Um Conjunto Validado de Maus Cheiros na Camada de Apresentação de Aplicações Android Nativas a ser apresentado à CPG para a dissertação

Suelen Goularte Carvalho

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Marco Aurélio Gerosa

Área de Concentração: Computação Móvel
Orientador: Prof. Dr. Marco Aurélio Gerosa

São Paulo, Julho de 2016

Um conjunto validade de maus cheiros na aplicação do padrão Model-View-Presenter no domínio de Aplicações Android Nativas a ser apresentado à CPG para dissertação

Esta é a versão original da dissertação elaborada pelo candidato Suelen Goularte Carvalho, tal como submetida a Comissão Julgadora.

Comissão Julgadora:

- Prof. Dr. Marco Aurélio Gerosa – IME-USP
- Prof. Dr. Nome 222 222 – IME-USP
- Prof. Dr. Nome 333 333 – IME-USP

Dedico esta dissertação de mestrado aos meus

...

...

....

Agradecimentos

A fazer.

Resumo

Dois objetivos de desenvolvedores de software são escrever código fácil de ser mantido e evoluído e detectar trechos de códigos problemáticos que podem ser melhorados. Há diversas práticas reconhecidas que nos ajudam nesta tarefa tais como Padrões de Projeto, catálogos de Maus Cheiros dentre outros. De fato estas práticas são excelentes ferramentas, porém cada plataforma tem suas especificidades que poderiam ser melhor trabalhadas com práticas pensadas especificamente para elas. Por exemplo, a plataforma Android possui um diretório chamado *res* ao qual são colocados todos os recursos usados pela aplicação como imagens, *layouts* de telas, textos para internacionalização, estilos dentre outros. Conforme o projeto cresce a quantidade de arquivos nesta pasta aumenta, torna-se difícil a manutenção, reaproveitamento de código dentre outras manutenções comuns para evolução de um aplicativo. Outra situação que se tem é com relação a *Activity*s, pode-se dizer que elas representam as controladoras de telas no Android. Um problema reconhecido com relação a *Activity*s é que elas costumam ser altamente acopladas tanto com *view*s quanto com *modelo*. Cenários como estes são específico da plataforma Android. Até o momento não há pesquisar no sentido de identidicar boas e más práticas em termos de qualidade de software específicos da plataforma Android. Por este motivo esta dissertação pretende pesquisar, validar e catalogar boas e más práticas no desenvolvimento da camada de apresentação de aplicações Android nativas. Escolhemos a camada de apresentação pois é nesta camada onde o Android apresenta suas maiores diferenças quando comparado com o desenvolvimento Java tradicional. Para isso aplicamos um questionário com mais de 2000 desevolvedores Android de forma a identificar boas e más práticas. O resultado do questinátio foi manualmente categorizado de forma a identificar os maus cheiros. Estes maus cheiros definidos foram validados com Android Experts. Por último, realizamos experimentos de forma a validar a percepção dos desenvolvedores com relação aos maus cheiros encontrados. Nossa pesquisa conclui com um catálogo de maus cheiros específicos para a plataforma Android onde poderá ajudar desenvolvedores a tomarem decisões melhores sobre como desenvolver aplicações Android de forma a ter mais qualidade, reduzir a propensão a alteração e defeitos.

Palavras-chave: Aplicações Móveis, Plataforma Android, Maus Cheiros de Código, Qualidade de Código.

Abstract

A fazer.

Keywords: Mobile Applications, Android Platform, Bad Code Smells, Code Quality.

Sumário

Lista de Abreviaturas

MVC *Model View Controller*

Lista de Símbolos

Σ Sistema de transição de estados

Lista de Figuras

Capítulo 1

Introdução

Ao longo da última década o desenvolvimento de aplicativos móveis Android nativo (*aplicativos Android*) atingiu um enorme sucesso [Hec15]. A Google Play Store registra em Fevereiro de 2016 um total de 2 milhões de aplicativos e este crescimento vem acentuando-se ano após ano [App16a]. Desta forma, *aplicativos Android* tem se tornado sistemas de software complexos que devem ser desenvolvidos rapidamente e evoluídos regularmente para se ajustarem aos requisitos de usuários, o que pode levar a escolhas ruins de design de código [Hec15].

Para mitigar a deteorização do código, é comum desenvolvedores se apoiarem em ferramentas de detecção de maus cheiros como PMD [PMD16], Sonarque [Son16] e JDeodoran [TC08] e algumas espcíficas para Android como Páprika [Hec15] e Refactoring [RB13] e boas práticas consolidades de desenvolvimento. De fato, muitas destas ferramentas e prática podem ser aplicados aos projetos Android e colaboram no desenvolvimento de um software com qualidade, fácil manutenção e evolução. Porém, projetos Android trazem desafios adicionais principalmente com relação ao desenvolvimento da camada de apresentação onde sua estrutura mais se difere de projetos tradicionais [Hec15].

Para entendermos estas particularidades, precisamos ter em mente a estrutura de um projeto Android. De forma simplificada, projetos Android possuem um diretório chamado `java` onde ficam as classes java. Um arquivo de configuração chamado `AndroidManifest.xml` que contém configurações tais como tela incial, permissões que serão solicitadas ao usuário, dentre outros. Por último, um diretório chamado `res`, abreviação para o termo *resources*, que contém subpastas com nomes pré-determinados pelo Android onde cada subpasta contém arquivos que desempenham papéis específicos na aplicação, por exemplo, a subpasta

LAYOUT contém arquivos que representam a parte visual de telas do Android, a subpasta DRAWABLE contém os elementos gráficos e assim por diante.

O diretório `res` é uma particularidade de aplicações Android que pode trazer problemas de manutenção e evolução de código pois não existe hoje mapeados o que são boas e más práticas com relação a esta parte de um projeto Android. O que se percebe é que conforme o projeto evoluí e diversos arquivos são adicionados e alterados, se torna difícil encontrar o arquivo correto, ou mesmo inserindo-se pontos de falha críticos ao longo que a complexidade visual de uma tela aumenta. Outra particularidade é com relação ao componente Android ACTIVITY, que representa uma tela no Android [And]. Estas classes costumam precisar de acesso tanto a elementos visuais, chamados de VIEW's, quando ao modelo da aplicação, ou seja, normalmente possuem um alto acoplamento. Estes são apenas alguns dos questionamento possíveis ao se tratar de boas e más práticas, em termos de manutenabilidade de código, na camada de apresentação de projetos Android.

1.1 Motivação

Smartphones existem desde 1993 porém nesta época o foco era voltado para empresas e suas necessidades. Em 2007 houve o lançamento do iPhone fabricado pela Apple, o primeiro smartphone voltado ao público em geral. Ao final do mesmo ano o Google revelou seu sistema operacional para dispositivos móveis, o Android [SS13]. Desde então, esta plataforma vem, ano após ano, registrando um grande crescimento, ultrapassando outras plataformas [App16a] [Gro16] [Ran15].

No entanto, pesquisas em torno desta plataforma não crescem na mesma proporção [MDA+] e a ausência de um catálogo de maus cheiros de código espefícido para a plataforma Android pode resultar em (i) uma carência de conhecimento sobre boas e más práticas a ser compatilhado entre praticantes desta plataforma, (ii) indisponibilidade de ter uma ferramenta de detecção de maus cheiros de forma a alertar automaticamente os desenvolvedores da existência de maus cheiros de código, e (iii) ausência de estudo empírico sobre o ímpacto destas más práticas na manutenabilidade do código de projetos Android, por este motivos, boas e más práticas que são específicos a uma platadforma, no caso Android, tem emergido como tópicos de pesquisa sobre manutenção de código [AGC+16].

1.2 Objetivos

A fazer.

1.3 Abordagem de Solução

A fazer.

1.4 Originalidade e Relevância

Diversas pesquisas em torno de maus cheiros de código veem sendo realizadas ao longo dos últimos anos. Já existem inclusive diversos maus cheiros mapeados, porém poucos deles são específicos da plataforma Android [MDA+]. Segundo [Hec15] estudos sobre maus cheiros de código sobre aplicações Android ainda estão em sua infância, porém ainda assim são muito relevantes inclusive notando-se que é mais comum identificar em aplicativos Android maus cheiros mapeados exclusivamente para esta plataforma do que os maus cheiros tradicionais [LVKM+].

1.5 Organização do Trabalho

A fazer.

Capítulo 2

Fundamentação Teórica

Para a compreensão deste trabalho é importante ter claro a definição de 4 itens, são eles: *Code Smells*, *Model View Controller*, padrões e desenvolvimento Android.

2.1 Aplicativos Móveis

Aplicações móveis são *softwares* aplicativos que são executados em dispositivos como *smartphones*, *tablets*, *smartwatchs* e mais recentemente outros dispositivos como carros inteligentes e *smart TVs* [Ver13] [And16] [And14]. Este último apesar de não ser um dispositivo móvel, nos últimos anos passou a suportar receber aplicações que inicialmente foram desenvolvidas para dispositivos móveis. Aplicativos são instalados nos dispositivos através de lojas de aplicativos onde os usuários podem escolher quais aplicativos desejam instalar.

Este dispositivos rodam sistemas operacionais feitos especificamente para eles [Per16]. Sistemas operacionais para dispositivos móveis são frequentemente chamados de OS móvel ou *smartphone* OS ou mesmo plataformas móveis. Nesta dissertação usaremos o termo **Plataforma Móvel** daqui em diante para nos referir a ele.

Conforme podemos observar na Figura 2.1, até início de 2010 as plataformas móveis dominantes eram, da mais dominante para a menos, Symbian, RIM e iOS. No final de 2010 em diante a plataforma móvel Android ultrapassou sua maior concorrente Symbian em dis-

positivos vendidos a usuários finais e dali em diante até os dias atuais vem aumentando este número [Mob16], estando hoje com 84% da fatia de mercado contra 14% de sua maior concorrente, o iOS.

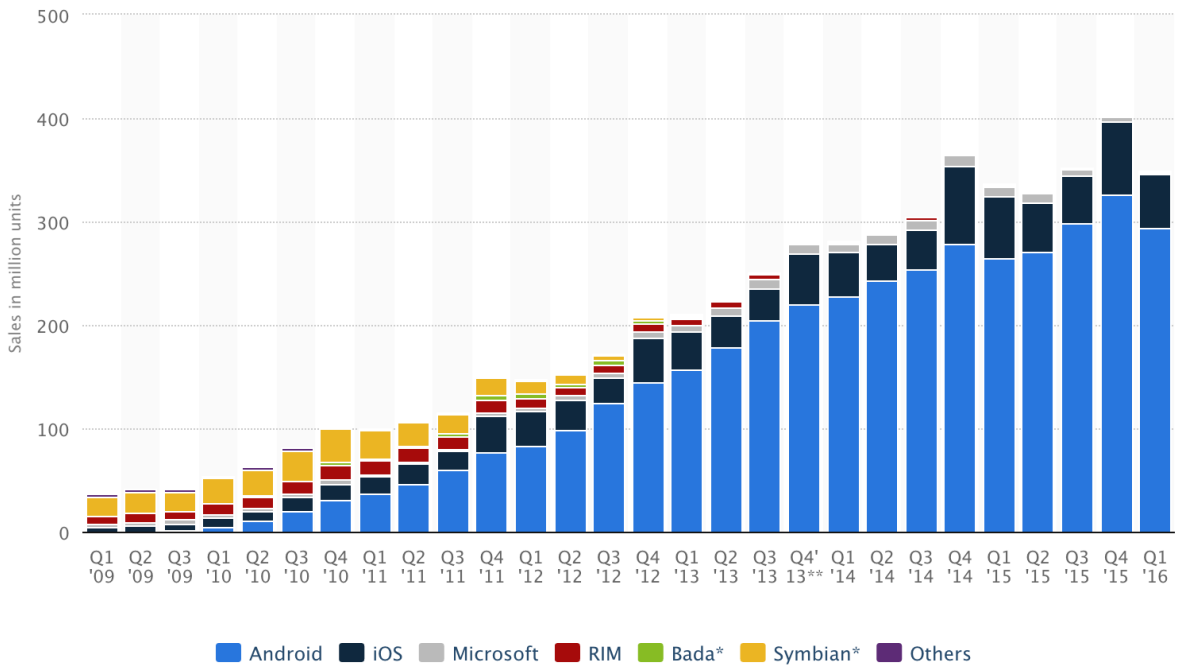

Figura 2.1: *Divisão global de plataformas móveis em vendas para usuários finais do 1ž quadrimestre de 2009 ao 1ž quadrimestre de 2016 [Per16].*

Desta forma, as principais plataformas móveis no momento são Android e iOS respectivamente e as projeções até 2020 mostram que o Android continuará na liderança [Gro16].

Existem diversas lojas de aplicativos, dentre elas, existem as oficiais, ou seja, as disponibilizadas pelas fabricantes de cada plataforma. Para Android temos a Google Play Store e para iOS temos a Apple App Store. Na Figura 2.3 e Figura 2.2 podemos observar o aumento da quantidade de aplicativos da Google Play Store e da Apple App Store respectivamente.

A quantidade de aplicativos disponível nas lojas e a quantidade de instalações realizadas por usuários são fortes métricas para avaliar o sucesso de uma loja de aplicativos. Segundo o *Wall Street Journal*, no começo de 2015, a Google Play Store tinha mais de 70% de downloads de aplicativos que sua principal concorrênte, a Apple App Store [WSJ15]. É interessante ressaltar que existe uma terceira métrica de lojas de aplicativos que diz com relação a receita gerada, e conforme o mesmo texto do *Wall Street Journal* a Apple App Store gera

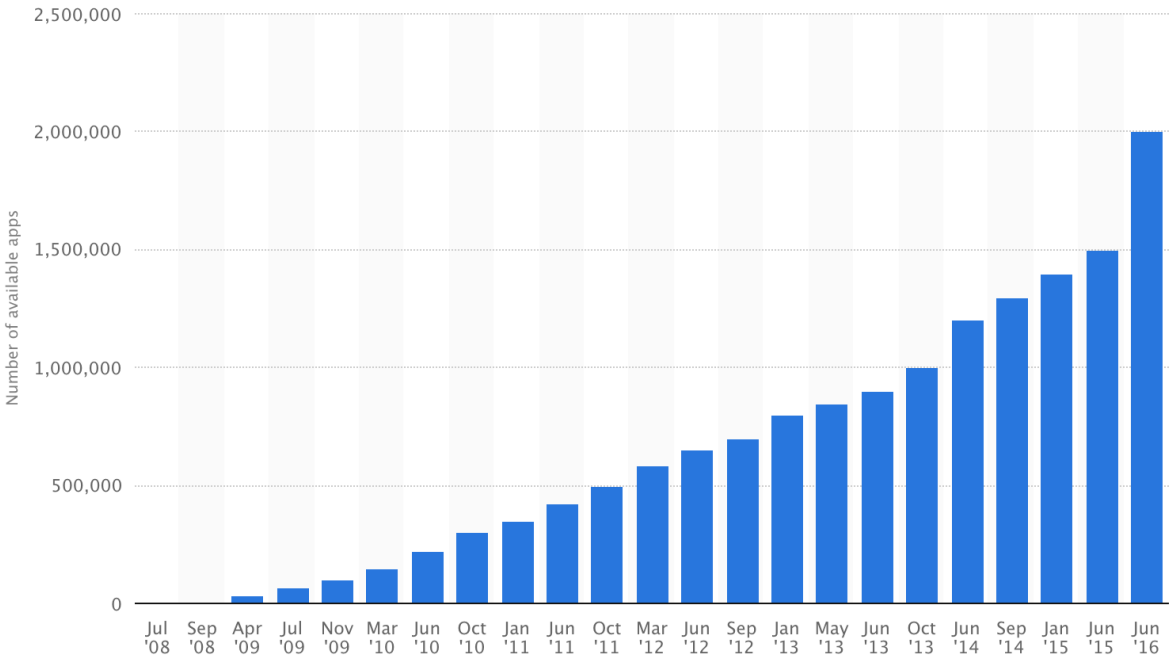

Figura 2.2: *Número de aplicações disponíveis na Apple App Store de Julho de 2008 a Junho de 2016 [App16b].*

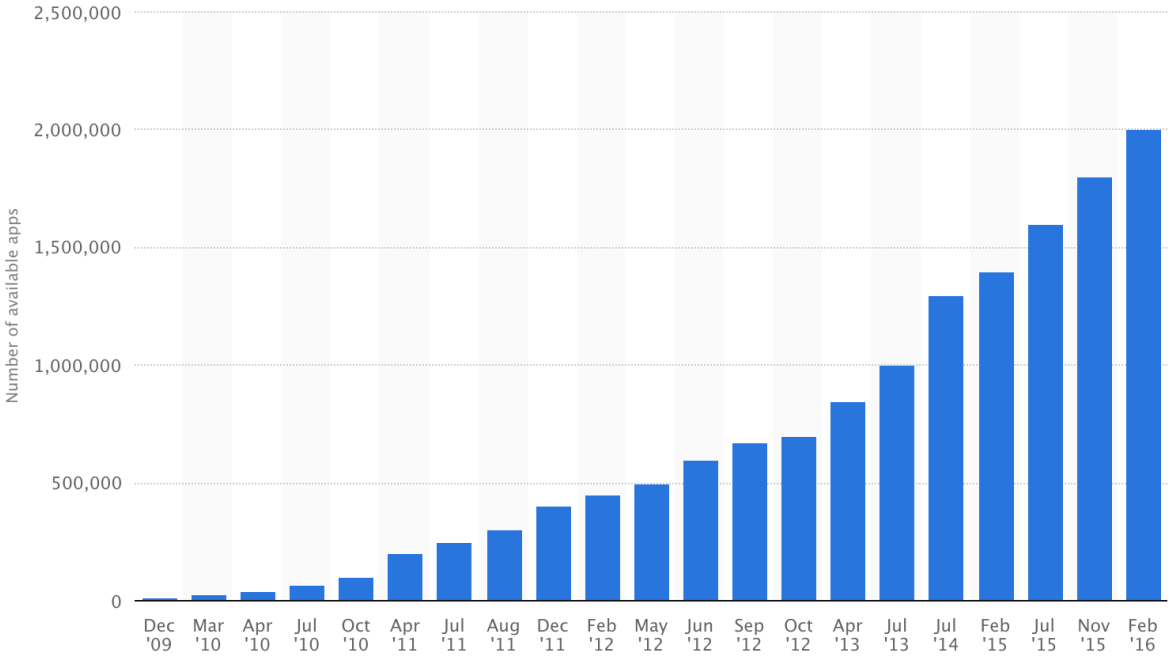

Figura 2.3: *Número de aplicações disponíveis na Google Play Store de Dezembro de 2009 a Fevereiro de 2016 [App16a].*

70% mais receita do que a loja da Google.

2.2 Maus Cheiros de Código

Mau cheiro de código é uma indicação superficial que usualmente corresponde a um problema mais profundo em um software. Por si só um *code smell*, seu termo em inglês, não é algo ruim, ocorre que frequentemente ele indica um problema mas não necessáriamente é o problema em si [Fow06]. O termo em inglês *code smell* foi cunhado pela primeira vez por Kent Beck enquanto ajudava Martin Fowler com o seu livro Refactoring [Fow99] [Fow06].

Code Smells são padrões de código que estão associados com um design ruim e más práticas de programação. Diferentemente de erros de código eles não resultam em comportamentos erroneos. *Code Smells* apontam para áreas na aplicação que podem se beneficiar de refatorações. [Ver13]. Refatoração é definido por "uma técnica para reestruturação de um código existente, alterando sua estrutura interna sem alterar seu comportamento externo" [Fow99].

Escolher não resolver *code smells* pela refatoração não resultará na aplicação falhar mas irá aumentar a dificuldade de mantê-la. Logo, a refatoração ajuda a melhorar a manutenabilidade de uma aplicação [Ver13]. Uma vez que os custos com manutenção são a maior parte dos custos envolvidos no ciclo de desenvolvimento de software [Tsa10], aumentar a manutenabilidade através de refatoração irá reduzir os custos de um software no longo prazo.

2.3 Model View Controller

A fazer.

2.4 Padrões

A fazer.

2.4.1 Formato de Padrões

A fazer.

2.4.2 Formato Alexandrino

A fazer.

2.4.3 Formato Adotado

A fazer.

Capítulo 3

Pesquisa

Este trabalho trata-se de uma pesquisa descritiva e exploratória. Descritiva porque este tipo de pesquisa visa observar, analisar, registrar e correlacionar os fatos ou fenômenos sem manipulá-los [Tog01] e exploratória porque são abstraídas práticas recorrentes objetivando descever sua natureza [Mal04] na forma de *code smells* se utilizando do formato de padrões.

TODO: Confirmar se faz sentido ser exploratória ou se faz mais sentido ser experimental.

3.1 Hipóteses

A fazer.

3.2 Processo de Pesquisa

A fazer.

3.2.1 Coleta de Dados

A fazer.

3.2.2 Análise dos Dados

A fazer.

3.3 Escrita dos Code Smells

A fazer.

Capítulo 4

Catálogo de Code Smells

A fazer.

4.1 Especialize sua Tela

Estou supondo que este possa ser um code smell. A ideia é identificar que uma activity está com muitas responsabilidades/ações de usuário (vendo a quantidade de listeners em uma mesma activity) e a solução seria quebrar em vários fragments, de forma a aumentar a coesão e componentizar para reaproveitar em disposivos de tamanhos de telas diferentes.

4.2 Code Smell 2

A fazer.

4.3 Code Smell 3

A fazer.

4.4 Code Smell 4

A fazer.

4.5 Code Smell 5

A fazer.

Capítulo 5

Conclusão

A fazer.

5.1 Principais contribuições

A fazer.

5.2 Trabalhos futuros

A fazer.

Referências Bibliográficas

[AGC+16] MaurÃŋcio Aniche, Bavota G., Treude C., Van Deursen A., and Gerosa M. A validated set of smells in model-view-controller architectures. 2016. 2

[And] Activities. https://developer.android.com/guide/components/activities.html. Last accessed at 29/08/2016. 2

[And14] Google android software spreading to cars, watches, tv. http://phys.org/news/2014-06-google-android-software-cars-tv.html, June 2014. Last accessed at 26/07/2016. 4

[And16] Ford terá apple carplay e android auto em todos os modelos nos eua. http://g1.globo.com/carros/noticia/2016/07/ford-tera-apple-carplay-e-android-auto-em-todos-os-modelos-nos-eua.html, 2016. Last accessed at 26/07/2016. 4

[App16a] Number of available applications in the google play store from december 2009 to february 2016. http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2016. Last accessed at 24/07/2016. viii, 1, 2, 6

[App16b] Number of available apps in the apple app store from july 2008 to june 2016. http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/, 2016. Last accessed at 24/07/2016. viii, 6

[Fow99] Martin Fowler. *Refactoring. Improving the Design of Existing Code.* Addison-Wesley, 1999. 7

[Fow06] Martin Fowler. Code smell. http://martinfowler.com/bliki/CodeSmell.html, February 2006. 7

[Gro16] Worldwide smartphone growth forecast to slow to 3.1% in 2016 as focus shifts to device lifecycles, according to idc. http://www.idc.com/getdoc.jsp?containerId=prUS41425416, June 2016. Last accessed at 23/07/2016. 2, 5

[Hec15] Geoffrey Hecht. An approach to detect android antipatterns. page 766âĂŞ768, 2015. 1, 3

[LVKM+] Mario Linares-VÃąsquez, Sam Klock, Collin Mcmillan, Aminata SabanÃľ, Denys Poshyvanyk, and Yann-GaÃńl GuÃľhÃľneuc. Domain matters: Bringing further evidence of the relationships among anti-patterns, application domains, and quality-related metrics in java mobile apps. 3

[Mal04] Márcia R. T. L. Malheiros. O processo de pesquisa na graduação. http://www.profwillian.com/_diversos/download/prof/marciarita/pesquisa_na_graduacao.pdf, 2004. Last accessed at 20/07/2016. 9

[MDA+] Umme Mannan, Danny Dig, Iftekhar Ahmed, Carlos Jensen, Rana Abdullah, and M Almurshed. Understanding code smells in android applications. 2, 3

[Mob16] Gartner says worldwide smartphone sales grew 3.9 percent in first quarter of 2016. http://www.gartner.com/newsroom/id/3323017, May 2016. Last accessed at 23/07/2016. 5

[Per16] Global mobile os market share in sales to end users from 1st quarter 2009 to 1st quarter 2016). http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, 2016. Last accessed at 24/07/2016. viii, 4, 5

[PMD16] Pmd (2016). https://pmd.github.io/, 2016. Last accessed at 29/08/2016. 1

[Ran15] Steve Ranger. ios versus android. apple app store versus google play: Here comes the next battle in the app wars. http://www.zdnet.com/article/ios-versus-android-apple-app-store-versus-google-play-here-comes-the-next-battle-in-the-app-w January 2015. Last accessed at 24/07/2016. 2

[RB13] Jan Reimann and Martin Brylski. A tool-supported quality smell catalogue for android developers. 2013. 1

[Son16] Sonarqube (2016). http://www.sonarqube.org/, 2016. Last accessed at 29/08/2016. 1

[SS13] Muhammad Sarwar and Tariq R. Soomro. Impact of smartphone's on society. *European Journal of Scientific Research*, 98(2):216–226, March 2013. http://www.europeanjournalofscientificresearch.com. 2

[TC08] N Tsantalis and T Chaikalis. Jdeodorant: Identification and removal of type-checking bad smells. 2008. 1

[Tog01] Marco A. Togatlian. Tipos de pesquisa. http://www.togatlian.pro.br/docs/pos/unesa/tipos.pdf, 2001. Last accessed at 20/07/2016. 9

[Tsa10] Nikolaos Tsantalis. *Evaluation and Improvement of Software Architecture: Identification of Design Problems in Object-Oriented Systems and Resolution through Refactorings*. PhD thesis, University of Macedonia, August 2010. 7

[Ver13] Daniël Verloop. *Code Smells in the Mobile Applications Domain*. PhD thesis, TU Delft, Delft University of Technology, 2013. 4, 7

[WSJ15] AppleâĂŹs app store widens revenue lead over google play âĂŞ report. http://blogs.wsj.com/digits/2015/04/15/apples-app-store-widens-revenue-lead-over-google-play-report/, April 2015. Last accessed at 26/07/2016. 5