UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

RELATÓRIO – TRABALHO FINAL QUALIDADE DE SOFTWARE
Planejador JavaFF

Equipe:

Antônia Deigela Lima Rufino

João Victor Aquino Correia

Professora:

Carla Ilane Moreira Bezerra

QUIXADÁ

Junho, 2022

SUMÁRIO

1 DESCRIÇÃO DO PROJETO

Planejamento automatizado é o processo que gera uma sequência de ações necessárias para que um agente inteligente resolva um determinado problema. Portanto, dado um problema, existirá um conjunto de ações necessárias para que, a partir de um estado inicial, chegue-se a uma resolução, que será o estado final. Sendo assim, dado uma situação inicial, um conjunto de ações e uma situação final desejada, a tarefa de planejamento consiste em determinar uma sequência de ações que solucionem o problema, ou seja, uma sequência que atinja a situação desejada a partir da situação inicial. No planejador JavaFF foram implementados diversos algoritmos de buscas (progressiva, heurística, gulosa de melhor escolha, busca A*), muitos deles utilizam heurísticas para chegar no melhor resultado. O domínio do robô de Marte (Mars Rover Domain) foi utilizado como experimento.

JavaFF tem como autor Andrew Coles, e foi desenvolvido com objetivo de fornecer uma experiência prática de aprendizado em planejamento de inteligência artificial (IA) para um público de graduação. Ele percebeu que ao projetar um currículo para um curso de graduação em IA, um dos principais desafios é como construir exercícios práticos para acompanhar o material ensinado, levando em consideração as habilidades dos alunos e a quantidade de tempo disponível para o trabalho prático. Foi neste contexto que o projeto chegou para nós, a Dra. Maria Viviane de Menezes passou como trabalho acrescentar a busca A* ao Planejador JavaFF.

O projeto foi implementado na linguagem Java, usando técnicas de programação orientada a objetos. E o domínio do robô de Marte está na linguagem PDDL (a planning domain definition language, ou PDDL, é uma linguagem baseada na lógica de predicados que busca permitir comparações de desempenho entre os planejadores em diferentes domínios).

Link do projeto: Qualidade_de_Software---Trabalho_final

Tabela 1 – Características do Projeto

Projeto	LOC	Número de classes	Número de releases
Planejador JavaFF	65.536	222	0

Tabela 1.1 - Característica do Projeto depois das refatorações

LOC	Número de classes	Número de releases
64.756	224	17

2 AVALIAÇÃO DO PROJETO

2.1 Medição 1 – Antes de refatorar o projeto

Na Tabela - 2 a seguir mostra a medição das 13 métricas (LCOM2, ACC, SCC, EVG, MaxNet, DIT, NOC, IFANIN, CBO, LOC, CLOC, NIM E CDL) e os 5 atributos de qualidade (coesão, complexidade, herança, acoplamento, e tamanho), antes do projeto ser refatorado. Foi utilizada a ferramenta Understand para detectar todas as métricas do projeto — Planejador JavaFF.

Tabela 2 – Medição dos atributos de qualidade antes de refatorar o projeto.

Sistema	Coesão	Complexidade				Herança			Acoplamento	Tamanho			
	LCOM2	ACC	SCC	EVG	MaxNet	DIT	NOC	IFANIN	CBO	LOC	CLOC	NIM	CDL
S1 antes da refatoração	4.001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222

Tabela 3 – Métricas dos atributos internos de qualidade (MCCABE, 1976; CHIDAMBER; KEMERER, 1994; LORENZ; KIDD, 1994; DESTEFANIS *et al.*, 2014)

Atributos	Métricas	Descrição
Coesão	*Lack of Cohesion of Methods (LCOM2) (CHIDAMBER; KEMERER, 1994)*	Mede a coesão de uma classe. Quanto maior o valor dessa métrica, menos coesiva é a classe.
Acoplamento	*Coupling Between Objects (CBO) (CHIDAMBER; KEMERER, 1994)*	Número de classes que uma classe está acoplada Quanto maior o valor dessa métrica, maior é o acoplamento de classes e métodos.
Complexidade	*Average Cyclomatic Complexity (ACC) (MCCABE, 1976)*	Média da complexidade ciclomática de todos os métodos. Quanto maior o valor dessa métrica, mais complexa são a classes e métodos.
	Sum Cyclomatic Complexity (SCC) (MCCABE, 1976)	Somatório da complexidade ciclomática de todos os métodos. Quanto maior o valor dessa métrica, mais complexos são as classes e métodos.
	Nesting (MaxNest) (LORENZ; KIDD, 1994)	Nível máximo de aninhamento de construções de controle. Quanto maior o valor dessa métrica, maior é a complexidade de classes e métodos.
	Essential Complexity (EVG) (MCCABE, 1976)	Mede o grau na qual um módulo contém construtores não estruturados. Quanto maior o valor dessa métrica mais complexas são as classes e métodos.
Herança	*Number Of Children (NOC) (CHIDAMBER; KEMERER, 1994)*	Número de subclasses de uma classe. Quanto maior o valor dessa métrica maior é o grau de herança de un sistema.
	Depth of Inheritance Tree (DIT) (CHIDAMBER; KEMERER, 1994)	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.
	Bases Classes (IFANIN) (DESTEFANIS et al., 2014)	Número imediato de classes base. Quanto maior o valor dessa métrica, maior o grau de herança de um sistema.
Tamanho	*Lines of Code (LOC) (LORENZ; KIDD, 1994)*	Número de linhas de código, excluindo espaços e comentários. Quanto maior o valor dessa métrica, maior é o tamanho do sistema.
	Lines with Comments (CLOC) (LORENZ; KIDD, 1994)	Número de linhas com comentários. Quanto maior o valor dessa métrica maior o tamanho do sistema.
	Classes (CDL) (LORENZ; KIDD, 1994)	Número de classes. Quanto maior o valor , maior o tamanho do sistema.
	Instance Methods (NIM) (LORENZ; KIDD, 1994)	Número de métodos de instância. Quanto maior o valor dessa métrica maior é o tamanho do sistema.

2.2 Detecção dos Code Smells

Foi identificado 108 Code Smells no projeto com a ferramenta JSpirit. Classificados em oito grupo de Code Smells: Feature Envy, Dispersed Coupling, Data Class, Refused Parent Bequest (Refused Bequest), Shotgun Surgery, Brain Method, Intensive Coupling e God Class.

Foram utilizadas outras ferramentas para testar a viabilidade, foram elas: PMD, JSparrow, JDeodorant. PMD demonstrou ser muito útil, é uma ferramenta que não faz refatoração automática, mas demonstra onde estão os code smells e mostra uma descrição que ajuda na refatoração. JSparrow ajudou muito pouco, pois ela é uma ferramenta paga. Já a JDeodorant é uma ferramenta que foi pouco utilizada, pois não tínhamos muito conhecimento nela.

Shotgun Surgery foi o code smell que mais apresentou dificuldade, pois parecia bem simples refatorar, no entanto a refatoração gerou muitos erros, logo, tivemos que desfazer o que foi feito para manter o código utilizável. God Class ficou em segundo lugar, pois no nosso projeto tinham classes muito longas, com muitos métodos e com a complexidade alta.

Feature Envy foi o code smell que apresentou uma dificuldade menor em relação aos code smells detectados pelo JSpirit.

Tabela 3 – Code Smells do projeto.

Nome do Code Smell	Quantidade
Feature Envy	58
Dispersed Coupling	21
Data Class	10
Refused Parent Bequest	4
Shotgun Surgery	3
Brain Method	8
Intensive Coupling	2
God Class	2

2.3 Medição 2 – Após Refatorar Code Smell Feature Envy

Foram refatorados vinte e um code smells deste grupo. A técnica utilizada para refatorar esse code smell foi **Move Method**. Em seguida foi feita a medição das métricas na ferramenta Understand para as Tabelas - 4 até a Tabela - 6. Dentre os 5 atributos de qualidade foi melhorado a métrica CLOC (número de linhas com comentários) e piorado o restante das métricas conforme as retiradas dos code smells desse grupo.

Tabela 4 - As métricas após a refatoração do code smell Feature Envy

Sistema	**Coesão**	**Complexidade**				**Herança**			**Acoplamento**	**Tamanho**			
	LCOM2	ACC	SCC	EVG	MaxNet	DIT	NOC	IFANIN	CBO	LOC	CLOC	NIM	CDL
S1 antes da refatoração	4.001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222
S2 após a refatoração	4.013	414	14.908	1.639	1.161	164	89	158	866	65.615	8.614	1.058	224

2.4 Medição 3 – Após Refatorar Code Smell God Class

Foram refatorados um code smells deste grupo. A técnica utilizada para refatorar foi a **Extract Class**. Dentre os 5 atributos de qualidade foi melhorado as métricas SCC (Somatório da complexidade ciclomática de todos os métodos), CLOC (número de linhas com comentários) e piorado o restante das métricas conforme as retiradas dos code smells desse grupo.

Tabela 5 - As métricas após a refatoração do code smell God class

Sistema	**Coesão**	**Complexidade**				**Herança**			**Acoplamento**	**Tamanho**			
	LCOM2	ACC	SCC	EVG	MaxNet	DIT	NOC	IFANIN	CBO	LOC	CLOC	NIM	CDL
S1 antes da refatoração	4.001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222
S2 após a refatoração	4.013	412	14.892	1.639	1.159	164	89	158	867	65.558	8.641	1.058	224

2.5 Medição 4 – Após Refatorar um conjunto de Code Smells (2 - Data Class, 1- Intensive Coupling, 13 - Dispersed Coupling, 1 - Refused Parent Bequest e 1 - Brain Method)

Foram refatorados um conjunto de code smells deste grupo. A técnica utilizada para refatorar esses code smells foi a **Extract Method, Move Method**. Dentre os 5 atributos de qualidade foi melhorado as métricas SCC (Somatório da complexidade ciclomática de todos os métodos), MaxNet (Nível máximo de aninhamento de construções de controle), DIT (O número de níveis que uma subclasse herda de métodos e atributos de uma superclasse na árvore de herança), NOC (Número de subclasses de uma classe), IFANIN (número imediato de classe base), LOC (Número de linhas de código), CLOC (número de linhas com comentários), NIM (Número de métodos de instância) e piorado o restante das métricas conforme as retiradas dos code smells desse grupo.

Tabela 6 - As métricas após a refatoração de um conjunto de code smells

Sistema	**Coesão**	**Complexidade**				**Herança**			**Acoplamento**	**Tamanho**			
	LCOM2	ACC	SCC	EVG	MaxNet	DIT	NOC	IFANIN	CBO	LOC	CLOC	NIM	CDL
S1 antes da refatoração	4.001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222
S2 após da refatoração	4.066	411	14.867	1.634	1.123	162	85	153	865	64.756	7.896	1.055	224

2.6 Medição 5 – Após a refatoração de 40 code smells do projeto

Nessa seção será apresentada a refatoração de 40 code smells do projeto, assim como foi proposto pela professora Carla Ilane. Foram refatorados 40 code smells como mostra a Tabela - 7.

Tabela 7 – Code Smells que foram refatorados

Nome do Code Smell	Quantidade geral	Quantidade de Refatoração
Feature Envy	58	21
Dispersed Coupling	21	13
Data Class	10	2
Refused Parent Bequest	4	1
Shotgun Surgery	1	0
Brain Method	8	1
Intensive Coupling	1	1
God Class	2	1

Após a refatorações de alguns grupos de code smell foi realizada uma nova medição das métricas utilizando a ferramenta Understand definidas na Tabela - 8. Antes da refatoração foram encontrados 108 code smells, após a refatoração o número caiu para 103. Dentre os 5 atributos de qualidade foi melhorado as métricas SCC, MaxNet, DIT, NOC, IFANIN, LOC, CLOC, NIM. E piorado o restante das métricas conforme as retiradas dos code smells.

Tabela 8 – Medição de todos os atributos após refatorar 40 code smells do projeto.

Sistema	Coesão	Complexidade				Herança			Acoplamento	Tamanho			
	LCOM2	ACC	SCC	EVG	MaxNet	DIT	NOC	IFANIN	CBO	LOC	CLOC	NIM	CDL
S1 antes da refatoração CS = 108	4001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222
S2 após refat. CS = 103	4.066	411	14.867	1.634	1.123	162	85	153	865	64.756	7.896	1.055	224

3 COMPARAÇÃO DOS RESULTADOS

O processo de refatoração melhorou na maioria das métricas. No entanto, outras métricas ficaram piores ou não tiveram alterações. Em síntese, o processo de refatoração é essencial e fundamental para manter o código mais limpo. Além de aprimorar a concepção de um software, essa técnica também evita a deterioração durante o ciclo de vida do código.

É perceptível que a refatoração no projeto do Planejador JavaFF gerou melhorias significativas na maioria dos atributos de qualidade. Na tabela - 8 é feita uma comparação detalhada do projeto em sua versão inicial antes da refatoração e na sua versão final após refatoração dos 40 code smells.

Por fim, ainda existem melhorias e outras técnicas de refatoração que poderão ser aplicadas no projeto. Ficará como possível trabalho futuro a diminuição dos code smells restantes.

3.1 Medição 6 – Medição dos atributos após refatoração de cada Code Smell do Projeto

Coesão apresentou uma variação positiva (65) na métrica LCOM2, subtraindo o valor apresentado na versão final com o valor apresentado na versão inicial.

Complexidade apresentou uma variação positiva (3) na métrica ACC, negativa (-33) na métrica SCC, positiva (8) na métrica EVG, negativa (-28) na métrica MaxNet, subtraindo os valores apresentados na versão final com os valores apresentados na versão inicial.

Herança apresentou uma variação de negativa -1 na métrica DIT, negativa (-4) na métrica NOC, negativa (-4) na métrica IFANIN, subtraindo os valores apresentados na versão final com os valores apresentados na versão inicial.

Acoplamento apresentou uma variação positiva (14) na métrica CBO, subtraindo os valores apresentados na versão final com os valores apresentados na versão inicial.

Tamanho apresentou uma variação de negativo (-780) na métrica LOC, negativo (-790) na métrica CLOC, negativo (-1) na métrica NIM, positivo (2) na métrica CDL, subtraindo os valores apresentados na versão final com os valores apresentados na versão inicial.

As variações negativas significam que o valor final diminuiu em relação ao valor inicial. As métricas CLOC e LOC apresentaram as maiores variações. Portanto o atributo Tamanho foi o que apresentou melhor resultado em relação às demais métricas.

As variações positivas significam que o valor final aumentou em relação ao valor inicial. LCOM2 apresentou a maior variação. Portanto o atributo Coesão foi o que apresentou pior resultado em relação às demais métricas.

Tabela 9 – Medição dos atributos após refatoração de cada Code Smell do Projeto.

Code Smells	**Coesão**	**Complexidade**				**Herança**			**Acoplamento**	**Tamanho**			
	LCOM2	ACC	SCC	EVG	MaxNet	DIT	NOC	IFANIN	CBO	LOC	CLOC	NIM	CDL
S1 antes da refatoração CS = 108	4.001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222
Feature Envy	4.013	414	14.908	1.639	1.161	164	89	158	866	65.615	8.614	1.058	224
God Class	4.013	412	14.892	1.639	1.159	164	89	158	867	65.558	8.641	1.058	224
Conjunto de code smells — Tabela 6	4.066	411	14.867	1.634	1.123	162	85	153	865	64.756	7.896	1.055	224

REFERÊNCIAS

https://refactoring.guru/refactoring/smells

Slide - Code Smells e Refatoração (Disponibilizado no moodle da disciplina de Qualidade de Software) Slides - Code Smells e Refatoração

Vídeo Júlio - Code Smells e ferramentas (Disponibilizado no moodle da disciplina de Qualidade de Software) Vídeo Júlio - Code smells e ferramentas