UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

ENGENHARIA DE SOFTWARE

DESCRIÇÃO DO PROJETO E MEDIÇÃO INICIAL DA QUALIDADE
Projeto Banco Imobiliário

Equipe:

Guilherme Rodrigues

Hugo Rodrigues

Professora:

Carla Ilane Moreira Bezerra

QUIXADÁ

Agosto, 2021

1 DESCRIÇÃO DO PROJETO

O trabalho e a análise foram feitas baseadas no projeto realizado como atividade final da cadeira de PDS (https://github.com/victorlucss/TrabalhoFinalPDS), um projeto criado por Victor Lucas que tem por objetivo reproduzir o jogo banco imobiliário. Como apresentado na descrição desse trabalho, seria necessária a utilização das ferramentas métricas em um software com uma complexidade relativamente avançada e que fornecesse dados satisfatórios às ferramentas, justificando assim nossa escolha de projeto.

Atualmente o projeto se encontra na fase de testes, contando com as principais funcionalidades essenciais para a jogabilidade implementadas. O back-end da aplicação, que é o alvo deste trabalho, est disponível a partir do seguinte link: https://github.com/victorlucss/TrabalhoFinalPDS

A seguir, está uma tabela que mostra algumas propriedades do projeto:

Tabela 2 - Propriedades do projeto Banco Imobiliário

Projeto	LOC	# de classes	# de releases
Banco Imobiliário	13.132	182	1

2 AVALIAÇÃO DO PROJETO

Medição 1 – Antes de refatorar o projeto

A aba Sem Refatoração da planilha disponível no seguinte link: https://docs.google.com/spreadsheets/d/1Y7cuYQm4Gi5lHEtJkebseJfNHmbX2flT/edit?usp=sharing&ouid=100549571424145734462&rtpof=true&sd=true mostra o resultado detalhado das medições realizadas a partir da ferramenta Understand antes da refatoração do projeto. Ela destaca o valor obtido para as métricas dos atributos Coesão, Complexidade, Herança, Acoplamento e Tamanho para cada uma das classes, métodos, pacotes e arquivos do projeto, por fim apresenta o somatório obtido para cada métrica e atributo.

A Tabela 3 destaca quais as métricas utilizadas para medir os atributos de qualidade enquanto a Tabela 4 mostra total obtido para cada uma dessas métricas antes do início das refatorações.

Tabela 3 - Métricas usadas para medir os atributos internos de qualidade.

Atributo	Métrica	Descrição
Coesão	LCOM	Mede a coesão de uma classe. Quanto maior o valor dessa métrica, menos coesiva é a classe.
Complexidade	ACC	Média da complexidade ciclomática de todos os métodos. Quanto maior o valor dessa métrica, mais complexos são as classes e métodos.
	SCC	Somatório da complexidade ciclomática de todos os métodos. Quanto maior o valor dessa métrica, mais complexos são as classes e métodos.
	EVG	Mede o grau na qual um módulo contém construtores não estruturados. Quanto maior o valor dessa métrica, mais complexos são as classes e métodos.

	Nesting	Nível Máximo de Aninhamento de construções de controle. Quanto maior o valor dessa métrica, maior é a complexidade de classes e métodos
Herança	DIT	O número de níveis que uma subclasse herda de métodos e atributos de uma superclasse na árvore de herança. Quanto maior o valor dessa métrica, maior é o grau de herança de um sistema.
	NOC	Número de subclasses de uma classe. Quanto maior o valor dessa métrica maior é o grau de herança de um sistema.
	Base Classes	Número imediato de classes base. Quanto maior o valor dessa métrica, maior o grau de herança de um sistema.
Acoplamento	CBO	Número de classes que uma classe está acoplada. Quanto maior o valor dessa métrica, maior é o acoplamento de classes e métodos.
Tamanho	LOC	Número de linhas de código, excluindo espaços e comentários. Quanto maior o valor dessa métrica, maior é o tamanho do sistema.
	CLOC	Número de linhas com comentários. Quanto maior o valor dessa métrica, maior é o tamanho do sistema.
	NIM	Número de métodos de instância. Quanto maior o valor dessa métrica, maior é o tamanho do sistema.
	CDL	Número de classes. Quanto maior o valor dessa métrica, maior é o tamanho do sistema.

Detecção dos Code Smells

Ao todo foram encontrados 39 ocorrências de 7 tipos de code smells diferentes. A detecção ocorreu a partir da ferramenta JSpirit e a tabela abaixo mostra o total de ocorrências para cada smell.

Tabela 5 – Code smells do projeto Desenvolva.

Nome do Code Smell	**Quantidade**
God Class	1
Brain Method	1

Shotgun Surgery	3
Tradition Breaker	3
Refused Parent Bequest	7
Dispersed Coupling	9
Feature Envy	15

Tabela 6 – Medição inicial

Atributo Interno de Qualidade	**Métrica**	**Valor da Métrica**	**Total do Atributo**
Coesão	LCOM	2498	2498
Complexidade	ACC	190	3154
	SCC	2305	
	EVG	437	
	Nesting	222	
Herança	DIT	146	311
	NOC	38	
	Base Classes	127	
Acoplamento	CBO	162	162
Tamanho	LOC	13172	15634
	CLOC	1071	
	NIM	1209	
	CDL	182	

Medição 2 - Após refatorar os code smells Dispersed Coupling

O projeto apresentava inicialmente 9 Dispersed Coupling e para resolvê-los foi necessário extrair responsabilidades dos métodos que apresentavam o code smell, para novos

métodos criados para a adequação do código(Split method). Devido a estrutura original do código, a refatoração inicial desses métodos revelou novos code smells não detectados anteriormente pela ferramenta Understand, o principal foi a god class ControlBancoImobiliario que foi resolvida inicialmente com a criação de uma nova classe (extract class) ControlPrisao que absorveu parte das responsabilidades da god class por meio da técnica move method, resolvendo parte do problema como pode ser visto em ControlPrisao, no entanto a classe continuou apresentando mais responsabilidades do que deveria, fazendo com que esse processo fosse necessário ser realizado mais uma vez, agora com a criação da classe ControlPagamentos, o processo pode ser visto em Control Pagamentos.

O resultado da medição realizada após a remoção de todos os Dispersed Coupling em conjunto com a God Class descoberta pode ser visto na Tabela 6. Nela é possível perceber que houve um aumento em quase todos os atributos de qualidade, mas isso não configura necessariamente uma piora da qualidade do projeto. A única métrica a diminuir de fato nessa medição foi a ACC no atributo de Complexidade, e duas métricas permaneceram iguais nas métricas de herança(NOC) e nas de tamanho (CLOC) , o restante apresentou aumento como pode ser visto na tabela a seguir.

Tabela 7 – Resultado da medição após a refatoração de Dispersed Coupling + nova God Class.

Atributo Interno de Qualidade	**Métrica**	**Valor da Métrica**	**Total do Atributo**
Coesão	LCOM	2634	2634
Complexidade	ACC	185	3319
	SCC	2441	
	EVG	461	
	Nesting	232	
Herança	DIT	148	315
	NOC	38	
	Base Classes	129	
Acoplamento	CBO	175	175
Tamanho	LOC	13672	16222
	CLOC	1071	
	NIM	1293	

	CDL	186	

Legenda: Maior Igual Menor

Medição 3 - Após refatorar o code smell Brain Method

Ao todo o Banco Imobiliário apresentava uma instância de Brain Method que foi refatorada com sucesso em conjunto com um dispersed coupling no método Companhia.aplicarEfeito. Para isso, foi utilizada a técnica Extract Method. Extract Method consiste basicamente em dividir um método maior e complexo em métodos menores, como podemos ver em Solução Brain Method, também foi necessário a renomeação da classe por falhas no portugues do projeto original.

O resultado da medição após a refatoração pode ser visto na Tabela 7. Novamente é possível perceber que houve um pequeno aumento nos atributos de qualidade, o que não configura necessariamente uma piora, dado que o code smell existente foi removido. Ocorreu mais uma vez diminuição em ACC de complexidade, enquanto as métricas de herança permaneceram iguais, junto com CLOC e CDL de tamanho. O restante das métricas se mantiveram a subir como pode ser visto na tabela a seguir.

Tabela 8 – Resultado da medição após a refatoração de Brain Method

Atributo Interno de Qualidade	Métrica	Valor da Métrica	Total do Atributo
Coesão	LCOM	2753	2753
Complexidade	ACC	181	3408
	SCC	2521	
	EVG	473	
	Nesting	233	
Herança	DIT	148	315
	NOC	38	
	Base Classes	129	
Acoplamento	CBO	177	177
Tamanho	LOC	13955	16547
	CLOC	1071	
	NIM	1335	
	CDL	186	

Legenda: Maior Igual Menor

Medição 4 - Após refatorar o code smell Shotgun Surgery

Inicialmente foram identificadas 3 instâncias de Shotgun Surgery no projeto, todas estavam ligadas a métodos get de classes base como Jogador e ContaBancaria, apesar de aparentemente não parecer um malefício de fato para o código, a equipe se concentrou na refatoração das instâncias identificadas e conseguiu resolvê-las com sucesso.

A estratégia usada para solucionar as ocorrências de Shotgun Surgery foi tentar diminuir a quantidade de pontos de mudança nas classes que usam o método afetado por esse smell e a criação de um novo método get nas classes base referentes que serviram para melhor identificação de que tipo de jogador estava sendo acessado determinado dado. Deste modo, criou-se um método local nas classes ligadas a cada método marcado com Shotgun Surgery para apenas fazer sua chamada. Após isso, substituiu-se as chamadas ao método Shotgun por chamadas ao novo método local. Assim, se um método Shotgun Surgery sofrer alguma alteração, apenas uma mudança precisará ser feita nas classes que o usam e o código fica, portanto, menos amarrado. Podemos ver como foi feito em Shotgun Surgery.

A Tabela 8 mostra o resultado da medição após remover as 3 ocorrências de Shotgun Surgery. Analisando a tabela consegue-se perceber que ocorreu redução nas métricas ACC e Nesting do atributo de qualidade Complexidade, mantiveram-se os números de todas as métricas de herança e das métricas CLOC e CDL em Tamanho. A equipe acredita que essa redução tem relação com a troca de encadeamentos de métodos get por somente uma chamada simples nas classes ligadas aos métodos Shotgun Surgery. O restante das métricas apresentou crescimento.

Tabela 9 – Resultado da medição após a refatoração de Shotgun Surgery

Atributo Interno de Qualidade	Métrica	Valor da Métrica	Total do Atributo
Coesão	LCOM	2766	2766
Complexidade	ACC	178	3467
	SCC	2569	
	EVG	488	
	Nesting	232	
Herança	DIT	148	315
	NOC	38	
	Base Classes	129	
Acoplamento	CBO	179	179
Tamanho	LOC	14065	16702
	CLOC	1071	
	NIM	1380	
	CDL	186	

Legenda: Maior Igual Menor

Medição 5 - Após refatorar o code smell Feature Envy

Inicialmente o projeto possuía 16 ocorrências de Feature Envy e após as refatorações desempenhadas surgiram novas ocorrências que foram imediatamente corrigidas, por fim restaram 8 Feature Envys em 2 classes. O alto número de ocorrências não resolvidas se dá devido ao fato de que as classes apontadas pelo jspirit são diretamente conectadas e dependentes das parent classes do Jplay, um framework do java para auxiliar em jogos 2D, a equipe julgou melhor seguir as medições sem a alterações dessas classes uma vez que não apresentavam conturbações ao projeto, veja melhor o caso em Collision.collided. De toda forma, para realizar essas refatorações foram utilizadas novamente as técnicas Extract Method e Move Method. Deste modo, a equipe buscou quebrar os métodos que usavam em excesso recursos de outras classes e mover operações sobre os dados para a classe mais fortemente relacionada a eles, fazendo, dessa forma, com que a classe marcada como Feature Envy fizesse menos acessos à classe invejada. Veja um exemplo de correção em Feature Envy Simples e também vale ressaltar a correção de um Feature Envy que acabou resultando na volta do nosso problema, a god class na classe ControlBancoImobiliario, sendo necessário assim a extração de métodos para a classe ControlCampos que pode ser vista em Featury Envy - God Class.

O Resultado da medição após a remoção das 8 instâncias de Feature Envy pode ser visto na Tabela 9. Analisando essa tabela é perceptível a não ocorrência de redução nas métricas de qualidade comparadas a medição anterior, no entanto os números se mantiveram iguais em ACC e Nesting de complexidade, todas as métricas de herança e nas CLOC e CDL de tamanho. Essa constância mantida em diferentes métricas pode ter sido gerada devido a forma de correção das Shotgun Surgeries analisadas anteriormente, que muitas vezes acabava resultando também na correção de Feature Envy dos métodos relacionados .

Ocorreu ainda aumento nas métricas LCOM (Coesão), SCC e EVG (Complexidade), CBO(Acoplamento), Loc e NIM (Tamanho). Tal aumento está relacionado à adição de mais métodos e atributos às classes base, que antes continham somente getters, setters e construtores.

Tabela 10 – Resultado da medição após a refatoração de Feature Envy.

Atributo Interno de Qualidade	Métrica	Valor da Métrica	Total do Atributo
Coesão	LCOM	2811	2811
Complexidade	ACC	178	3522
	SCC	2613	
	EVG	499	
	Nesting	232	
Herança	DIT	148	315
	NOC	38	
	Base Classes	129	

Acoplamento	CBO	186	186
Tamanho	LOC	13955	16547
	CLOC	1071	
	NIM	1335	
	CDL	186	

Legenda: Maior Igual Menor

Medição 5 - Após refatorar o code smell God Class

As God Classes do nosso projeto foram identificadas após outras correções, inicialmente apenas uma foi identificada pela ferramenta, porém o código continha um total de 3. Dessas 3 instâncias presentes , a equipe conseguiu remover completamente 2, a que não foi refatorada foi devido ao fato dela estar diretamente ligada ao pacote Jplay, dificultando a correção sem gerar problemas na execução do código.

A estratégia utilizada para solucionar as ocorrências de God class foi a mesma apresentada nas resoluções em suas ocorrências anteriormente reportadas, foi aplicado o Move Method diminuir as responsabilidades da God Class e foram criados novas classes quando necessário, como exemplo temos ControlBancoImobiliario e Terreno

Os resultados da medição após as refatorações de God Class podem ser vistos na Tabela 11. Foi difícil isolar os resultados de suas resoluções em específico devido ao fato de suas identificações ocorrerem em diferentes momentos da refatoração, acreditamos que no momento certo teríamos um resultado diferente no quadro. No entanto, podemos observar nessa análise que a maioria das métricas de qualidade aumentaram, com exceção de ACC de complexidade, NOC de herança e CLOC tamanho.

Tabela 11 – Resultado da medição após a refatoração de God Class

Atributo Interno de Qualidade	**Métrica**	**Valor da Métrica**	**Total do Atributo**
Coesão	LCOM	2847	2847
Complexidade	ACC	178	3556
	SCC	2637	
	EVG	505	
	Nesting	236	
Herança	DIT	149	417
	NOC	38	
	Base Classes	130	

Acoplamento	CBO	193	543
Tamanho	LOC	14203	10610
	CLOC	1071	
	NIM	1431	
	CDL	188	

Legenda: Maior Igual Menor

Análise Final

De forma geral, todos os atributos de qualidade aumentaram, mas se olharmos mais detalhadamente para cada uma das métricas analisadas, podemos perceber a melhora em ACC de complexidade e a constância em NOC herança e CLOC tamanho.

Em termos de dificuldade no momento da refatoração, percebeu-se a importância dos padrões de projeto para a arquitetura do código, devido ao mal uso de elementos de POO, resolver God Classes e Dispersed Coupling foram as tarefas mais difíceis, levando assim horas de análise do código para que houvesse um entendimento do que o elemento fazia e a onde o alocar de maneira correta.

Tabela 12 – Resultado das medições e análise

Atributo Interno de Qualidade	Dispersed Coupling	Brain Method	Shotgun Surgery	Feature Envy	God Class	Mudança Final
Coesão	LCOM	LCOM	LCOM	LCOM	LCOM	+ 349
Complexidade	ACC	ACC	ACC	ACC	ACC	- 12
	SCC	SCC	SCC	SCC	SCC	+ 332
	EVG	EVG	EVG	EVG	EVG	+ 68
	Nesting	Nesting	Nesting	Nesting	Nesting	+ 14
Herança	DIT	DIT	DIT	DIT	DIT	+ 3
	NOC	NOC	NOC	NOC	NOC	0
	Base Classes	Base Classes	Base Classes	Base Classes	Base Classes	+ 3
Acoplamento	CBO	CBO	CBO	CBO	CBO	+ 31
Tamanho	LOC	LOC	LOC	LOC	LOC	+ 1031
	CLOC	CLOC	CLOC	CLOC	CLOC	0
	NIM	NIM	NIM	NIM	NIM	+ 222
	CDL	CDL	CDL	CDL	CDL	+ 6

Legenda: Maior; Igual, Menor