Análise da aparição de bad smells em projetos Ruby on Rails.

Victor Matheus Fernandes dos Santos - 1602632

Introdução

Inspirado no artigo “**When and Why Your Code Starts to Smell Bad” [1].**

De acordo com Kent Beck, *bad smells* são “**estruturas no código que sugerem a possibilidade de refatoração**”.[2]

Também chamada de code smells - e daqui pra frente, neste conjunto de slides, apenas de smells -, com o tempo surgiram ferramentas denominadas **Code Smells Detector[3]** com o fim de detectá-las.

Introdução

Para este trabalho foram coletados dados de dois projetos, desenvolvidos em Ruby on Rails e disponíveis na plataforma GitHub, para análise.

Com base nos dados que foram coletados tentaremos responder às seguintes questões:

Questões

RQ1: Em quais camadas houveram maior número de inserção de smells em cada projeto?

RQ2: Quais foram os tipos de smell mais identificadas em cada projeto?

RQ3: Há alguma tendência ao desaparecimento de smells em cada projeto ?

Smells a serem analisadas

Todos os projetos serão analisados utilizando um software chamado **Reek**.

Reek é um Code Smell Detector para projetos Ruby que contempla a detecção de todos os tipos de smells a seguir.

Smells a serem analisadas

- Attribute
- Class Variable
- Control Couple
- Data Clump
- Duplicate Method Call
- Instance Variable Assumption
- Irresponsible Module
- Large Class
- Long Parameter List
- Low Cohesion
- Module Initialize

Smells a serem analisadas

- Nested Iterators
- Prima-Donna-Method
- Simulated Polymorphism
- Subclassed From Core Class
- Too Many Statements
- Uncommunicative Name
- Unused Parameters
- Unused Private Method

Método Coleta dos dados

Foram coletados dados de dois projetos desenvolvidos em Ruby on Rails da plataforma GitHub, sendo eles:

AirCasting (1214 commits e 17 contribuidores) - Uma plataforma que auxilia no envio de informações sobre sua saúde via celular para um banco de dados.

Adopt a Hydrant (1730 commits e 16 contribuidores) - Uma plataforma onde civis podem ¨adotar¨ um hidrante e cuidar deles em períodos onde tem muita neve.

Método

Estruturação dos dados do projeto

Os dados foram estruturados da seguinte forma:

Nome	**Tipo**
Nome do Projeto	String
Lista de Commits	List<Commit>

Método
Estruturação dos dados de cada commit

Os dados foram estruturados da seguinte forma:

Nome	**Tipo**
Identificador do Commit	String
Autor do Commit	String
Data do Commit	Date
Índice do Commit	Integer
Lista de Smells identificadas	List<Smell>

Método

Estruturação dos dados de cada smell

Os dados foram estruturados da seguinte forma:

Nome	**Tipo**
Identificador da smell	String
Tipo da smell	String
Descrição da smell	String
Caminho do arquivo associado	String

Método

Exemplo dos dados em formato tabular

	commit_id	commit_author	commit_index	date	filename	smell_id	smell_type	smell_description
1	6d04f1936605c3...	Erik Michaels-Ober <sferik@gmail.com>	2	14-02-2011 13:49:00	codeforamerica-adopt-a-hydrant/app /controllers /welcome_controller.rb	d8214f204602d...	IrresponsibleModule	[1]:IrresponsibleModule: WelcomeController has no descriptive comment [https://github.com/troessner /reek/blob/master/docs/Irresponsible-Module.md]
2	6d04f1936605c3...	Erik Michaels-Ober <sferik@gmail.com>	2	14-02-2011 13:49:00	codeforamerica-adopt-a-hydrant/app/helpers /welcome_helper.rb	539c883bf838c...	IrresponsibleModule	[1]:IrresponsibleModule: WelcomeHelper has no descriptive comment [https://github.com/troessner/reek/blob /master/docs/Irresponsible-Module.md]
3	6d04f1936605c3...	Erik Michaels-Ober <sferik@gmail.com>	2	14-02-2011 13:49:00	codeforamerica-adopt-a-hydrant/test /functional /welcome_controller_...	60c5e0f798c72...	IrresponsibleModule	[3]:IrresponsibleModule: WelcomeControllerTest has no descriptive comment [https://github.com /troessner/reek/blob/master /docs/Irresponsible-Module.md]
4	6d04f1936605c3...	Erik Michaels-Ober <sferik@gmail.com>	2	14-02-2011 13:49:00	codeforamerica-adopt-a-hydrant/test /unit/helpers /welcome_helper_tes...	3614210cb4afe...	IrresponsibleModule	[3]:IrresponsibleModule: WelcomeHelperTest has no descriptive comment [https://github.com/troessner /reek/blob/master/docs/Irresponsible-Module.md]

Método Coleta e estruturação dos dados

Os dois projetos foram escolhidos de forma aleatória na plataforma GitHub, pesquisando sobre projetos que utilizassem **Ruby On Rails**.

Um dos motivos influenciadores para a baixa amostragem de projetos foi o tempo de processamento para coleta e conversão dos dados para .csv . Será tratado com mais detalhes na seção **Ameaças a validação**.

Método
Coleta e estruturação dos dados

Os dados de cada projeto foram coletados através de um script desenvolvido em Ruby pelo autor(eu).

O script é responsável por **clonar os repositórios** de cada projeto, **navegar por todos os commits** localmente e **rodar o software Reek** sobre todos os arquivos modificados no commit que está sendo analisado atualmente.

Em quais camadas houveram maior número de inserção de smells?

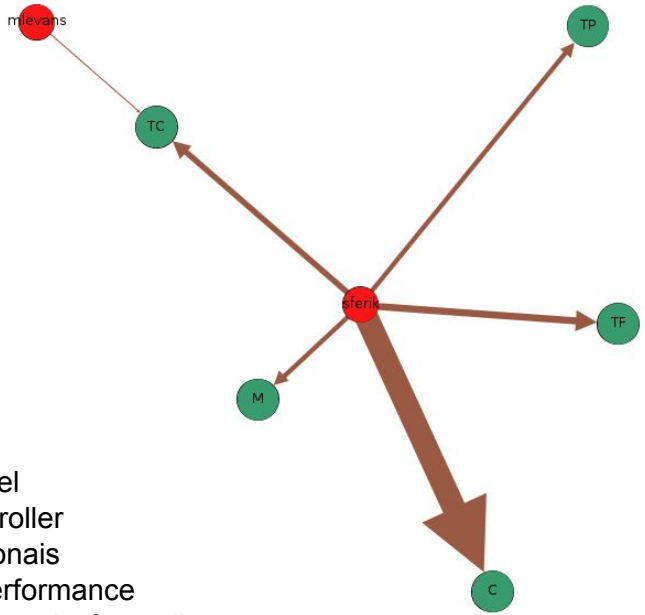
Um pré-processamento foi aplicado para mesclar nós que representavam os mesmos contribuidores.

Este fenômeno ocorre pois um contribuidor pelo definir sua variável **user.name** e **user.email** do git de várias formas diferentes durante o tempo de contribuição no projeto.

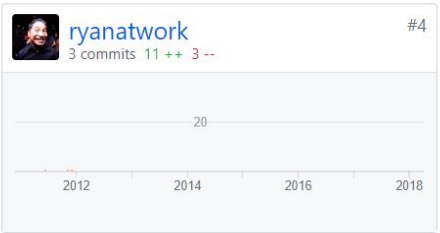
Os grafos bipartido de ligação de nós a seguir gerado pela ferramenta Gephi representa como cada contribuidor colaborou para a inserção de smells em cada camada do projeto.

Suas arestas representam a quantidade de smells que um certo contribuidor inseriu na camada e o tamanho no nó representa a quantidade de contribuidores que houveram para aquela camada.

Em quais camadas houveram maior número de inserção de smells no projeto **Adopt a Hydrant**?



M = Camada Model
C = Camada Controller
TF = Testes funcionais
TP = Testes de performance
TC = Testes da camada Controller

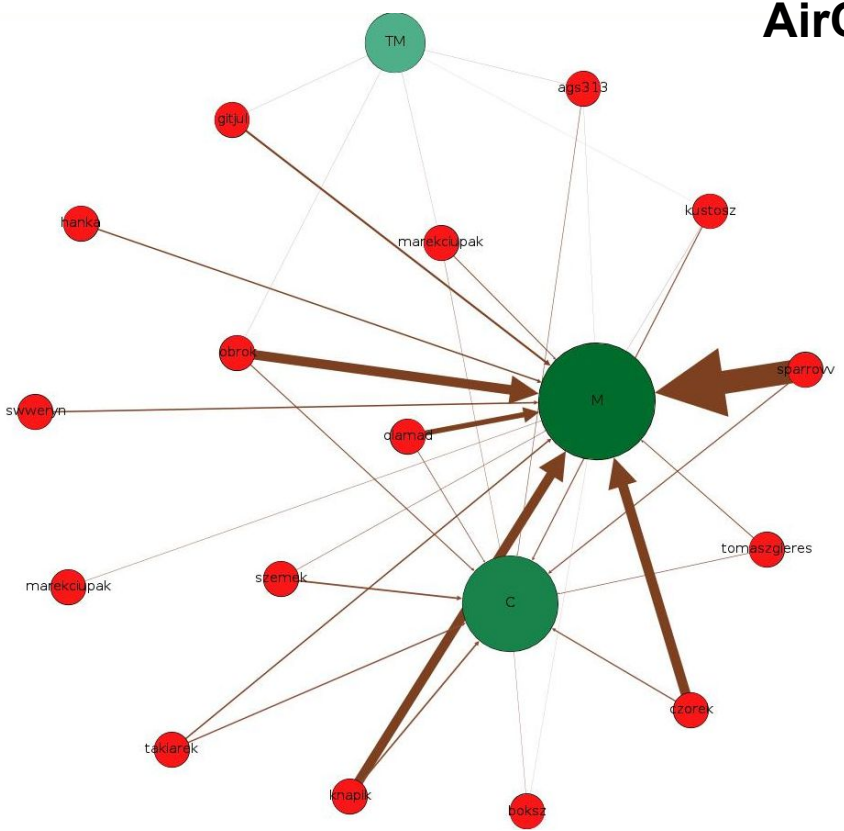


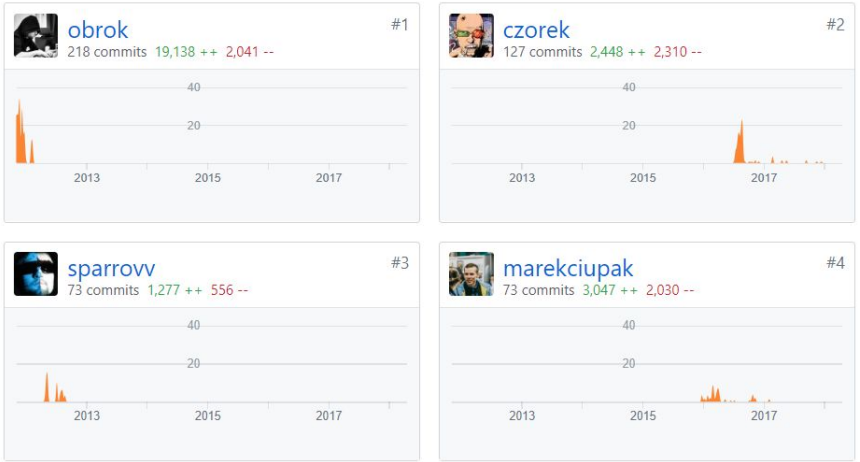
Conclusões

Podemos concluir com base na visualização gerada que, no projeto Adopt a Hydrant:

- Houve uma gigantesca inserção de smells por parte do contribuidor ***sferik*.**
- A camada a mais sofrer com a inserção de smells foi a camada **Controller**.
- Os demais contribuidores do projeto, com exceção do ***mlevans***, não inseriram sequer uma smell no projeto.

Em quais camadas houveram maior número de inserção de smells no projeto **AirCasting**?





M = Camada Model
C = Camada Controller
TM = Testes na camada de modelo

Conclusões

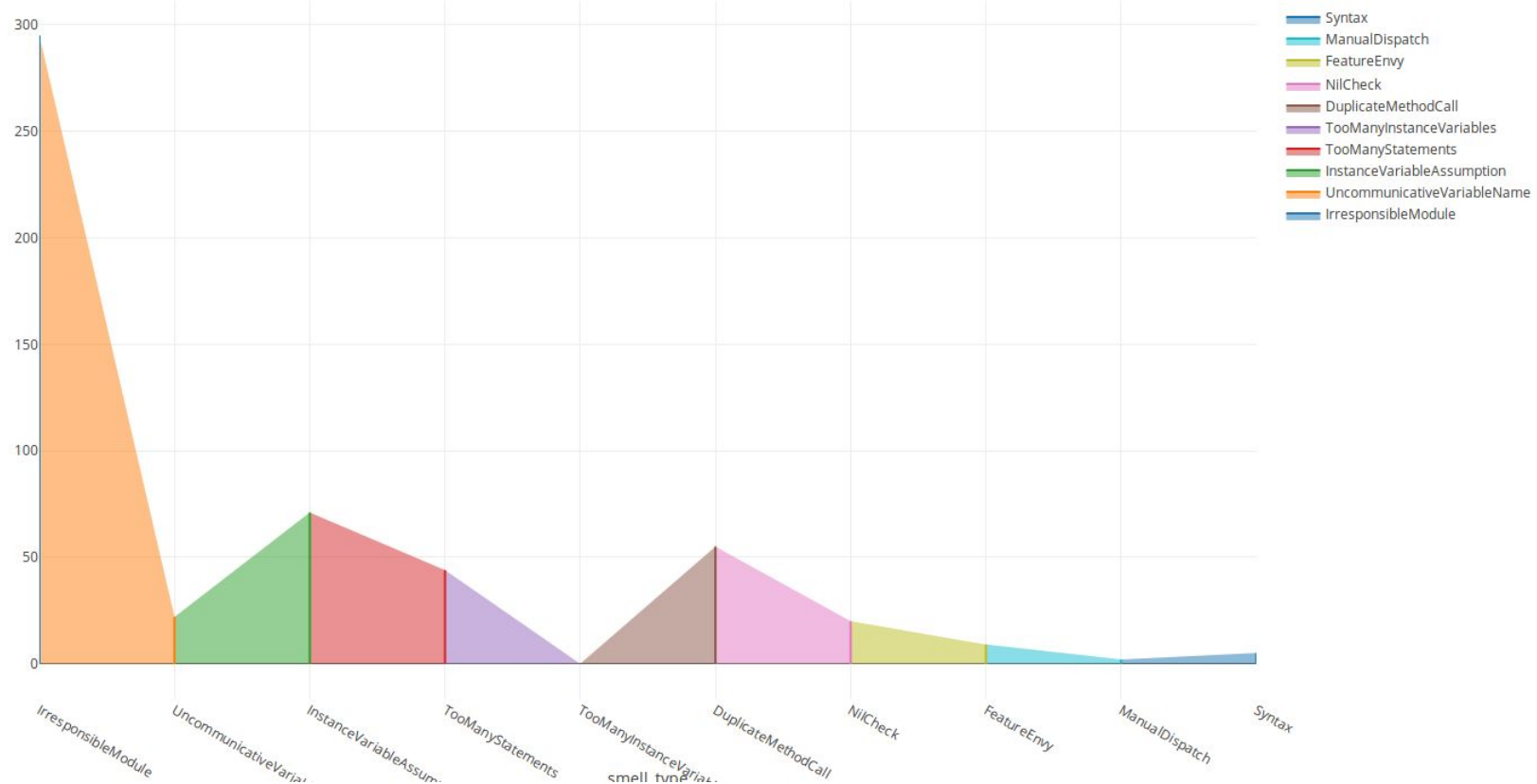
Podemos concluir com base na visualização gerada que, no projeto **AirCasting**:

- A camada **Model** teve grande número de inserção de smells por alguns colaboradores.
- A camada **Controller** teve pequenas inserções de smells por muitos colaboradores.
- Os colaboradores que inseriram smells na camada **Teste Model** inseriram um número menor de smells na Model.

Quais foram os tipos de smell mais identificadas?

O gráfico de área a seguir gerado pela plataforma Ploty representa os tipos de smells que mais apareceram no decorrer dos projetos.

Quais foram os tipos de smell mais identificadas no projeto **Adopt a Hydrant**?

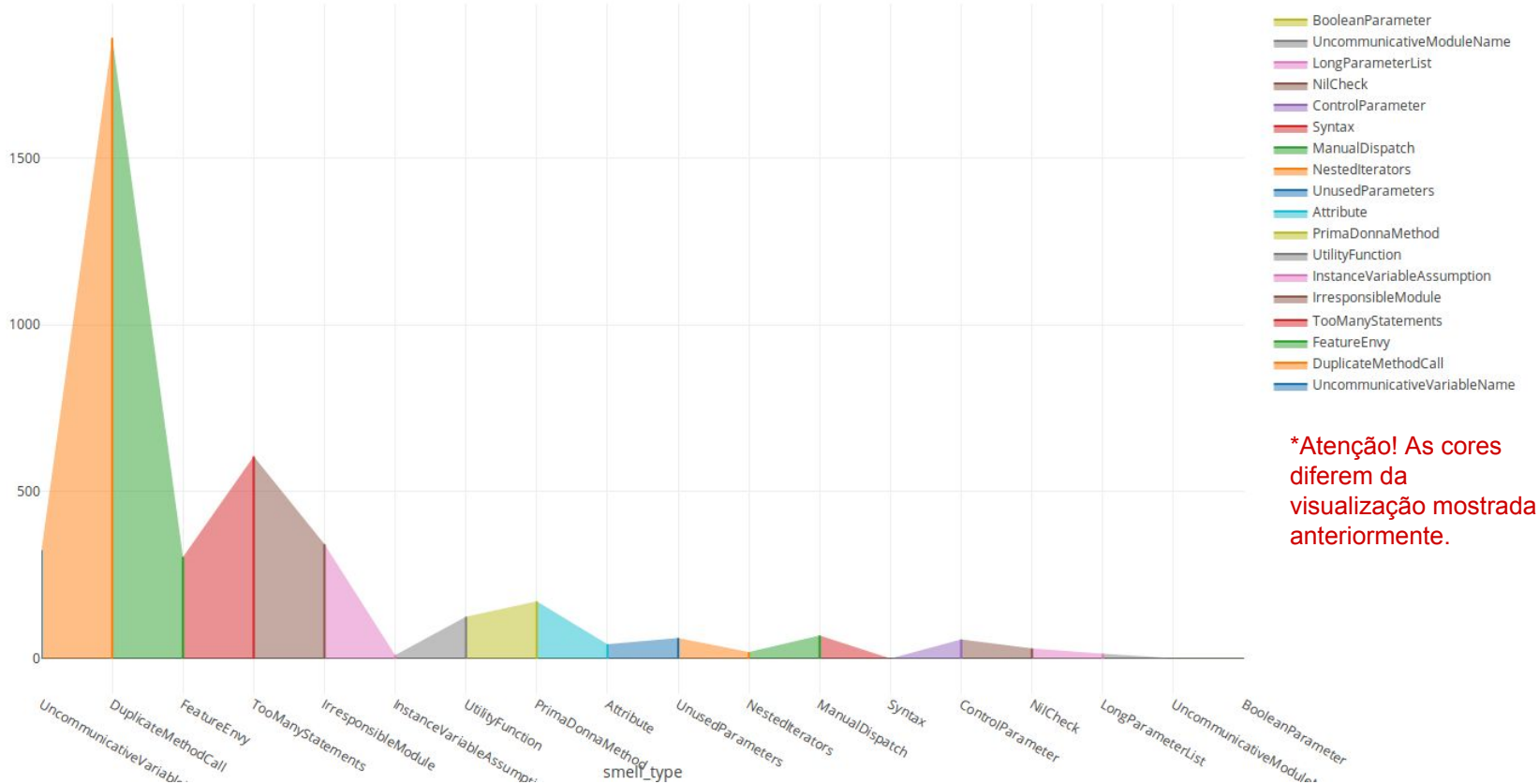


Conclusões

Podemos concluir com base na visualização gerada que, no projeto **Adopt a Hydrant**:

- O tipo de smell **IrresponsibleModule** foi a mais recorrente disparada.
- Houveram raras ocorrências do tipo de smell **TooManyInstanceVariables**, **Syntax** e **Manual Dispatch**.

Quais foram os tipos de smell mais identificadas no projeto **AirCasting**?



*Atenção! As cores diferem da visualização mostrada anteriormente.

Conclusões

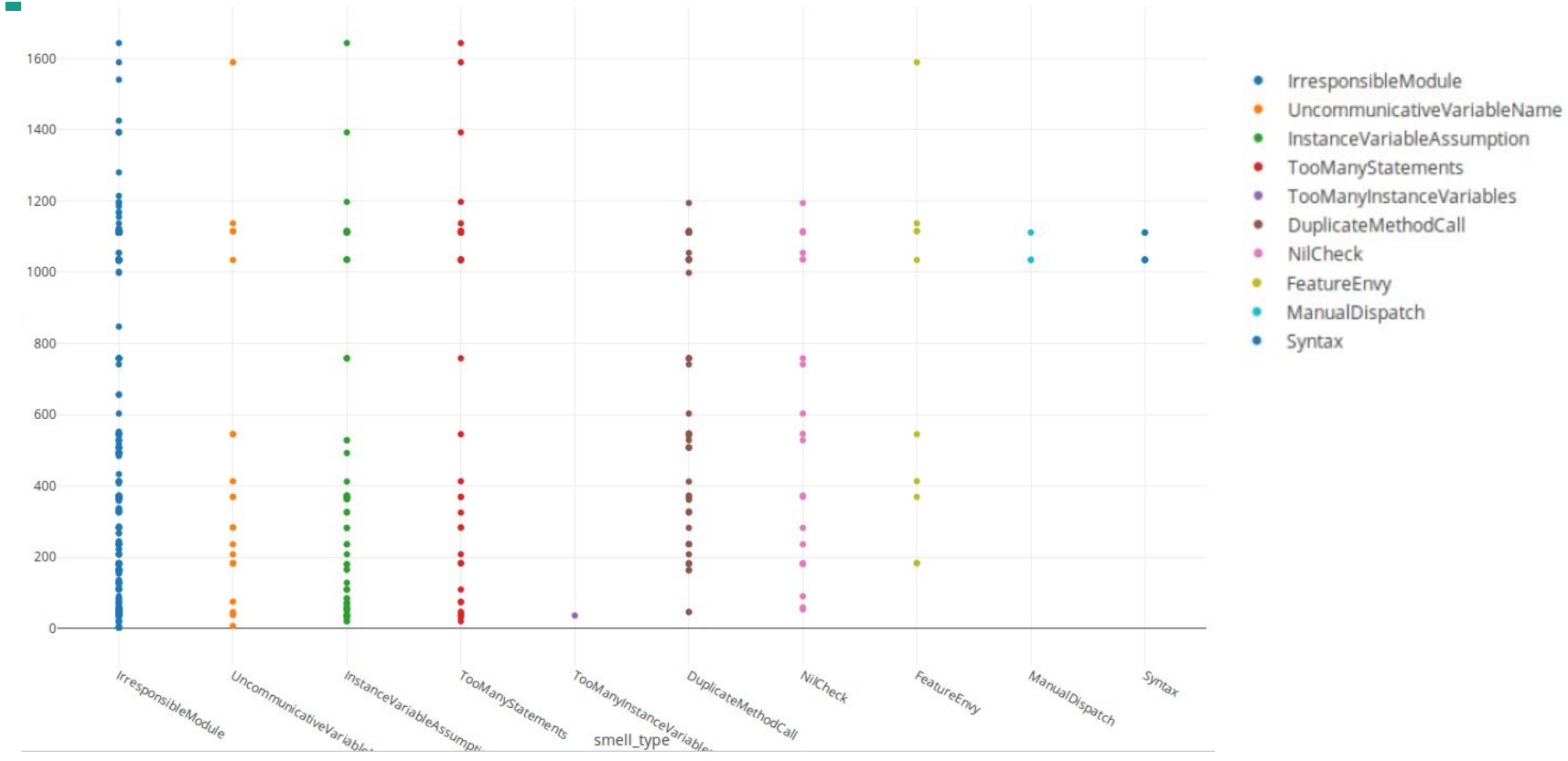
Podemos concluir com base na visualização gerada que, no projeto **AirCasting**:

- O tipo de smell **DuplicateMethodCall** foi a mais recorrente disparada.
- Houve apenas uma ocorrência do tipo de smell **TooManyInstanceVariables, Boolean Parameter e Uncommunicate Variable**.

Tendência ao desaparecimento de smells

O scatterplot a seguir gerado pela plataforma Ploty representa a quantidade de smells que apareceu no decorrer os commits realizados.

Tendência ao desaparecimento de smells no projeto **Adopt a Hydrant**



Conclusões

Podemos concluir com base na visualização gerada que, no projeto **Adopt a Hydrant**:

- O número de smells de todos os tipos diminuíram no decorrer do tempo.
- Smells do tipo Syntax e Manual Dispatch demoraram a ser inseridas neste projeto.

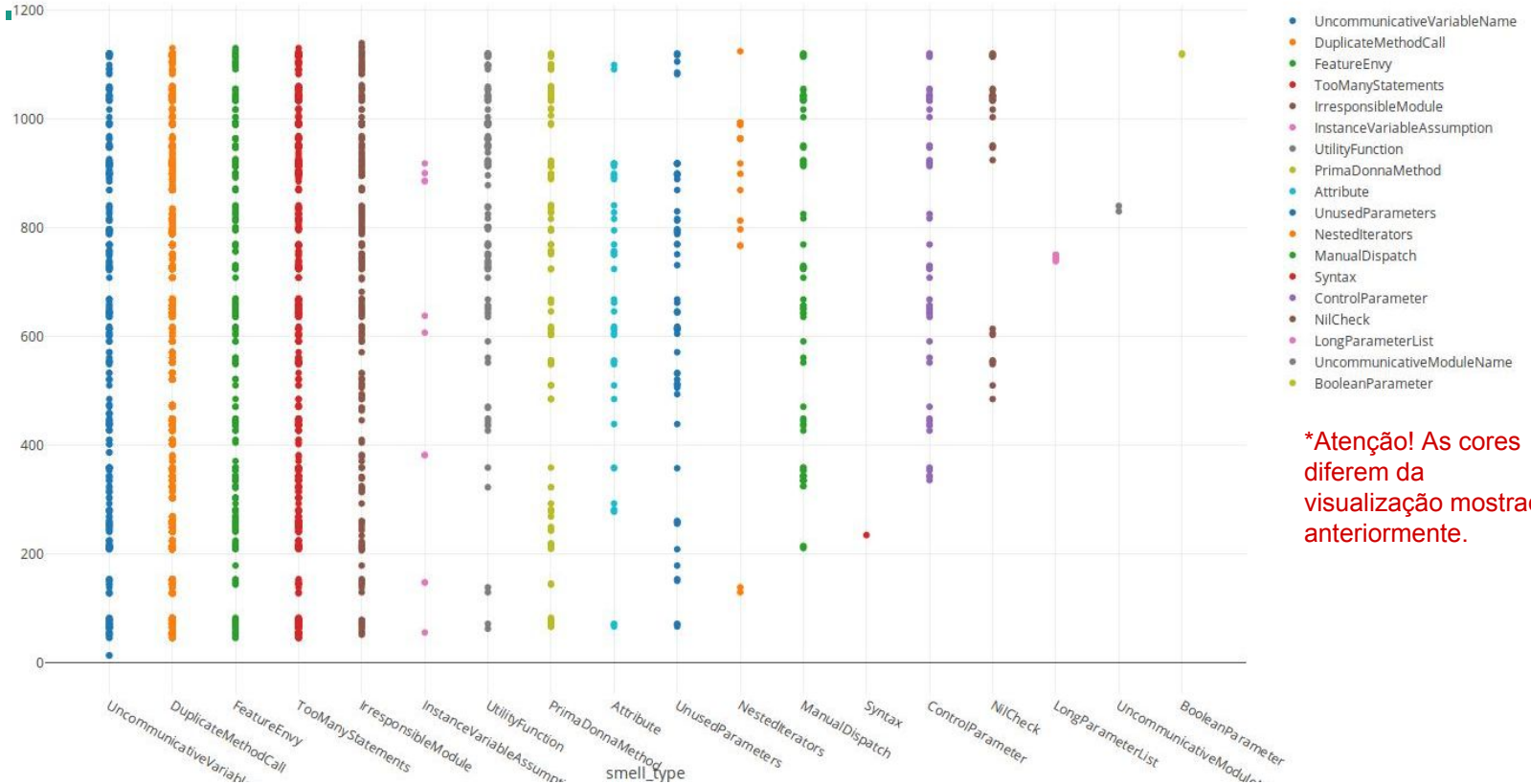
Tendência ao desaparecimento de smells no projeto **Adopt a Hydrant**

Conclusões

Podemos concluir com base na visualização gerada que, no projeto **Adopt a Hydrant**:

- sferik está se tornando bom na utilização das boas práticas propostas pelo pessoal do Reek.

Tendência ao desaparecimento de smells no projeto **AirCasting**



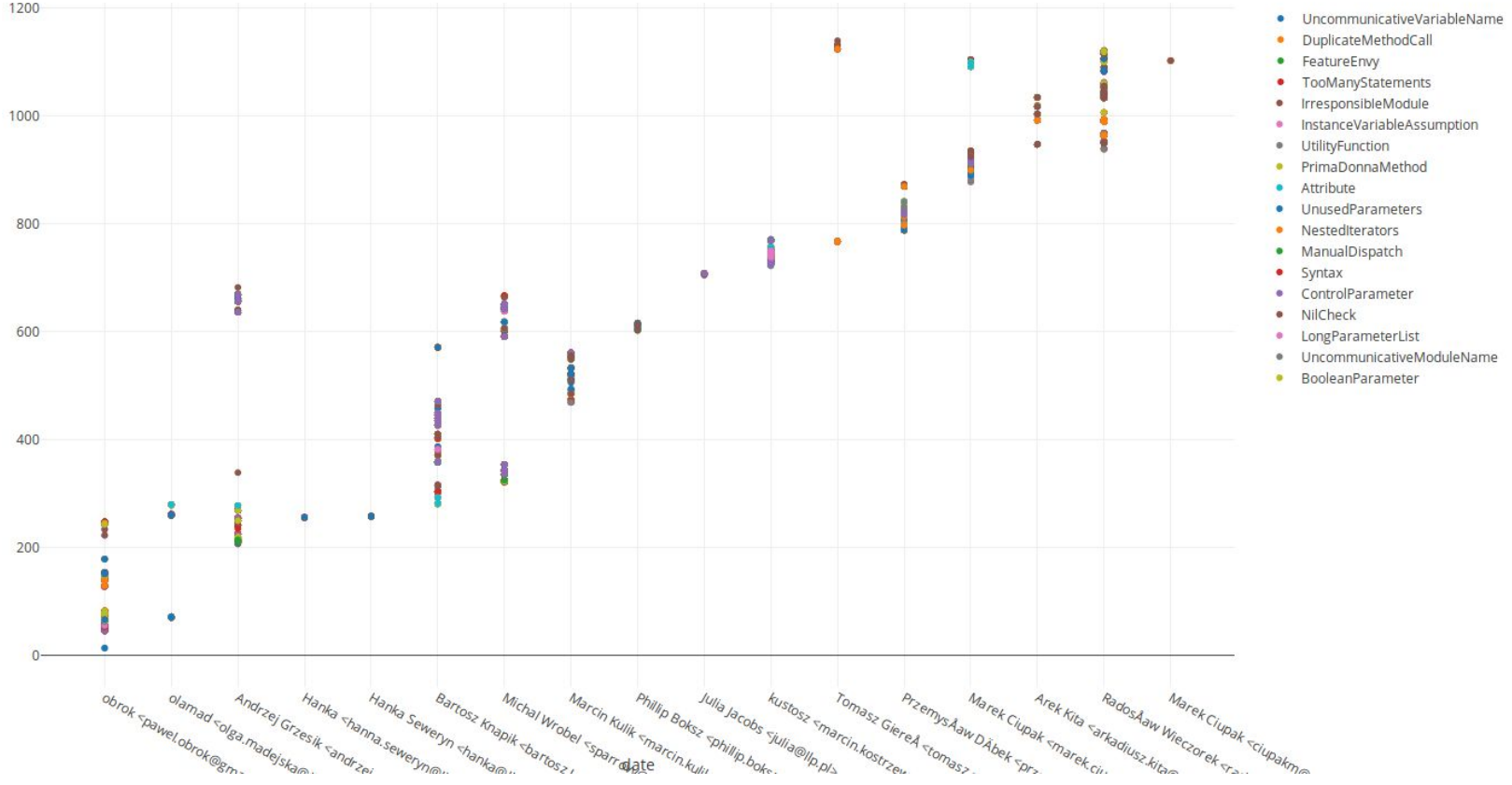
*Atenção! As cores diferem da visualização mostrada anteriormente.

Conclusões

Podemos concluir com base na visualização gerada que, no projeto **AirCasting**:

- A maioria dos tipos de smell se manteve presente no decorrer de todo o projeto.
- Mais tipos de smells estão presentes neste projeto do que no Adopt a Hydrant, cuja maioria das contribuições está centralizada em uma só pessoa.

Tendência ao desaparecimento de smells no projeto **AirCasting**



Conclusões

Podemos concluir com base na visualização gerada que, no projeto **AirCasting**:

- A inserção de smells em massa não parte de todos os usuários.

Ameaças a validação das conclusões

- Baixa amostragem para qualquer conclusão ampla.
- Baixo número de contribuidores ativos nos projetos analisados.
- Conclusões focadas no escopo de cada projeto.
- Ruby teve muitas evoluções ao longo do tempo. No passado boas práticas não eram tão padronizadas e difundidas.

Dúvidas

Espaço aberto para dúvidas ou sugestões.

Referências

[1] When and Why Your Code Starts to Smell Bad. Disponível em: <https://ieeexplore.ieee.org/document/7194592/>. Acesso em: 1 jul. 2018.

[2]Como resolver code smells, bad smells e Clean code. Disponível em: <https://www.devmedia.com.br/como-resolver-code-smells-bad-smells-e-clean-code/34094>. Acesso em: 1 jul. 2018.

[3]On the evaluation of code smells and detection tools. Disponível em: <https://jserd.springeropen.com/articles/10.1186/s40411-017-0041-1>. Acesso em: 1 jul. 2018.