a equipe está ciente do uso de interfaces internas por clientes, e que essa é uma das razões para essa nova estratégia. O texto a seguir apresenta um trecho da sua resposta.

"Yes, we are aware that many of the originally internal interfaces are used by clients. It's one of the reasons why we're using a different approach of annotating APIs instead of using package names for JUnit 5."

Esse desenvolvedor alegou que cinco interfaces internas da lista apresentada já foram promovidas para o público, entretanto as interfaces foram depreciadas para manter a compatibilidade com versões anteriores. Essa promoção de interfaces internas é objeto de estudo na literatura recente [Hora et al. 2016].

Mockito.[2] Dois desenvolvedores da biblioteca Mockito responderam a *issue* que foi aberta. Obteve-se duas respostas. O primeiro desenvolvedor surpreendeu-se com o uso das interfaces internas por seus clientes, e comentou sobre prováveis ajustes realizados nesses projetos para usar essas interfaces. Adicionalmente, comentou sobre uma possível falta de recursos na biblioteca. O segundo desenvolvedor alegou que algumas funcionalidades deveriam ser adquiridas através de interfaces públicas. Por exemplo, a interface interna org.mockito.internal.verification.Times deve ser acessada via interface pública Mockito.times(). O mesmo informou que já ocorreram discussões sobre o uso de interfaces internas entre os membros da equipe e que realmente considera tal uso uma má pratica de programação. O texto a seguir apresenta um trecho da sua resposta.

"*It seems that quite some internal classes are used instead of relying on the implementation provided by the Mockito.* method. E.g. Times should be obtained via Mockito.times(). I am not sure if we can do something about that.*"

Google Gson.[3] Um dos desenvolvedores da biblioteca Google Gson informou que não tinha conhecimento da quantidade de clientes usando as interfaces internas, entretanto não surpreendeu-se com essa má prática adotada por alguns desenvolvedores clientes. Segundo ele, nenhuma das interfaces listadas por nós na *issue* são candidatas a promoção para o público. O texto a seguir apresenta um trecho da suas resposta.

[1] https://github.com/junit-team/junit5/issues/305
[2] https://github.com/mockito/mockito/issues/428
[3] https://github.com/google/gson/issues/874

"I didn't quite know this, but am not very surprised. None of these are candidates for promotion to public APIs."

Assim, após a consulta aos desenvolvedores de sistemas provedores de interfaces internas, observa-se que essa má prática é conhecida. As respostas dos desenvolvedores JUnit levantam também uma discussão em torno de outra atividade crítica, além do uso de interfaces internas, sistemas clientes estão usando interfaces internas depreciadas.

5. Riscos à validade

Validade externa. Os resultados desse estudo estão restritos a projetos Java GitHub, ou seja, eles não podem ser generalizados para outras linguagens e outros repositórios de código-fonte. Além disso, dentre os 260 mil sistemas analisados, alguns podem ser repositórios GitHub das bibliotecas analisadas nesse trabalho.

Validade interna. Projetos que importaram somente a biblioteca não foram considerados clientes das interfaces, já que a importação de uma biblioteca não assegura o uso de todas as suas interfaces no código-fonte.[1]

Validade de construção. O script Boa recuperou apenas interfaces/bibliotecas que foram importadas no código-fonte, e algumas dessas importações podem ser classes do próprio projeto. Além disso, pode-se ter interfaces que não foram utilizadas (*warning*), e projetos que copiaram o código-fonte da biblioteca.

6. Trabalhos Relacionados

Existem estudos que concentram-se na evolução e uso de interfaces e bibliotecas [Hora and Valente 2015, Mileva et al. 2010, McDonnell et al. 2013]. Adicionalmente alguns trabalhos relatam o uso indevido de interfaces por sistemas clientes, como por exemplo, a má prática de programação relacionada ao uso de interfaces internas [Businge et al. 2015, Mastrangelo et al. 2015]. Outros estudos concentram-se na promoção dessas interfaces internas para o público [Hora et al. 2016].

Um estudo relacionado ao uso de interfaces internas da biblioteca Eclipse serviu de inspiração para esse trabalho [Businge et al. 2015, Businge et al. 2013]. O mesmo analisa *releases* de 512 *plug-ins* Eclipse do repositório SourceForge, com o objetivo de investigar motivos que levam ao uso (ou desuso) de interfaces internas. Os autores também realizam entrevistas com alguns clientes. Outro estudo nessa área concentra-se no uso da interface interna sun.misc.Unsafe provida pelo JDK [Mastrangelo et al. 2015]. O artigo identifica 14 razões que levam clientes a usarem essa interface e analisa cerca de 86 mil arquivos Java. Ao contrário dos trabalhos mencionados, que concentram-se em bibliotecas específicas, este trabalho analisa o uso de interfaces internas na linguagem Java, e o conhecimento de clientes e provedores sobre o uso dessas interfaces. Além disso o trabalho faz uso de um *dataset* maior, com aproximadamente 260 mil projetos, 131 milhões de APIs e 16 milhões de arquivos.

7. Conclusões

Este trabalho apresentou um estudo empírico em larga escala sobre a utilização de interfaces internas em 260 mil projetos Java. Na primeira questão de pesquisa, investigou-se a frequência com que clientes usam interfaces internas. Na segunda questão de pesquisa

[1]Por exemplo, importar a biblioteca java.util.* não implica no uso das suas interfaces ArrayList e List.

analisou-se a distribuição do uso dessas interfaces por biblioteca. Além disso, realizou-se através de uma análise qualitativa, uma consulta aos clientes e provedores de interfaces internas. Apresenta-se a seguir os principais resultados desse estudo:

QP #1. Verificou-se que algumas bibliotecas tem mais de 20% dos seus clientes usando interfaces internas.

QP #2. Verificou-se que algumas interfaces internas das bibliotecas atraem mais clientes. Além disso, observou-se que as 50 interfaces internas mais usadas pertencem as bibliotecas JDK, Action Bar Sherlock e Android.

Entrevista. Na consulta com os provedores de algumas bibliotecas, observou-se que o uso de interfaces internas por sistemas clientes é conhecido, sendo considerada uma má prática de programação pelos desenvolvedores. Entre os clientes consultados, observou-se que o uso de interfaces internas é motivado pela falta de conhecimento da biblioteca.

Como trabalho futuro, pretende-se analisar o total de interfaces internas de cada biblioteca, e quantas dessas interfaces são usadas por clientes.

Agradecimentos: Esta pesquisa é financiada pela FAPEMIG e pelo CNPq. Agradecemos também aos desenvolvedores GitHub consultados.

Referências

Businge, J., Serebrenik, A., and van den Brand, M. G. J. (2013). Analyzing the eclipse API usage: Putting the developer in the loop. In *17th European Conference on Software Maintenance and Reengineering (CSMR)*, pages 37–46.

Businge, J., Serebrenik, A., and van den Brand, M. G. J. (2015). Eclipse API usage: the good and the bad. In *Software Quality Journal*, pages 107–141.

Dyer, R., Nguyen, H. A., Rajan, H., and Nguyen, T. N. (2013). Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *35th International Conference on Software Engineering*, pages 422–431.

Hora, A., Valent, M. T., Robbes, R., and Anquetil, N. (2016). When should internal interfaces be promoted to public? In *24th International Symposium on the Foundations of Software Engineering (FSE)*, pages 1–12.

Hora, A. and Valente, M. T. (2015). apiwave: Keeping track of API popularity and migration. In *31st IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 321–323.

Mastrangelo, L., Ponzanelli, L., Mocci, A., Lanza, M., Hauswirth, M., and Nystrom, N. (2015). Use at your own risk: the Java unsafe API in the wild. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 695–710. ACM.

McDonnell, T., Ray, B., and Kim, M. (2013). An empirical study of API stability and adoption in the android ecosystem. In *International Conference on Software Maintenance*, pages 70–79.

Mileva, Y. M., Dallmeier, V., and Zeller, A. (2010). Mining API popularity. In *International Academic and Industrial Conference on Testing - Practice and Research Techniques*, pages 173–180.