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	# de classes	# de releases
Planejador JavaFF	65.536	222	0

2 AVALIAÇÃO DO PROJETO

2.1 Medição 1 – Antes de refatorar o projeto

Tabela com a medição das métricas de coesão, acoplamento, complexidade, herança e tamanho, antes do projeto ser refatorado. Foi utilizada a ferramenta Understand na Tabela 2 para apresentar a descrição das métricas do projeto do Planejador JavaFF.

Tabela 2 – Medição dos atributos 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	4001	408	14.900	1.626	1.151	163	89	157	851	65.536	8.686	1.056	222
S1 após refat. CS X													
S1 após refat. CS X													

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 105 Code Smells no projeto. Classificados como: Feature Envy, Dispersed Coupling, Data Class, Refused Parent Bequest, Shotgun Surgery, Brain Method, Intensive Coupling e God Class. Na Tabela 3 está indicando os 5 Code Smells que mais apareceram no projeto.

Tabela 3 – Code smells do projeto.

Nome do Code Smell	Quantidade
Feature Envy	58
Dispersed Coupling	21
God Class	2
Brain Method	8
Refused Parent Bequest	4

2.3 Medição 2 – Após Refatorar Code Smell X

Nessa Seção você deve indicar os valores de todas as métricas da Tabela 2, após refatorar um determinado code smell. Esse code smell deve ser totalmente refatorado até não ser mais detectado pela JSPirit. Você deve também incluir a técnica de refatoração utilizada para retirar o code smell. Isso deve ser feito para cada code smell detectado no projeto. Após a refatoração de cada code smell deve ser realizada uma nova medição na ferramenta Understand. Deve ser realizada também uma análise dos 5 atributos de qualidade e que métricas pioram ou melhoram de acordo com a retirada desses code smells.

2.4 Medição 3 – Após Refatorar Code Smell Y

.....

2.5 Medição Z – Após a refatoração de todos os code smells do projeto

Após todos os code smells refatorados, deverá ser realizada a medição final do projeto conforme as métricas da Tabela 2. Deve também ser feita a análise final se as métricas pioraram ou melhoraram de acordo com a retirada dos code smells.

3 COMPARAÇÃO DOS RESULTADOS

Leia o artigo:
https://www.sciencedirect.com/science/article/pii/S0950584920301142?casa_token=xcwL1BwaRFUAAAAA:wZjXB0Wx-0FiMSpZSzyi0b7iRe7ZJOr8FdwihzEkvzeQHh0Iz6mxPCF769JgRiZ69TyfI5l8BP0

Faça uma comparação dos resultados do seu projeto de acordo com esse artigo.

REFERÊNCIAS

AZEEM, Muhammad. Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. Information and Software Technology, v. 108, p. 115-138, 2019.

SABIR, Fatima. A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems. Software: Practice and Experience, v. 49, n. 1, p. 3-39, 2019.

APÊNDICE A

Incluir possíveis documentos que possam ser gerados no desenvolvimento do sistema.