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

A quantidade de aplicações Android tem crescido de forma acentuada. Projetos Android cada vez mais, precisam ser desenvolvidos e evoluídos de forma rápida para atender aos requisitos dos usuários, tornando-os grandes e complexos projetos de software. Esta velocidade pode levar a más decisões de *design* de código e consequentemente, com o tempo, deteriorar a qualidade do projeto, tornando a manutenção difícil. Apesar de existirem boas práticas reconhecidas para mitigar este problema, projetos Android possuem particularidades onde não se consegue reaproveitá-las, por exemplo com relação a sua camada de apresentação. Nossa pesquisa busca identificar, validar e documentar no formato de maus cheiros de código boas e más prática no desenvolvimento da camada de apresentação de projetos Android. Já existem hoje, maus cheiros catalogados específicos para Android, no entanto, eles indicam problemas com relação ao mau uso de recursos do dispositivo como uso de bateria, cpu e memória. Em contrapartida, nossa pesquisa pretende documentar maus cheiros com relação a legibilidade e manutenabilidade da camada de apresentação de projetos Android. O foco na camada de apresentação é importante pois é onde projetos Android mais se diferenciam de projetos de software tradicionais. Ao final do trabalho, teremos um catálogo validado de maus cheiros na camada de apresentação de projetos Android nativos. Com isso será possível a implementação de ferramentas de detecção automática destes maus cheiros, auxiliando assim, no desenvolvimento de projetos Android com mais qualidade, aumentando a manutenabilidade e reduzindo a propensão a *bugs*.

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

Abstract

The amount of Android applications has grown aggressively. Android projects increasingly need to be developed and evolved quickly to meet the users's requirements, making them large and complex sofrware projects. This speed can lead to poor design decisions and consequently, over time, decrease the quality of the code, making maintenance difficult. Although there are recognized good practices to mitigate this problem, Android projects have specifics where you can not repackage them, for example with respect to the presentation layer. Our research seeks to identify, validate and document in the form of code smells good and bad practice in the development of presentation layer of Android projects. There are bad smells cataloged specific to Android, however, they indicate problems regarding the misuse of the device features such as battery usage, CPU and memory. In contrast, our research aims to document code smells regarding the readability and maintainability of presentation layer of the Android project. The focus on the presentation layer is important because it is where most Android projects differ from traditional software projects. At the end of work, we have a validated catalog of code smells in the presentation layer of native Android projects. It will be possible to implement automatic detection tools, thus helping in the development of Android projects with more quality, increasing the maintainability and reducing the proneness to *bugs*.

Keywords: android, software engineering, code smells, software maintance, code metrics, 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 [App16]. 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], estas ferramentas geram relatórios indicando códigos bons e ruins. Existem também boas práticas reconhecidas de desenvolvimento de software. De fato, muitas destas ferramentas e prática podem ser aplicados aos projetos Android e colaboram no desenvolvimento de um software com qualidade, legível e de fácil manutenção. 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 con-

tém 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, não vemos este diretório em projetos tradicionais. Outra particularidade é com relação ao componente Android ACTIVITY, que representa uma tela [And]. 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, ou seja, podem se tornar classes altamente acopladas.

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.

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...

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 na plataforma.

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 [App16] [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 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 propostos 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 Aplicações Android

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.

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

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

[App16] 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, 3

[Fow99] Martin Fowler. *Refactoring. Improving the Design of Existing Code.* Addison-Wesley, 1999. 5

[Fow06] Martin Fowler. Code smell. http://martinfowler.com/bliki/CodeSmell.html, February 2006. 5

[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. 3

[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. 3, 4

[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. 3

[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. 5

[Ver13] Daniël Verloop. *Code Smells in the Mobile Applications Domain*. PhD thesis, TU Delft, Delft University of Technology, 2013. 5