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

Na última década, a quantidade de aplicativos Android tem crescido de forma acentuada. Estes tem se tornado complexos projetos de software que precisam ser rapidamente desenvolvidos e regularmente evoluídos para atender aos requisitos dos usuários. Entretando, este contexto, pode levar a decisões ruins de *design* de código conhecidas como *antipatterns* que podem degradar a qualidade do projeto, tornando-o de difícil manutenção. Ferramentas de detecção automática de maus cheiros de código ajudam desenvolvedores a identificar estes trechos ruins para serem refatorados. No entanto, projetos Android possuem especificadades que os diferenciam de projetos orientados a objetos tradicionais e apesar de já existirem diversos maus cheiros de código catalogados, pesquisas neste sentido específicas para projetos Android, ainda estão em sua infância e ainda é considerado um problema em aberto. Nossa pesquisa pretende identificar, validar e documentar maus cheiros de código específicos da camada de apresentação de projetos Android. Na camada de apresentação é onde encontramos diferenças relevantes com relação a projetos orientados a objetos tradicionais. Ao contrário de outras pesquisas que catalogaram maus cheiros específicos para Android, pretendemos focar em boas e más práticas que influênciem na qualidade do código. Ao final, teremos um catálogo validado de maus cheiros na camada de apresentação de projetos Android nativos relacionados a legibilidade e manutenabilidade. Com isso será possível a implementação de ferramentas de detecção automática destes maus cheiros, auxiliando desenvolvedores, no desenvolvimento de projetos Android com mais qualidade, aumentando legibilidade e manutenabilidade.

Palavras-chave: android, maus cheiros, qualidade de código, engenharia de software, manutenção de software, métricas de código.

Abstract

Keywords: android, software engineering, code smells, software maintance, code metrics, code quality.

Sumário

Lista de Abreviaturas

SDK *Software Development Kit*

IDE *Integrated Development Environment*

APK *Android Package*

ART *Android RunTime*

Lista de Símbolos

Σ Sistema de transição de estados

Lista de Figuras

Capítulo 1

Introdução

Identificar o contexto e a motivação para o trabalho, especificar o problema, discutir trabalhos relacionados (principalmente as suas limitações), definir as contribuições, enfatizar os resultados principais e a organização do texto.

[CONTEXTO + MOTIVAÇÃO] 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 [App16b]. Assim, 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 boas práticas reconhecidas de desenvolvimento de software e ferramentas de detecção de maus cheiros como PMD [PMD16], Sonarque [Son16] e JDeodoran [TC08] e Páprika [Hec15] e Refactoring [RB13] mais específicas para Android. Estas ferramentas geram relatórios indicando códigos bons e ruins.

[PROBLEMA] De fato, muitas destas ferramentas e práticas podem ser aplicados aos projetos Android e colaboram no desenvolvimento de um software com qualidade, legível e de fácil manutenção. No entanto, projetos Android trazem desafios adicionais principalmente com relação ao desenvolvimento da camada de apresentação onde sua estrutura mais se difere de projetos tradicionais [Hec15].

A camada de apresentação do Android possui uma estrutura bem específica, combinando

código Java com arquivos `xml` que representam dos mais variados recursos como LAYOUT's, elementos gráficos, arquivos de internacionalização, dentre outros. Esta combinação torna o Android diferente de projetos tradicionais. E intensifica a necessidade de pesquisas focadas específicas para guiar o desenvolvimento de forma a mitigar a deteorização.

O diretório `res` é uma particularidade de aplicações Android pois não o vemos em projetos tradicionais. Desafios de manutenabilidade que podemos encontrar são em termos de organização deste diretório, implementação dos arquivos nele contido, nomenclaturas, dentre outros. Outra particularidade é com relação ao componente Android ACTIVITY, que representa uma tela [Anda]. Estas classes respondem as interações do usuário com a tela, sempre estão vinculadas a um LAYOUT e normalmente precisam de acesso a classes do modelo da aplicação, sendo assim, altamente acopladas.

[RELACIONADOS] Atualmente já existe alguns maus cheiros especificos para Android catalogados. Num excelente trabalho desempenhado por FULANO, ele cataloga 30 maus cheiros com relação a utilização de recursos e experiência do usuário [Hec15], o que se difere da nossa pesquisa que busca identificar maus cheiros em termos de qualidade, legibilidade e manutenabilidade de código.

Ainda com relação a estudos já realizados, temos...

[CONTRIBUIÇÕES] Nossa pesquisa busca identificar, através de questionários aplicados na comunidade de desenvolvimento Android, boas e más práticas para o desenvolvimento da camada de apresentação de projetos Android. Os dados obtidos com o questionário serão categorizados e extraídos maus cheiros de código. De forma a validar que os maus cheiros extraídos fazem sentido, iremos primeiramente validá-los com especialistas Android e depois realizar um experimento com desenvolvedores Android de forma a captar a percepção deles com relação aos maus cheiros definidos.

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 [App16b] [Gro16] [**?**].

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 Questões da Pesquisa

A questão primária desta pesquisa é **Quais são as boas e más práticas específicas à camada de apresentação em aplicações Android nativas?**. Para responder a esta questão, temos as seguintes questões secundárias:

- **Q1: Quais são as classes que representam a camada de apresentação no Android?**

  Visto que não há registros de estudos que delimitam quais arquivos e tipos de classes representam a camada de apresentação no Android, o objetivo desta prmeira questão é, através de um questionário, definir, em termos desta pesquisa, quais elementos de um projeto Android representam a camada de apresentação.

- **Q2: Existe maus cheiros específicos para a camada de apresentação do Android?**

  Após delimitar exatamente quais elementos representam a camada de apresentação, pretende-se coletar, novamente através de um questionário, boas e más práticas utilziadas e/ou percebidas por desenvolvedores em cada elemento da camada de apresentação. Por exemplo, certamente aquivos do tipo LAYOUT e ACTIVITIES são fortes candidatos a serem parte da camada de apresentação do Android. Desta forma, para cada elemento identificado, serão feitas as seguintes perguntas:

  - Você conhece ou utiliza alguma boa prática para lidar com X?
  - Você considera algo uma má prática para lidar com X?

Onde "X" indica o elemento parte da camada de apresentação do Android. Estes dados serão analisados, categorizados e extraídos maus cheiros. Estes maus cheiros serão validados com especialistas em desenvolvimento Android.

- **Q3: Desenvolvedores percebem as classes afetadas pelos maus cheiros prpopostos como problemáticas?**

  Após a definição de um catálogo de maus cheiros, pretende-se validar se desenvolvedores os percebem de fato como indicativos de trechos de códigos ruins. Para isso pretende-se conduzir um experimento. [Completar aqui]

  Esta questão contempla duas subquestões sobre a relação dos maus cheiros e a tendência a mudanças e introdução de *bugs* no projeto.

  - **Q3.1: Qual a relação entre os maus cheiros propostos e o tendência a mudanças de classes?**
  - **Q3.2: Qual a relação entre os maus cheiros propostos e o tendência a introdução de *bugs* nas classes?**

1.3 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+].

Capítulo 2

Fundamentação Teórica

Para a compreensão deste trabalho é importante ter claro a definição de 2 itens, são eles: *Aplicações Android* e *Maus Cheiros de Código*.

2.1 Android

Android é um sistema operacional móvel de código aberto, baseado no kernel do Linux, desenvolvido pelo Google para dispositivos móveis como *smartphones* e *tablets*. Atualmente é possível encontrá-lo sendo executado em outros dispositivos como painéis de carros, *smart TVs* e *smartwatchs*.

O primeiro *smartphone* com Android foi lançado em 2008 chamado G1, ou *Google One*. Desde seu lançamento o Android vem registrando um crescimento extremamente acentuado. Atualmente é a plataforma móvel mais utilizada no mundo, com 84% do *marketshare* [Mob16].

Aplicativos Android são arquivos do tipo `apk` que podem ser obtidos a partir de lojas de aplicativos virtuais. A *Google Play Store*, loja virtual oficial do Android, registra em 2016 mais de 2 milhões de aplicativos disponíveis e mais de 200 bilhões de downloads [App16b] [App16a].

Aplicativos Android

Aplicativos Android (apps) são desenvolvidos utilizando a linguagem Java e o Android SDK [Wika] e possuem recursos como imagens e arquivos `XML`. São executados sob uma máquina virtual, Dalvik ou ART, a depender da versão do Android em execução, sendo a última padrão nas versões mais recentes [Wikb] [Andb]. A IDE oficial para desenvolvimento é o Android Studio [Andc].

Um app consiste de um único arquivo com extensão `apk` contendo o código fonte compilado, recursos da aplicação, arquivos de dados e um arquivo de manifesto. Um projeto Android tem a seguinte estrutura:

- **Diretório** `java` contendo o código-fonte Java.
- **Diretório** `res`, derivado da palavra *resources*, contém todos os recursos como imagens, layout de telas e outros.
- **Arquivo** `AndroidManifest.xml` que contém configurações do projeto como: tela inicial, permissões de acesso a recursos do dispositivo e outros.

Componentes da Aplicação

Recursos

2.2 Qualidade de Código

2.3 Maus Cheiros

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.

Capítulo 3

Pesquisa

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 Maus Cheiros

4.1 Code Smell 1

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. 3

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

[Andb] Android: entenda as diferenÃğas entre o art e o dalvik. http://www.tecmundo.com.br/android/54387-android-entenda-diferencas-entre-art-o-dalvik.htm. Last accessed at 30/08/2016. 6

[Andc] Android studio. https://developer.android.com/studio/index.html. Last accessed at 30/08/2016. 6

[App16a] App annie index market q1 2016: China takes japanâĂŹs #2 spot for ios revenue. https://www.appannie.com/insights/market-data/app-annie-index-market-q1-2016/, April 2016. Last accessed at 30/08/2016. 5

[App16b] 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. 1, 2, 5

[Fow99] Martin Fowler. *Refactoring. Improving the Design of Existing Code.* Addison-Wesley, 1999. 6

[Fow06] Martin Fowler. Code smell. http://martinfowler.com/bliki/CodeSmell.html, February 2006. 6

[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

[Hec15] Geoffrey Hecht. An approach to detect android antipatterns. page 766âĂŞ768, 2015. 1, 2, 4

[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. 4

[MDA+] Umme Mannan, Danny Dig, Iftekhar Ahmed, Carlos Jensen, Rana Abdullah, and M Almurshed. Understanding code smells in android applications. 2, 4

[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

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

[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

[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. 6

[Wika] Wikipedia android. https://pt.wikipedia.org/wiki/Android. Last accessed at 30/08/2016. 5

[Wikb] Wikipedia android runtime. https://en.wikipedia.org/wiki/Android_Runtime. Last accessed at 30/08/2016. 6