UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

JANAIDE NOGUEIRA DE SOUSA XIMENES

UMA ABORDAGEM INTERATIVA PARA O PROBLEMA DE SELEÇÃO DE CASOS DE TESTE DE AGENTES RACIONAIS FUNDAMENTADA EM AGENTES RACIONAIS

FORTALEZA – CEARÁ

2020

JANAIDE NOGUEIRA DE SOUSA XIMENES

UMA ABORDAGEM INTERATIVA PARA O PROBLEMA DE SELEÇÃO DE CASOS DE TESTE DE AGENTES RACIONAIS FUNDAMENTADA EM AGENTES RACIONAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Gustavo Augusto Lima de Campos

FORTALEZA – CEARÁ

2020

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Sistema de Bibliotecas

Janaide Nogueira de Sousa, Ximenes.
Uma abordagem interativa para o problema de seleção de casos de teste de agentes racionais fundamentada em agentes racionais [recurso eletrônico] / Ximenes Janaide Nogueira de Sousa. - 2020.
1 CD-ROM: il.; 4 ¾ pol.

CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico com 84 folhas, acondicionado em caixa de DVD Slim (19 x 14 cm x 7 mm).

Dissertação (mestrado acadêmico) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Mestrado Acadêmico em Ciência da Computação, Fortaleza, 2020.
Área de concentração: Algoritmos, otimização e inteligência computacional.
Orientação: Prof. Dr. Gustavo Augusto Lima de Campos.

1. Agentes Inteligentes. 2. Teste. 3. Otimização. I. Título.

GOVERNO DO ESTADO DO CEARÁ
SECRETARIA DE CIÊNCIA, TECNOLOGIA E EDUCAÇÃO SUPERIOR
UNIVERSIDADE ESTADUAL DO CEARÁ
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO - MACC

ATA DE DEFESA PÚBLICA DE DISSERTAÇÃO DE MESTRADO

Aos vinte e sete dias de fevereiro de dois mil e vinte, no(a) sala 5, realizou-se a sessão pública de defesa da dissertação de JANAIDE NOGUEIRA DE SOUSA XIMENES, aluno(a) regularmente matriculado(a) no curso MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO - MACC, Intitulada: **"UMA ABORDAGEM INTERATIVA PARA O PROBLEMA DE SELEÇÃO DE CASOS DE TESTE DE AGENTES RACIONAIS FUNDAMENTADA EM AGENTES RACIONAIS "**. A Banca Examinadora reuniu-se no horário de 11:00h às 12:20 horas, sendo constituída por: Prof. Dr. GUSTAVO AUGUSTO LIMA DE CAMPOS (Orientador e Presidente da Banca/UECE), Profa. Dra. MARIELA INES CORTES (UECE) e Profa. Dra. FRANCISCA RAQUEL DE VASCONCELOS SILVEIRA (IFCE). Inicialmente o(a) mestrando (a) expôs seu trabalho e a seguir foi submetido(a) à arguição pelos membros da Banca, dispondo cada membro de tempo para tal. Finalmente a Banca reuniu-se em separado e concluiu por considerar o(a) mestrando(a) Aprovada, por sua dissertação e sua defesa pública. Eu, Prof. Dr. GUSTAVO AUGUSTO LIMA DE CAMPOS, orientador(a) e presidente da banca, lavrei a presente ata que será assinada por mim e os demais membros. Fortaleza, 27 de Fevereiro de 2020.

Prof. Dr. GUSTAVO AUGUSTO LIMA DE CAMPOS
(Orientador e Presidente da Banca/UECE)

Profa. Dra. MARIELA INES CORTES
(UECE)

Profa. Dra. FRANCISCA RAQUEL DE VASCONCELOS SILVEIRA
(IFCE)

Mestrado Acadêmico em Ciência da Computação
UECE
MACC

À Deus, por me proporcionar este momento. À minha filha, meu amor por você, me fez mais forte. Aos meus pais, meus irmãos e ao meu esposo, por acreditarem e me apoiarem.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, em todos os momentos é o maior mestre que alguém pode conhecer.

Agradeço aos meus pais, Gerardo e Liduina, que com muito sacrifício e renúncia, me trouxeram até aqui. Me ajudaram a cuidar da minha filha, e compreenderam minha ausência durante todo este período. Sem vocês essa vitória não teria sido possível!

Obrigada meus irmãos, Ronieri e Roney, que nos momentos de minha ausência, com muito carinho ajudaram a cuidar da minha filha.

A minha filha, Ana Clara, que mesmo tão pequena, por vezes abriu mão de meus cuidados por tanto tempo sem reclamar, por tentar entender e aceitar minha ausência por um longo período, sempre com muito amor e um abraço bem apertado em todos os meus retornos.

Ao meu esposo, Rhyan, que sempre me apoiou, e compreendeu minha ausência como esposa e como mãe, me incentivou a enfrentar os desafios e a superar os obstáculos que surgiram. Agradeço pelos conselhos, companhia e amor doados em todo esse tempo. Agradeço por estar ao meu lado nos melhores e piores momentos de minha vida.

Agradeço a todos os professores e funcionários da UECE envolvidos no programa de Mestrado Acadêmico em Ciências da Computação.

Agradeço especialmente ao meu orientador, professor Gustavo Augusto, sua paciência, incentivo, apoio, confiança, conselhos e ajuda foram essenciais para que, apesar de todos os obstáculos, eu conseguisse produzir esse trabalho. Considero-o mais que um professor e orientador. Considero-o um amigo e exemplo de pessoa.

Agradeço também as professoras Mariela e Raquel, por aceitarem o convite para participação em minha Banca Examinadora.

"Paciência e perseverança tem o efeito mágico de fazer as dificuldades desaparecerem e os obstáculos sumirem."

(John Quincy Adams)

RESUMO

A Inteligência Artificial é um ramo da Ciência da Computação cada vez mais presente nas mais diversas áreas do conhecimento, tendo como premissa a utilização das mais diversas técnicas computacionais para a resolução dos mais diversificados problemas. Assim sabe-se que dentre eles a utilização de agentes racionais consiste em uma tecnologia promissora principalmente para a resolução de problemas de seleção de casos de testes. Este trabalho tem como objetivo principal apresentar uma abordagem para otimizar o problema de seleção de casos de testes de agentes racionais, por meio da otimização interativa, mais especificamente, na interação entre o teste do agente e o usuário. O agente que seleciona os casos de testes baseia-se na estrutura do programa agente orientado por utilidade. Assim concebendo uma abordagem para o problema de seleção de unidades de casos de teste para programas de agentes artificiais inteligentes, que sejam capazes de resolver problemas de tomada de decisão em ambientes de tarefas não triviais que podem ser representados por meio de grafos. A abordagem proposta no trabalho está fundamentada na noção de agentes racionais integrada com a otimização interativa para a geração e seleção dos casos de teste gerados. Mais especificamente, a abordagem consiste em uma extensão ao *framework* para o teste de agentes racionais inicialmente desenvolvido em (SILVEIRA, 2013). Esta extensão consiste na inserção de novos componentes ao *framework*, capacitando-o à realização de novas formas de teste. O primeiro componente inserido no *framework* consiste em um gerador de casos deteste (GERADOR) capaz de gerar objetivamente, conforme o desejo do Projetista do agente em teste (Agent), representações em grafos de ambientes de tarefas para medir o desempenho de Agent, interagindo com um programa ambiente (Amb) que incorpora estas representações. O segundo componente consiste na capacidade de monitorar (MOA) os episódios de uma história do agente Agent no ambiente Amb, e descobrir falhas e indicar qual(is) módulo(s) de processamento de informação de Agent está(ão) causando a falha em Amb. Embora tenha sido gerado resultado para o mundo do aspirador de pó, pode-se afirmar que a arquitetura proposta é genérica o suficiente para ser aplicada em diversos outros ambientes com padrões ambientais.

Palavras-chave: Agentes Racionais. Casos de Testes. *Framework*. Otimização.

ABSTRACT

Artificial Intelligence is a branch of Computer Science that is increasingly present in the most diverse areas of knowledge, with the premise of using the most diverse computational techniques to solve the most diverse problems. Thus, it is known that among them the use of rational agents consists of a promising technology mainly for the resolution of test case selection problems. This work has as main objective to present an approach to optimize the problem selection of rational agent test cases, through interactive optimization, more specifically, in the interaction between the agent test and the user. The agent that selects the test cases is based on the structure of the utility-oriented agent program. Thus conceiving an approach to the problem of selecting test case units for intelligent artificial agent programs, which are able to solve decision-making problems in non-trivial task environments that can be represented by graphs. at work it is based on the notion of rational agents integrated with the interactive optimization for the generation and selection of the generated test cases. More specifically, the approach consists of an extension to the framework for testing rational agents initially developed in (SILVEIRA, 2013). This extension consists of the insertion of new components to the framework, enabling it to perform new forms of testing. The first component inserted in the textit framework consists of a detest case generator (GENERATOR) capable of objectively generating, according to the designer's desire for the agent under test (Agent), graphical representations of task environments to measure the performance of Agent, interacting with an environment program (Amb) that incorporates these representations. The second component consists of the ability to monitor (MOA) the episodes of an Agent agent story in the Amb environment, and discover faults and indicate which Agent information processing module (s) is causing the failure in Amb. Although results have been generated for the vacuum cleaner world, it can be claim that the proposed architecture is generic enough to be applied in several other environments with environmental standards.

Keywords: Rational Agents. Test Cases. *Framework*. Optimization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AGI	Algoritmo Genético Interativo
BDI	*Belief, Desire, Intentionl*
CEI	Computação Evolucionária Intereativa
CT	Caso de Teste
MAPC	*Multi-Agent Programming Contest*
PEAS	*Performance, Environment, Actuators, Sensors*

LISTA DE SÍMBOLOS

μ	Valor de pertinência que indica o grau de adequação de cada regra ao estado atual do sistema
Ω	Conjunto de descrições de ambientes de tarefa factíveis de instanciar em Amb e testar Agent
$\sum_{n}^{1}$	Somatório da quantidade de salas sujas
S	Sequência finita de estados
V	Conjunto de vértice
E	Conjunto de arestas
R	Conjunto de números reais empregados para avaliar os recursos nos vértices
C	Subgrafo conectado

SUMÁRIO

1 INTRODUÇÃO

Este trabalho visa conceber uma abordagem para o problema de seleção de unidades de casos de teste (casos de teste) para programas de agentes artificiais inteligentes, ou seja, capazes de resolver problemas de tomada de decisão em ambientes de tarefas não triviais e que podem ser representados por meio de grafos.

De maneira genérica, os problemas de tomada de decisão dos agentes podem requisitar a solução de um ou mais subproblemas no ambiente de tarefa como, por exemplo, aqueles denominados por: monitoramento, reconhecimento de padrões, classificação, diagnóstico, controle, planejamento e otimização.

Mais especificamente, por não trivial, entenda-se que estes subproblemas ocorrem em ambientes de tarefas específicos que podem ser classificados como: conhecidos, discretos, sequenciais, estáticos ou dinâmicos, agente único ou multiagentes, parcialmente observáveis e, como consequência, não determinísticos.

Por exemplo, vale ressaltar os ambientes de tarefas descritos nos mundos do monstro Wumpus e dos agentes artificiais aspirador de pó, e outros problemas do mundo real como, por exemplo, o cálculo de rotas em ambientes parcialmente observáveis. Em comum, o processo de tomada de decisão nestes ambientes requisita o projeto de cursos de ações racionais, ou seja, que realizem objetivos, descritos de acordo com uma medida de avaliação de desempenho.

Grande parte da literatura sobre agentes inteligentes está focada no processo de projeto de programas de agentes artificiais racionais. Especificamente relacionado com os objetivos deste trabalho, vale ressaltar os trabalhos mais fundamentais sobre arquiteturas de agentes inteligentes desenvolvidos por (NORVIG; RUSSELL, 2014) e (WOOLDRIDGE, 2002).

Norvig e Russell (2014) especificaram quatro tipos básicos de programas de agentes racionais que podem ser adaptados para resolver problemas em diversos tipos de ambientes de tarefa difíceis. Estes programas incorporam os princípios subjacentes a quase todos os sistemas inteligentes, são classificados levando-se em consideração as informações empregadas no processo de tomada de decisão e as etapas componentes deste processo.

Wooldridge (2002) descreveu formalmente as arquiteturas abstratas do agente reativo e do agente com estado interno, considerando três Subsistemas (módulos) de processamento de informação componentes. Em seguida, destacou quatros maneiras diferentes de se concretizar projetos de agentes racionais, ou seja, os agentes lógicos, BDI (*Belief, Desire, Intentionl*) e com arquiteturas em camadas.

Por outro lado, em uma proporção bem menor, diversos trabalhos têm sido desenvolvidos com foco, principalmente, no teste dos programas destes agentes artificiais. Também relacionado com os objetivos deste trabalho, vale ressaltar os trabalhos sobre geração de casos de teste, e geração e seleção automática de casos de teste desenvolvidos por (ATHAMENA; HOUHAMDI, 2012), (NGUYEN *et al.*, 2009) e outros, e (SILVEIRA, 2013) e outros.

Athamena e Houhamdi (2012) desenvolveram uma abordagem para testes de aceitação de *software* orientado a meta. A abordagem especifica um processo de teste que visa complementar a metodologia de desenvolvimento orientada a objetivos Tropos, fornecendo uma maneira sistemática de derivação de casos de teste a partir da análise da meta do agente em seu ambiente de tarefa.

Em Nguyen *et al.* (2009) desenvolveram um procedimento que serve para uma ampla gama de contextos de casos de teste, o qual procura por casos de teste que exigem dos agentes, mas que não estão aparentes para seus desenvolvedores. Mais especificamente, o trabalho consiste em uma primeira abordagem na literatura que emprega otimização evolutiva para gerar casos de teste exigentes para agentes autônomos.

Já Silveira (2013) formulou o problema de seleção de casos de teste de agentes racionais como um problema de otimização sem restrições e uma abordagem com solução fundamentada em um agente orientado por utilidade (Thestes), que seleciona casos de teste empregando meta-heurísticas baseadas em população e simulações das interações entre o agente testado e seu ambiente de tarefas. A abordagem foi empregada para selecionar casos de teste para programas de agentes racionais reativos e com estado interno em ambientes de tarefa com observabilidade total e parcial.

Estes trabalhos foram desenvolvidos com foco no teste caixa-preta/unitário de agentes, isto é, os casos de teste são selecionados examinando-se os aspectos funcionais dos agentes por meio de uma interface, sem levar em consideração a estrutura interna dos agentes em teste, conforme definidas por Norvig e Russell (2014); Wooldridge (2002). Nos experimentos que foram realizados buscou-se, principalmente, avaliar a Racionalidade das ações e dos planos executados pelo agente em seu ambiente de tarefa.

Apesar dos referenciais disponíveis para orientar o projeto de agentes racionais, existem poucas técnicas de testes propostas para validá-los. Sabe-se que essa validação depende dos casos de teste selecionados, os quais devem gerar informações que permitam a identificação dos componentes na estrutura do programa do agente artificial testado que estão causando o desempenho insatisfatório. Entretanto, nem sempre os melhores casos estão disponíveis a priori

e, dependendo do ambiente de tarefa do agente, pode existir uma grande quantidade de casos a serem observados.

Este trabalho pode ser visto como uma extensão do trabalho desenvolvido por Silveira (2013) objetivando aumentar o poder de seleção da abordagem, à medida em que insere no processo aspectos relacionados ao teste caixa-branca/unitário de programas de agente. Assim, além de considerar a racionalidade das ações e planos do agente em teste em seu ambiente de tarefas, a abordagem proposta considera também a estrutura interna dos programas de agentes racionais no processo de seleção de casos de teste.

Portanto, a extensão proposta deve ser capaz de selecionar casos de teste em que o agente em teste produz baixo desempenho, mas que, de acordo com o interesse do projetista, foi causado por um ou mais dos seus módulos de processamento de informações. Mais especificamente, este trabalho propõe uma extensão nos modos e momentos de interação do projetista do agente em teste com o agente orientado por utilidade, denominado Thestes, desenvolvido por Silveira (2013) para a seleção de conjuntos de casos de teste em que o agente em teste produz baixo desempenho.

Na versão desenvolvida, o projetista interage com o agente Thestes no início do processo de seleção de casos de testes, fornecendo para Thestes diversos parâmetros associados à meta-heurística para a busca de conjuntos de casos de teste e à simulação das interações do agente em teste com seu ambiente de tarefas, e no final do processo de busca, recebendo de Thestes um conjunto de caso de testes selecionado.

Na versão a ser concebida, além de fornecer outras informações para Thestes no início do processo de seleção, propõe-se que o projetista interaja com o agente Thestes durante o processo de seleção, nos moldes das estratégias de Otimização Interativa existentes (MIETTINEN, 1999), (ATHAMENA; HOUHAMDI, 2012).Acrescentando outros aspectos com o objetivo de melhorar ainda mais o poder de seleção da abordagem proposta.

Este trabalho busca apresentar uma abordagem com otimização interativa para a seleção de casos de teste de agentes racionais, tendo em vista a fadiga do usuário, propõe-se um agente que simule o comportamento do usuário, podendo até vir a substituí-lo.

1.1 MOTIVAÇÃO

Devido às propriedades peculiares dos agentes racionais (propriedades reativas, de memória, orientação por metas e por utilidade, e de aprendizagem) e de seus ambientes de tarefa,

o teste unitário de cada agente em um sistema é uma tarefa não trivial. Algumas abordagens focam a produção de artefatos de teste para dar suporte às metodologias de desenvolvimento de sistemas de agentes. A maioria dos trabalhos desenvolvidos consiste em adaptações das técnicas de testes de *software* convencional.

No caso dos agentes racionais, sabe-se que estas adaptações devem buscar avaliar a racionalidade das ações e dos planos executados pelo agente em seu ambiente de tarefa. O pressuposto na maioria dos trabalhos é que uma boa avaliação do agente depende dos casos de testes selecionados. Bons casos de teste devem providenciar a geração de informações sobre o desempenho insatisfatório dos componentes na estrutura do agente e do funcionamento destes componentes de maneira integrada.

Entretanto, bons casos de teste não estão disponíveis para o projetista. Eles devem ser descobertos em um espaço de casos de teste que, dependendo do problema de seleção, pode ser de tamanho muito grande e cheio de soluções de baixa qualidade, em termos de geração de informações relevantes para que o projetista possa identificar os módulos de processamento de informações que estão causando o baixo desempenho no agente em teste.

Em seu trabalho Nguyen (2008) afirma que tem sido escasso os estudos que visam propor métodos e técnicas eficazes e que garantam a qualidade dos sistemas baseados em agentes.

Enquanto os agentes estão cada vez mais comuns, existe uma necessidade imprescindível para orientações das particularidades do processo de teste de agentes (HOUHAMDI, 2011). Some-se a isto, ainda existe uma lacuna em termos de técnicas de testes interativos que possam ser aplicadas especificamente para testar agentes inteligentes.

1.2 OBJETIVOS

Neste capítulo apresentam-se os objetivos gerais e especifícos que visam serem alcançados na abordagem descrita.

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo principal apresentar uma abordagem para otimizar o problema de seleção de casos de testes de agentes racionais, por meio da otimização interativa, mais especificamente, na interação entre o teste do agente e o usuário. O agente que seleciona os casos de testes baseia-se na estrutura do programa agente orientado por utilidade. A abordagem proposta consiste em uma extensão e em um aprimoramento da abordagem

desenvolvida por Silveira (2013), por meio da inserção de processos de auxílio à busca de soluções utilizadas em técnicas de otimização interativa.

Esta abordagem visa incorporar as preferências humanas antes e durante o processo de teste. Para cada caso de teste são realizadas as interações entre o projetista que envia o agente a ser testado e as informações que são necessárias para a realização do teste, o agente Thestes, executa os testes fazendo com que o agente testado interaja com o ambiente gerando assim casos de teste e seleciona conjuntos de casos de teste satisfatórios. Visando reduzir o problema da fadiga humana, foi proposta a utilização do modo automático de geração de casos de testes.

1.2.2 Objetivos Específicos

Mais especificamente, a concretização da abordagem envolve a realização dos seguintes objetivos específicos:

a) Conceber uma ferramenta para a derivação sistemática de casos de teste, considerando a medida de avaliação de desempenho do agente em teste e outras informações sobre as relações existentes entre os componentes da estrutura interna do agente e seu ambiente de tarefas, e que permita ao projetista interagir adequadamente com a abordagem apresentando conjuntos de casos de teste tanto no início como durante o processo de busca.

b) Conceber uma ferramenta para a avaliação de casos de teste, considerando a medida de avaliação de desempenho e as informações sobre as relações entre estrutura interna e ambiente de tarefas do agente em teste, e que permita ao projetista interagir adequadamente com a abordagem avaliando conjuntos de casos de teste durante o processo de busca.

c) Conceber um sistema agente capaz de incorporar a medida de avaliação de desempenho e as informações sobre as relações entre estrutura interna e ambiente de tarefas, automatizando parte das interações e, consequentemente, diminuindo o número de interações que o projetista mantém com a abordagem de teste durante o processo de busca.

1.3 ESTRUTURA DO TRABALHO

O trabalho está organizado em seis capítulos, incluindo a presente introdução. De modo geral, o restante dos capítulos são resumidos abaixo:

a) **Capítulo 02 - Fundamentação Teórica:** são apresentadas as principais definições que permeiam a pesquisa desenvolvida. Inicialmente, aborda-se os princípios que envolvem a Otimização Interativa. Logo após explana-se sobre os Algoritmos Evolucionários, evidenciando os Algoritmos Genéticos. Posteriormente apresentam-se os conceitos de agentes inteligentes e os ambientes em que os agentes podem estar inseridos, bem como a estrutura dos agentes racionais. Em seguida explana-se os fundamentos relacionados ao Teste de Agentes, abordando os principais tipos de testes, incluindo os testes utilizados no presente trabalho. Também, é apresentado um breve conceito de geração de casos de teste. Em seguida discute-se os conceitos que envolvem a Lógica de Fuzzy. Por fim, apresentam-se as particularidades referentes ao testes interativo de agentes.

b) **Capítulo 03 - Trabalhos Relacionados:** são discutidos os principais trabalhos relacionados ao Teste de Agentes e com o teste com otimização interativa.

c) **Capítulo 04 - Metodologia:** Descreve a abordagem proposta, apresentando a abordagem de Silveira (2013), o agente gerador de casos de testes e descreve o agente de monitoramento e diagnóstico de falhas.

d) **Capítulo 05 - Resultados:** Além de descrever o ambiente de tarefas e a estrutura dos agentes a serem testados, apresenta também as funcionalidades da ferramenta para os modos de interação com o Projetista. São apresentados resultados de alguns dos experimentos para cada um dos tipos de interação.

e) **Capítulo 06 - Conclusões e Trabalhos Futuros:** são discutidas as as principais contribuições alcançadas, trabalhos resultantes, limitações encontradas durante a pesquisa e por conseguite são apresentadas oportunidades de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

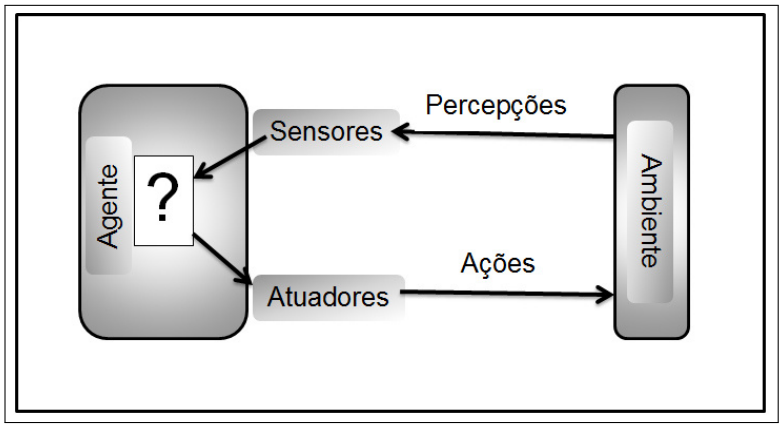
Este capítulo realiza um estudo mais aprofundado sobre os principais conceitos necessários para um melhor entendimento da abordagem proposta.

2.1 AGENTES

A concepção de agentes inteligentes foi inicialmente introduzida na década de 70 (HEWITT, 1977). Na literatura há diversas definições para agentes. Para Magedanz, Rothermel e Krause (1996) um agente é um componente autônomo de software e que é responsável por uma tarefa. Amandi (1997) complementa afirmando que um agente possui comportamento autônomo que permite tomar suas decisões para agir, considerando as mudanças ocorridas no ambiente em que o mesmo atua e o propósito de atingir seus objetivos.

Um agente é meramente algo que age, este tipo de programa diferencia-se de todos os outros pois, opera sob controle autônomo, percebe seu ambiente, persiste por um período de tempo prolongado, adapta-se a mudanças e é capaz de criar e perseguir metas (NORVIG; RUSSELL, 2014). A Figura 1 ilustra um agente percebendo o ambiente por meio dos sensores e agindo no ambiente por meio dos atuadores.

Figura 1 – Agente interagindo com o ambiente por meio de seus sensores e atuadores.



Fonte – Adaptado de Russell e Norvig (2014)

O comportamento do agente é induzido pela associação entre o domínio da aplicação externa e um modelo interno, que constitui-se de uma base de conhecimento e um motor de inferência (TECUCI, 1998). Segundo Norvig e Russell (2014) o agente percebe seu ambiente por intermédio dos sensores e age sobre o ambiente que o mesmo está inserido por meio dos atuadores. Sendo assim o agente verifica uma sequência de percepções (histórico completo de

todas as percepções do agente) e escolhe qual será a próxima ação.

2.1.1 Ambientes

De acordo com Norvig e Russell (2014) ao projetar um agente independentemente da aplicação deste, a primeira etapa deve especificar o ambiente de maneira tão completa quanto possível. O ambiente de tarefa é caracterizado pelos componentes do acrônimo PEAS (*Performance, Environment, Actuators, Sensors* - Desempenho, Ambiente, Sensores e Atuadores).

Norvig e Russell (2014) classifica os ambientes quanto a suas particularidades:

- **Completamente observável ou parcialmente observável:** um ambiente de tarefa é completamente observável quando, os sensores permitem todo o acesso ao estado do ambiente. Entretanto, um ambiente é parcialmente observável se apenas partes do estado podem ser acessadas, os sensores são imprecisos ou houver ruídos.
- **Agente único ou multiagente:** um ambiente é classificado como um ambiente de tarefa com agente único se há somente um agente no ambiente, porém quando existe interações entre agentes são caracterizados como multiagente.
- **Determinístico ou Estocástico:** em um ambiente determinístico o estado seguinte do ambiente é totalmente determinado pelo estado atual e pela ação executada pelo agente. Quando não é presumível detectar o estado que o ambiente assumirá após a execução de uma ação, este é caracterizado como estocástico.
- **Episódico ou Sequencial:** num ambiente episódico em cada episódio o agente recebe uma percepção e em seguida executa uma única ação, onde a experiência do agente é separada em episódios atômicos, caso contrário, se a decisão atual afetar os próximos episódios, o ambiente é sequencial.
- **Estático ou dinâmico:** um ambiente é dinâmico se o ambiente for capaz de se alterar enquanto um agente está deliberando; caso contrário, o ambiente é estático. Entretanto, se o ambiente não se alterar com o decorrer do tempo, porém a medida de desempenho do agente se alterar, dispomos de um ambiente semidinâmico.
- **Discreto ou contínuo:** a diferenciação entre ambiente discreto e contínuo é aplicado tanto para o estado do ambiente, como para as ações e percepções como também para para o tratamento do tempo. Em um ambiente contínuo é possível atribuir inúmeros estados. No ambiente discreto o número de estados é finito.
- **Conhecido ou desconhecido:** esta classificação não é referente ao ambiente em si, todavia

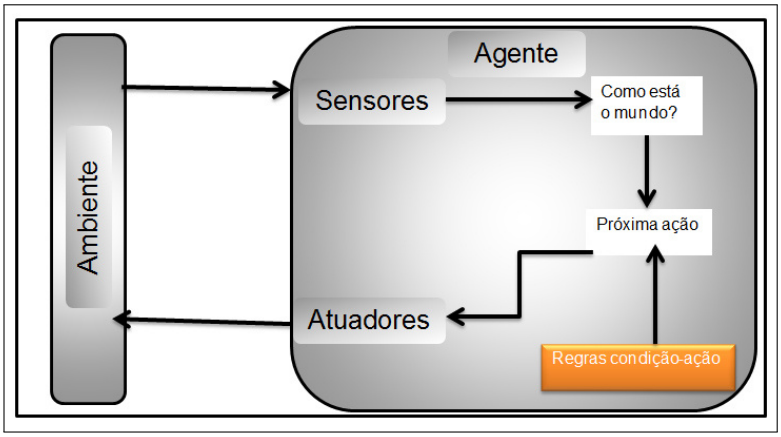
ao estado de conhecimento do agente em relação as "leis da física"no meio ambiente. Em um ambiente conhecido são proporcionadas saídas ou probabilidades de saída para todas as ações. Se o ambiente é desconhecido, o agente irá aprender o funcionamento, com o propósito de tomar boas decisões.

2.1.2 Estrutura de Agentes Racionais

2.1.2.1 Agentes Reativos Simples

Agente que não leva em consideração o histórico de percepções, ou seja, seleciona a ação a partir da percepção atual do ambiente. Comportamentos reativos simples acontecem mesmo em ambientes mais complexos, como é o caso de um motorista de táxi automatizado (NORVIG; RUSSELL, 2014). Conforme por ser observado na Figura 2, em que o agente percebe o ambiente através dos sensores e a partir da percepção os atuadores executam a ação.

Figura 2 – Agente reativo simples.



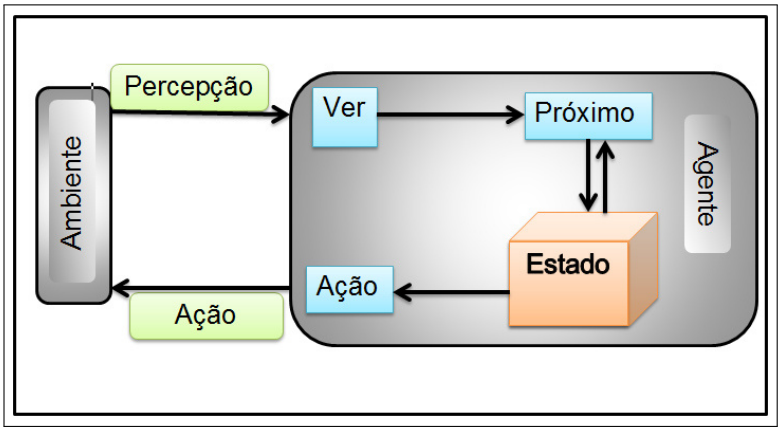
Fonte – Adaptado de Russell e Norvig (2014)

Através dos sensores, o agente recebe os dados do ambiente que são uma sequência finita de estado, **S = {s1, s2, s3, ..., sn}**, um subsistema de percepção verifica cada estado {S*} e mapeia um conjunto de percepções definidas **P = {p1, p2, p3, ..., px}**; um subsistema de tomada de decisão, **Ação = P -> A**, processa o conjunto de percepções **P*** e escolhe uma ação no conjunto de regras condição-ação, **A = {a1, a2, a3, ..., am}** e por intermédio dos atuadores são executadas as ações.

2.1.2.2 Agentes Reativos Baseados em Modelos

Os agentes reativos baseados em modelos possuem uma estrutura apropriada às situações em que o ambiente de execução, nos quais estão introduzidos é observável, entretanto a geografia desse ambiente não é conhecida (WEISS, 1999). A tomada de decisão deste tipo de agente é leva em consideração essas informações (WOOLDRIDGE, 2002).

Figura 3 – Agente reativo baseado em modelos.



Fonte – Adaptado de Wooldridge (2002)

A Figura 3 apresenta os módulos propostos para essa arquitetura. O subsistema de percepção realiza o mapeamento do estado do ambiente em **S*** em uma percepção **P***, o **Próximo = EI x P -> EI**, que atualiza o estado interno, onde é mapeado em **P** e o estado interno atual em **EI = { ei1, ei2, ei3, ...,ein}**, em um estado interno corrente, este é processado, **Ação: EI -> A**, a fim de selecionar uma possível ação em **A**, por meio dos atuadores, o agente envia a ação que foi selecionada para o ambiente (GONÇALVES, 2009).

Figura 4 – Arquitetura de agente reativo baseado em modelos.

Função AGENTE-BASEADO-EM-MODELOS(*percepção*) **retorna** uma ação
persistente: *estado*, a concepção do agente do estado atual do mundo
modelo, uma descrição de como o próximo estado depende do estado atual e da ação
regras, um conjunto de regras condição-ação
ação, a ação mais recente, inicialmente nenhuma

estado ← ATUALIZAR-ESTADO (*estado, ação, percepção, modelo*)
regra ← REGRA-CORRESPONDENTE (*estado, regras*)
ação ← *regra*, AÇÃO
retornar *ação*

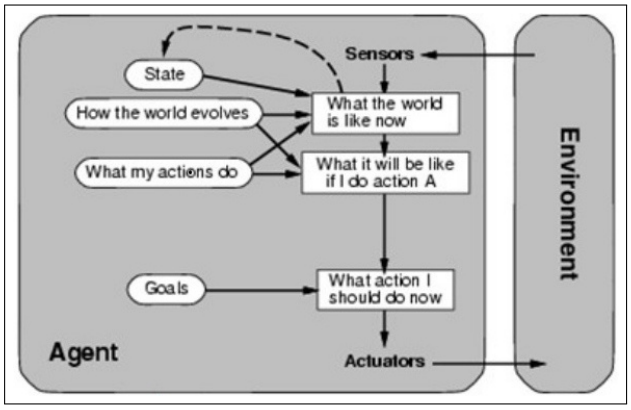
Fonte – Russell e Norvig (2014)

Pode-se observar na Figura 4 que um agente baseado em modelos mantém o atual estado do mundo utilizando um modelo interno. Logo após, o agente escolhe uma ação da mesma maneira que o agente reativo simples (NORVIG; RUSSELL, 2014) .

2.1.2.3 Agentes Baseados em Objetivos

Da mesma maneira que o agente necessita de uma descrição atual do ambiente, também necessita de um espécie de informação em relação ao objetivo que descreva situações desejáveis (NORVIG; RUSSELL, 2014).

Figura 5 – Arquitetura de agente baseado em objetivos.



Fonte – Russell e Norvig (2014)

A Figura 5 demonstra que o agente monitora o estado do mundo, assim como um conjunto de objetivos que está buscando atingir e elege uma ação que no termino ocasionará a realização dos objetivos (NORVIG; RUSSELL, 2014).

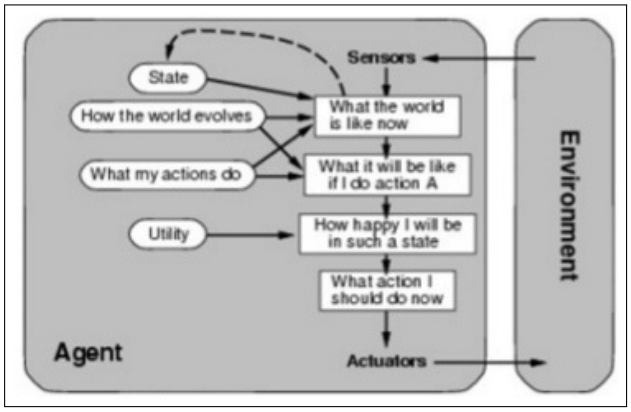
Por exemplo, se um agente tanque de guerra em um jogo de batalha, tendo como objetivo eliminar todos os inimigos sem ser atingido, e este possuindo a capacidade de atirar, virar à esquerda, virar à direita, seguir em frente ou recuar. Complementarmente o agente é capaz de utilizar informações, para selecionar uma única ação ou uma sequência delas até encontrar maneira de alcançar o objetivo.

2.1.2.4 Agentes Baseados em Utilidade

Individualmente os objetivos não são realmente eficientes para gerar um comportamento de alta qualidade quando inseridos na maioria dos ambientes. Os objetivos apenas permitem uma distinção binária crua entre "estados felizes"e "infelizes", enquanto uma medida de desempenho mais geral deve admitir uma comparação entre diferentes estados, de acordo

com o grau exato de felicidade que proporcionariam ao agente. O termo Utilidade substitui esta distinção "feliz"(NORVIG; RUSSELL, 2014).

Figura 6 – Arquitetura de agente baseado em utilidade.



Fonte – Russell e Norvig (2014)

Na Figura 6 é apresentado a estrutura do agente baseados em modelo e orientado para utilidade. Este utiliza um modelo do mundo e adiciona uma função utilidade que mensura as preferências entre estados do ambiente. Um agente racional baseado em utilidade escolhe a ação que maximiza a utilidade esperada dos resultados da ação, ou seja, a utilidade que o agente espera alcançar (NORVIG; RUSSELL, 2014).

2.2 TESTES DE AGENTES

Para o teste de agentes, são considerados quatro tipos de testes: testes de agente único, teste de integração, teste de sistema e testes de aceitação (NGUYEN; PERINI; TONELLA, 2009). Zina (2011) descreve os objetivos de cada tipo:

- **Testes de agente único:** ao testar um único agente, tem-se como objetivo testar a sua funcionalidade interior e capacidade do agente para cumprir os objetivos, de observar e executar ações no ambiente.
- **O teste de integração:** tem por objetivo, se certificar de que um grupo de agentes e o ambiente trabalham corretamente em conjunto.
- **Teste de sistema:** envolve a garantia de que todos os agentes do sistema trabalham em conjunto como pretendido.
- **O teste de aceitação:** testa apenas sistemas multiagentes no ambiente de execução do cliente, com o intuito de verificar se ele se encontra com as características das partes interessadas.

Watkins (2000) classifica os testes como: teste caixa preta ou funcional, teste caixa branca ou estrutural e teste caixa cinza.

- **Teste Caixa preta:** O testes de caixa preta visam identificar as funcionalidades e a aderência aos requisitos, em uma visão externa ou do usuário, sem se basear em qualquer conhecimento da lógica interna e do código do componente a ser testado. Para Watkins (2000) são testes planejados a partir da especificação abstrata, ou seja, não há conhecimento do código.
- **Teste Caixa Branca:** Os testes caixa branca avaliam as cláusulas de código, as configurações, a lógica interna do componente e outros elementos técnicos que possam ser relevantes (RIOS, 2006). São testes definidos a partir do conhecimento de detalhes da implementação.
- **Teste Caixa Cinza:** É basicamente um teste caixa preta, fundamentado no conhecimento limitado dos detalhes da implementação (CARTAXO, 2006).

2.3 GERAÇÃO DE CASOS DE TESTE

O processo de teste apoia a concepção de casos de teste, a qual é definida como uma especificação detalhada de testes, incluindo algumas informações importantes sobre o teste, como pré-condições, pós-condições, dados de entrada, interdependência, etc. (BASTOS *et al.*, 2007).

Um conjunto de casos de teste determinado para uma certa aplicação é denominado de *suíte* de teste. A execução dos casos de teste inclui estabelecer as condições necessárias, fornecendo as entradas de casos de teste, observando os resultados, comparando-os com os resultados esperados, e assegurar a existência de pós-condições esperadas para determinar se o teste passou. De tudo isto, torna-se claro que os casos de teste são valiosos e imprescindíveis, pelo menos tão valiosos quanto o código-fonte. Os casos de teste precisam ser desenvolvidos, revisados, utilizados, gerenciados e salvos (JORGENSEN, 2016).

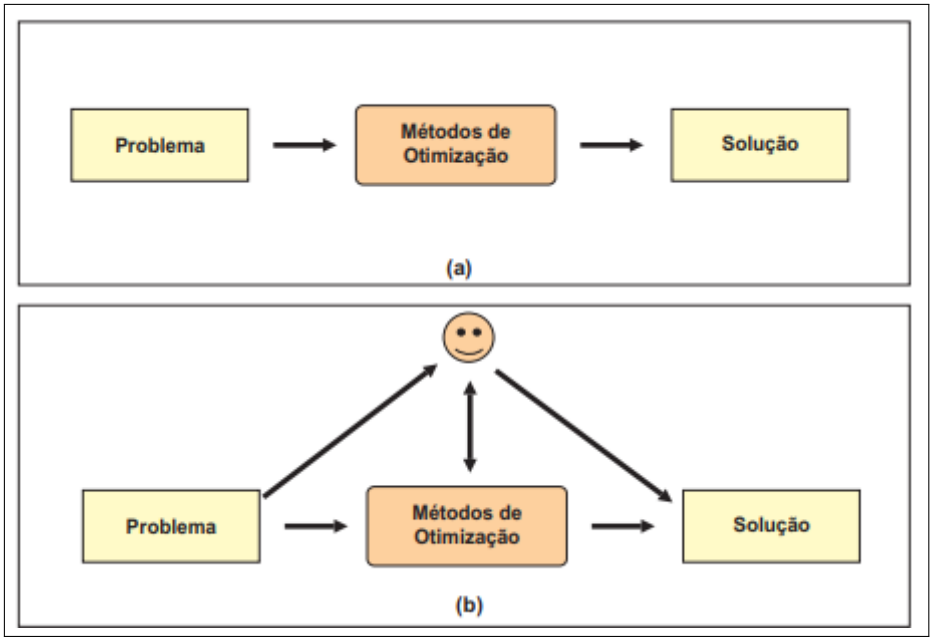
2.4 TESTES INTERATIVOS DE AGENTES

Levando em consideração a complexidade em incluir um tomador de decisão nos problemas de otimização, principalmente quando refere-se a teste de agentes racionais, pode-se destacar a Otimização Interativa. A Otimização Interativa propõe inserir o ser humano de maneira protagonista no processo de avaliação das soluções. Harman, Mansouri e Zhang (2009), afirmam

que a inserção do ser humano neste processo, visa a incorporação de seu conhecimento e de diversos aspectos psicológicos.

Vários problemas de otimização do mundo real não podem ser adequadamente resolvidos utilizando apenas processos automáticos. Como por exemplo, os horários de aula, onde além dos conflitos de salas de aula há a preferência de horário dos professores. Sistemas centralizados em pessoas são pertinentes para problemas complexos e dinâmicos, pois proporcionam uma fusão entre o conhecimento tácito e a simulação computacional (PARMEE *et al.*, 2006). Como extensão da Otimização Interativa, surge a CEI (Computação Evolucionária Intereativa), a qual possui dois fragmentos chaves, que por sua vez são a busca computacional através de estratégias evolucionárias bioinspiradas e a avaliação humana.

Figura 7 – (a) Modelo de otimização tradicional. (b) Modelo de otimização interativa.



Fonte – (FERREIRA, 2006)

Segundo Nascimento (2003), no fim da década de 1980, surgiram os primeiros sistemas interativos de otimização e, apesar de não generalizarem o conceito de otimização interativa, já explicavam as noções de unir a inteligência humana com o poder computacional dos métodos tradicionais. Entretanto, foi a partir do ano de 1995 que os conceitos envolvendo interação homem - computador para problemas de otimização começaram a ser utilizados no campo científico. A Figura 7 ilustra dois modelos de otimização, o primeiro o tradicional e o segundo o modelo de otimização interativa.

Entretanto na maioria de suas aplicações não há a capacidade do ser humano interferir

no processo de busca. Logo, o AGI (Algoritmo Genético Interativo), mantém os mesmos conceitos da versão tradicional do Algoritmo Genético, com exceção do quesito relativo à avaliação de soluções, pois nessa abordagem a responsabilidade de julgamento das soluções é confiada ao usuário ao invés de apenas uma função matemática em especial (TAKAGI, 2001).

É possível afirmar que usuário ao interferir nas avaliações, consequentemente, os resultados podem ser diferentes. Vale a pena ressaltar que para a abordagem proposta, auxilia para que os casos de teste não sejam muito similares, ampliando assim, a cobertura dos testes executados. A avaliação humana em um AGI permite uma melhor adequação dos resultados do algoritmo a um determinado ambiente, o qual é um trabalho dificultoso de mapear uma função *fitness* tradicional, ao relacionar com o teste de agentes esta tarefa torna-se bastante difícil, considerando o fato dos agentes testados possuírem comportamentos diferentes(São flexíveis).

Em relação aos testes de *softwares* tradicionais, surgiu no inicío do desenvolvimento de *software*, a atividade de teste se resumia em navegar pelo código e corrigir problemas que já eram conhecidos. No início da década de 1970, a engenharia de *software* foi adotada como modelo para organizações e universidades, entretanto existia pouco consenso em relação ao que viria a ser teste. Myers em 1979, produziu um dos primeiros trabalhos mais completos e profundos. Nesse trabalho, foram definidos testes como um processo para encontrar erros e não apenas provar a boa funcionalidade de um *software* (BARTIÉ, 2002).

Segundo Rios (2006), o processo de teste é uma atividade complexa, onde sempre existirão itens ou atividades a serem aperfeiçoadas. Testar agentes é uma tarefa muito difícil, já que os mesmos possuem a característica de operarem de maneira independente. A otimização interativa busca o auxílio humano para que tarefas difíceis como testes de agentes, possam ser auxiliadas.

3 TRABALHOS RELACIONADOS

Esta seção apresenta uma revisão bibliográfica sobre as principais abordagens presentes na literatura para otimização interativa e testes de agentes racionais. É apresentado um resumo das abordagens descritas com suas principais características.

3.1 OTIMIZAÇÃO INTERATIVA

Em relação a otimização interativa é possível encontrar na literatura meios de incorporação, em algumas situações, que são desenvolvidas particularmente para algoritmo utilizado ou para o problema tratado. Pode-se deparar com as mais diversificadas maneiras de introduzir a subjetividade dentro do processo de busca.

Tendo como exemplo, algumas abordagens em que o usuário avalia algumas soluções geradas pelos algoritmos de busca (SIMONS; SMITH; WHITE, 2014). Em outras, podem ser incorporadas as preferências disponibilizando valores para parâmetros,dividindo os requisitos em categorias, selecionando requisitos ou avaliando um release completo Ferreira (2015), ou além do mais, um algoritmo que utiliza as predileções do usuário para desempatar elementos que são matematicamente iguais em que este utiliza a interatividade como uma maneira de diminuir a quantidade de comparações no processo de priorização de requisitos, agrupando as preferências do usuário quando há um empate entre requisitos (TONELLA; SUSI; PALMA, 2010).

Inoue *et al.* (1999) investiga o problema de alocação de enfermeiros, o mesmo sugere um agendamento automático, utilizando um algoritmo interativo evolutivo. O sistema aprende critérios de avaliação do usuário do sistema, e gera programações baseadas nos critérios obtidos, e exibe alguns dos melhores horários para o usuário. Em seguida, ela pode decidir quais partes do horário irá adotar, ou melhorar. O sistema reprograma as partes que precisam ser melhoradas. Repetindo este processo, o sistema e o usuário auxiliam para um resultado satisfatório

3.2 TESTE DE AGENTES

Nguyen *et al.* (2012), propõem uma metodologia que consiste em: representar os objetivos como função de qualidade (os objetivos que são necessários para avaliar o agente são transformados em uma função de qualidade para mensurar a satisfação dos *stakeholders*). O processo de geração de casos de teste é realizado a partir dos testes evolucionários, em que é gerado a população inicial, logo após, é realizado a execução do teste e o monitoramento, em

seguida, é realizada a coleta dos dados e a reprodução. Nesta abordagem não há uma simulação que possa monitorar e diagnosticar o comportamento do agente ao executar uma ação que envolve os objetivos ou metas.

Lam e Barber (2004), propõem um processo para compreender os comportamentos do agente de *software*. A abordagem imita o que um usuário humano testador faz na compreensão *software*, que são: construir e aperfeiçoar uma base de conhecimento sobre os comportamentos dos agentes, e usá-la para verificar e explicar os comportamentos dos agentes em tempo de execução. Com base na análise dos critérios para avaliação das abordagens, a abordagem não dispõe da utilização da noção de agentes racionais, não lida com a geração de casos de teste, não aborda o monitoramento e diagnóstico da ação do agente testado em nível de teste caixa branca.

Houhamdi (2011), apresenta uma abordagem de teste orientada a objetivos para programas de agentes. Na abordagem é especificado um processo de teste que complementa a metodologia Tropos, que é de Mylopoulos e Castro (2000) e intensifica a relação mútua entre a análise de objetivos e testes. Os autores propõem a derivação de casos de teste partindo da análise de artefatos de requisitos orientado aos objetivos do agente para a realização dois tipos de teste: unitário, para certificar de que todas as partes que fazem parte de um agente, tais como metas, base de conhecimento, agem conforme projetado, e de agente, que corresponde a testar a integração dos diferentes módulos dentro de um agente, pode-se perceber que a abordagem utiliza os testes de caixa branca e caixa preta.

Silveira (2013) propõe uma abordagem para o teste de agentes racionais que considera a noção de agentes racionais na utilização de casos de teste. Os casos de teste são gerados de acordo com a simulação das interações do agente testado com seu ambiente, para encontrar casos de teste e histórias correspondente em que o agente não obteve o resultado desejado. A abordagem considera um Projetista interagindo com programas de agentes envolvidos, ou seja, o programa agente a ser testado, que foi concebido pelo Projetista, o programa de ambiente de tarefa um programa de agente para a seleção de casos de teste. A abordagem formaliza um agente para monitorar e diagnosticar o teste de programa de agente.

Athamena e Houhamdi (2012) introduzem uma nova abordagem para testes de aceitação de software orientado por meta. No trabalho é proposto um processo de teste de aceitação que explora a ligação entre os requisitos e casos de teste. O processo especificado complementa o objetivo orientado a metodologia Tropos, que é de Mylopoulos e Castro (2000) e fortalece a relação mútua entre a análise de meta e testes.

Além disso é definido um processo de teste de aceitação estruturado e abrangente

para agentes de software de engenharia, fornecendo uma maneira sistemática de derivação de casos de teste a partir de análise de meta.

Durante o processo, são apontados os seguintes componentes: atores, os objetivos dos atores(representam objetivos e intenções dos usuários com relação ao sistema a que se destinam, de modo que o cumprimento destes objetivos é um ponto de referência fundamental para a aceitação do sistema), e as dependências entre atores. Estes atores incluem: (i) as partes interessadas e (ii) os atores do sistema. Os atores, são as partes interessadas em apresentar as suas intenções para os atores do sistema. As partes interessadas representam objetivos e requisitos em relação ao sistema. Silveira, Campos e CORTÉS (2015) apresentam uma abordagem que visa contribuir para o teste de programas agentes, incorporando um agente de monitoramento no processo de teste de agentes racionais. Este agente monitora o teste e realiza o diagnóstico de falhas presentes em um agente testado, identificando o agente de processamento de informações do subsistema que está causando as falhas.

3.3 TESTES COM OTIMIZAÇÃO INTERATIVA

Em relação à área de teste de software tradicionais Marculescu *et al.* (2015) propõem um *software* que faz uma diferenciação entre os interesses do domínio de aplicação e da engenharia de software, proporcionando aos especialistas de domínio interagir com o sistema a fim de selecionar os critérios de qualidade a serem operados. Na proposta é levado em consideração apenas as escolhas do especialista de domínio, já que cada domínio têm critérios únicos de qualidade e uma imensa variação na modelagem do software. A pesquisa tem como objetivo propor um sistema interativo para auxiliar a área de testes de *software*.

Junior, Filho e Araújo (2015) sugere uma ferramenta interativa para apoiar a construção de testes unitários de softwares, que consiste em analisar um arquivo, logo após, é gerado um grafo da complexidade ciclomática para cada método, permitindo ao desenvolvedor percorrer pelos caminhos independentes dos Grafos e verificar condições para que os caminhos sejam satisfeitos.

Nguyen *et al.* (2009) adota uma abordagem evolucionária para testes de agentes autônomos, a proposta consiste basicamente em representar os objetivos como uma função de qualidade (os objetivos que avaliam o agente são transformados em função de qualidade para estimar a satisfação dos *stakeholders*) e testes evolucionários (com a finalidade de conceber uma variedade de casos de teste com alto nível de dificuldade).

Os casos de teste são gerados a partir de quatro passos principais: geração da população inicial (é gerado de acordo com as especificações do testador ou randomicamente), execução e monitoramento (a execução do teste é realizada ao inserir o agente no ambiente de teste, já o mecanismo de monitoração é supervisiona o comportamento do agente), coleta dos dados observados e cálculo do valor de fitness e reprodução. Para cada agente testado é atribuído um período experimental em que são executados um número de testes com diferentes níveis de dificuldade.

4 ABORDAGEM PARA SELEÇÃO INTERATIVA DE CASOS DE TESTE

A abordagem proposta para o teste de agentes racionais está fundamentada, principalmente, na noção de agentes racionais integrada com a otimização interativa para a geração e seleção dos casos de teste gerados. Mais especificamente, a abordagem consiste em uma extensão a ferramenta para o teste de agentes racionais inicialmente desenvolvido em (SILVEIRA, 2013). Esta extensão consiste em uma inserção de novos componentes a ferramenta, capacitando-o a realização de novas formas de teste.

O primeiro componente inserido na ferramenta consiste em um gerador de casos de teste (GERADOR) capaz de gerar objetivamente, conforme o desejo do projetista do agente em teste (Agent), representações em grafos de ambientes de tarefas para medir o desempenho de Agent, interagindo com um programa ambiente (Amb) que incorpora estas representações.

O segundo componente inserido consiste em um agente racional capaz de monitorar (MOA) os episódios de uma história do agente Agent no ambiente Amb, e descobrir falhas e indicar qual(is) módulo(s) de processamento de informação de Agent está(ão) causando a falha em Amb.

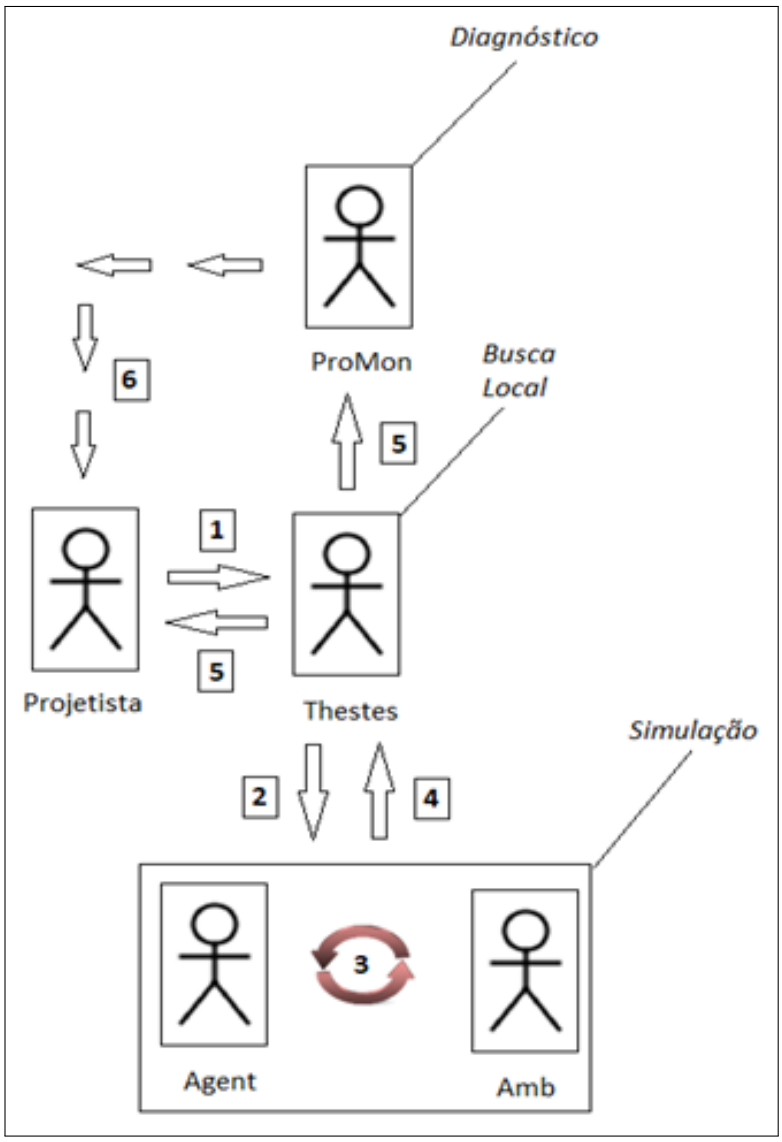
4.1 FERRAMENTA PROPOSTO POR SILVEIRA (2013)

A Figura 8 ilustra os principais componentes presentes na ferramenta para o teste de agentes racionais, desenvolvido inicialmente por Silveira (2013), e as etapas envolvidas no processo de busca local para a seleção de conjuntos de casos de teste, que maximizem uma função de utilidade, ou seja, que considera um vetor composto pelo negativo dos valores retornados por n funções objetivos (funções escalares), isto é, representando o negativo da medida de avaliação de desempenho.

Na versão da ferramenta desenvolvida por Silveira (2013), é possível identificar cinco agentes interagindo, um humano e outros quatro agentes artificiais, ou seja:

- **Projetista:** pessoa(s) envolvida(s) na concepção de um agente artificial racional, empregando como referencial teórico a arquitetura abstrata de agentes inteligentes descrita em (WOOLDRIDGE, 2002).
- **Agent:** programa do agente artificial que o Projetista deseja testar.
- **Amb:** programa que implementa o comportamento do ambiente de tarefas, empregado no processo de simulação das interações com Agent, obedecendo a um protocolo de comunicação.

Figura 8 – Abordagem de Silveira (2013).



Fonte – Adaptado de (SILVEIRA, 2013).

- **Thestes:** agente de resolução de problemas de seleção de conjuntos de casos de teste. Considera o problema de seleção como um problema de otimização e implementa esquemas de busca local no espaço de estados do problema, orientado por uma função de utilidade, visando encontrar casos em que o agente Agent é mal avaliado em Amb.
- **ProMon:** programa que recebe de Thestes os conjuntos de casos de teste selecionados como solução do problema de otimização, e envia para o projetista informações a respeito dos módulos de processamento de informação (ver, próximo e ação) que estão causando as falhas de Agent em Amb.

Além das informações a respeito do programa em teste Agent e da medida de avaliação de desempenho a ser aplicada neste programa, no início do processo de resolução do

problema de otimização (Etapa 1), o Projetista envia informações associadas ao processo de busca local implementado por Thestes e ao processo de simulação das interações entre Agent e Amb como, por exemplo, o número de interações entre eles e outras informações sobre os valores de parâmetros associados ao ambiente de tarefas, que Thestes deve instanciar no programa Amb (Etapas 2, 3 e 4).

Ao final do processo de busca, o agente Thestes envia para o agente ProMon e para o Projetista um conjunto de casos de teste, o conjunto de histórias do agente Agent no ambiente Amb, associadas ao conjunto de casos, e os valores de desempenho associados aos episódios componentes de cada história, e o desempenho médio de Agent em Amb associado com a história (Etapa 5). Finalmente, fechando o ciclo, o agente ProMon foi projetado para analisar cada um dos episódios em uma história, visando diagnosticar aqueles módulos de processamento da informação que estão causando as falhas de Agent em Amb, e enviar (assim como Thestes) estas informações para o Projetista do agente em teste (Etapa 6).

Assim, na versão concebida por Silveira (2013), o Projetista interage com a ferramenta no início do processo de busca de um conjunto de casos de teste, por meio do agente Thestes, e no final do processo também por meio do agente Thestes. Apesar de concebido o agente ProMon não entrou em desenvolvimento na primeira versão da ferramenta, mas seu quadro formal foi utilizado para validar alguns dos principais resultados obtidos por Thestes.

Além do mais, Thestes implementa um teste do tipo caixa-preta, visto que ele não é capaz de identificar quais módulos de Agent causam baixo desempenho no ambiente de tarefas. Por outro lado, apesar de implementar um teste do tipo caixa-branca, no esquema de interação na primeira versão da ferramenta, o agente ProMon não é capaz de interagir com o agente Thestes durante o processo de busca, nem com o Projetista independentemente do agente Thestes.

4.2 FERRAMENTA ESTENDIDA

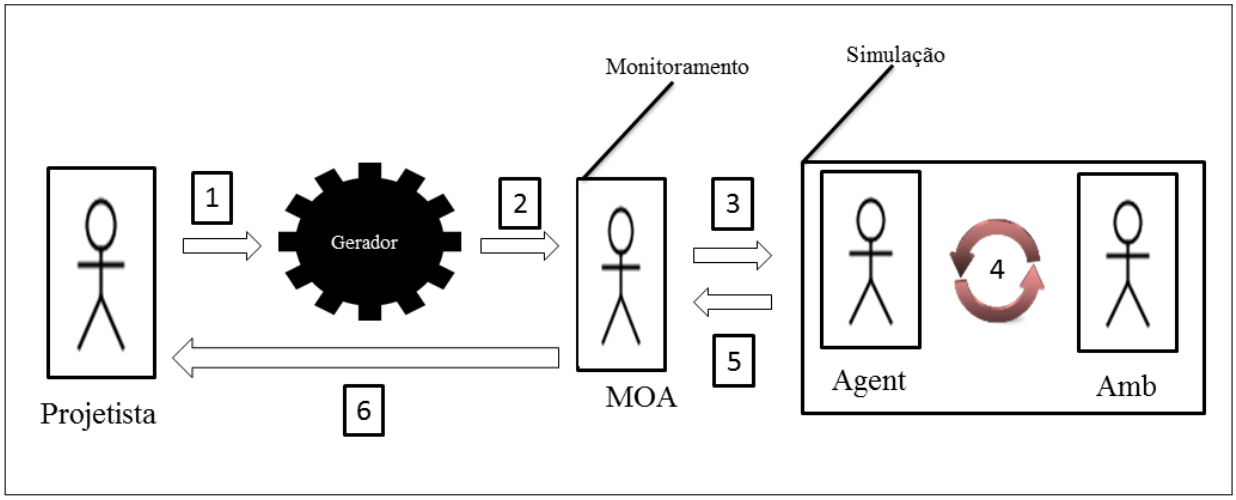
Esta nova versão da ferramenta estende o *framework* proposto por Silveira (2013) através da inserção das noções de padrão ambiental e de dois novos agentes, visando produzir novas formas de interação com o Projetista no início do processo de busca para resolver o problema de seleção de casos de teste, e com o agente Thestes e com o Projetista durante o processo de busca, ou seja:

- **GERADOR:** artefato que pode ser utilizado pelo agente Projetista para gerar "objetivamente" casos de testes visando avaliar o desempenho de Agent em Amb.

- **MOA:** agente responsável por receber um caso de teste gerado pelo Projetista e/ou por GERADOR, simular e monitorar as interações de Agent em Amb, visando descobrir os módulos de processamento da informação de Agent que estão causando falhas em Amb, e computar um valor de desempenho que leva em consideração os módulos descobertos (teste caixa-branca), diferentemente do agente Thestes que emprega uma função utilidade para representar a medida de avaliação de desempenho estabelecida pelo Projetista.

Considerando estes dois componentes, o Projetista pode interagir com a ferramenta estendida em, pelo menos, duas maneiras distintas conforme o tempo e a energia física disponíveis para esta interação. A Figura 9 ilustra a primeira situação, aqui denominada **Situação 1**, ou seja, em que o Projetista utiliza o gerador de casos de teste GERADOR e interage com o agente de monitoramento MOA, visando descobrir os módulos de processamento de informação causando falha em Agent, sem precisar interagir com os agentes Thestes e ProMon, conforme acontece no *framework* original, ilustrado na Figura 9.

Figura 9 – Modo de Interação do projetista com a ferramenta estendida - Situação 1.



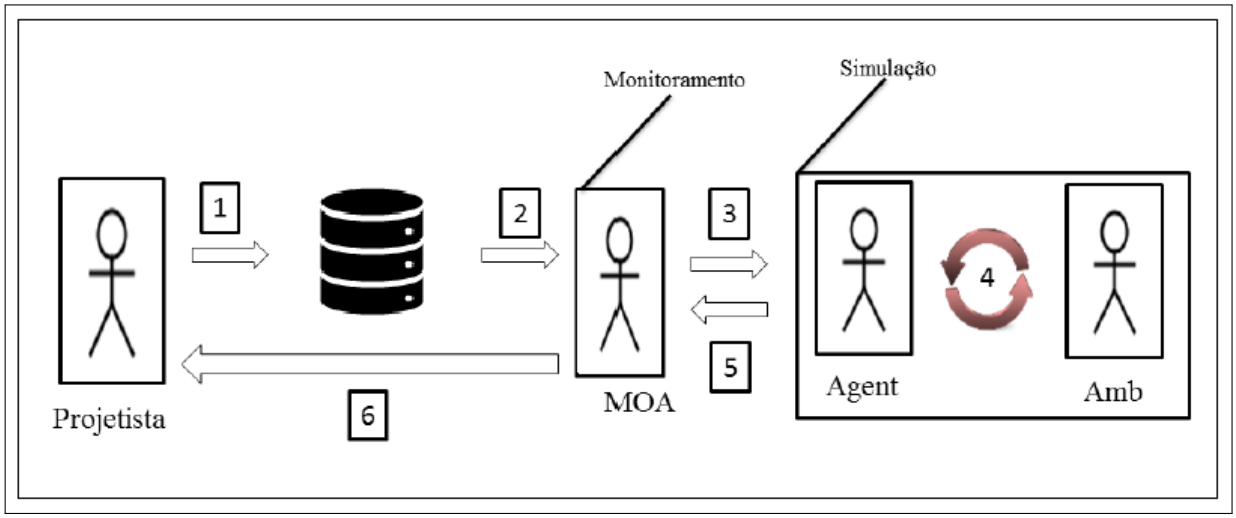
Fonte – Adaptado de (SILVEIRA, 2013).

Neste modo de utilização da ferramenta, o Projetista utiliza o GERADOR para gerar casos de teste de maneira objetiva, ou seja, cada caso gerado deve obedecer a um "padrão ambiental"especificado pelo Projetista, isto é, um padrão definido de acordo com os valores desejados de determinados atributos associados ao ambiente de tarefas representado no caso de teste (Etapa 1).

Considerando cada caso gerado pelo Projetista (Etapa 2), o agente MOA simula as interações de Agent em Amb (Etapas 3 e 4), instanciado com o ambiente de tarefas representado no caso, e retorna para o Projetista uma medida de avaliação de desempenho e informações a

respeito dos módulos de processamento de informação de Agent que estão causando falhas em Amb (Etapas 5 e 6).

Figura 10 – Modo de interação do Projetista com a ferramenta estendida - Situação 2.



Fonte – Adaptado de (SILVEIRA, 2013).

A Figura 10 ilustra o segundo modo de interação do Projetista com a ferramenta estendida, aqui denominada **Situação 2**. Nesta situação, o Projetista utiliza casos de testes em uma base de casos, previamente avaliados pelo agente MOA e armazenados pelo projetista, por considerar casos relevantes para o teste de Agent. Da mesma forma que na situação anterior, o agente MOA simula as interações entre Agent e Amb, e envia as informações resultantes para o Projetista (Etapa 6).

Nas duas situações, o Projetista interage com o agente de monitoramento MOA, interrompendo os testes para interagir com o GERADOR alterando o ambiente que ainda não foi alterado por Agent, visando perceber determinados comportamentos de Agent, a fim de descobrir os módulos de processamento de informação que estão causando falha em Agent. Assim, como no caso de Thestes no *framework* original, o Projetista recebe do agente MOA o valor da medida de avaliação de desempenho do agente Agent e, diferentemente do *framework* original, recebe também as informações sobre os módulos de processamento de Agent que estão causando falhas no ambiente Amb.

4.3 GERADOR DE CASOS DE TESTE

4.3.1 Casos de Teste em Ambientes de Tarefas de Agentes

Os conjuntos de casos de testes gerados são usados pelo Projetista para avaliar o agente Agent em desenvolvimento para decidir eventualmente se o agente está pronto para ser implantado ou se precisa de melhorias adicionais. O requisito básico para a aceitação de Agent implica que todos os objetivos na medida de avaliação de desempenho, estabelecida pelas partes interessadas, sejam alcançados. Esta seção exemplifica alguns destes casos de teste conhecidos da literatura sobre agentes.

Por exemplo, conforme adotado em Silveira (2013) no processo de validação do agente Thestes, vale considerar o primeiro exemplo um sistema composto por vários agentes aspiradores de pó projetados para trabalhar em uma área pública como, por exemplo, um aeroporto conforme adotado na abordagem de (NGUYEN *et al.*, 2009). Os agentes são responsáveis por manterem aeroporto limpo. A Figura 11 ilustra este ambiente de tarefas.

Figura 11 – Um cenário especial para testar o agente de limpeza.

Fonte – (NGUYEN *et al.*, 2009).

Na Figura 11 são fixadas as localizações de duas estações de carga de energia, dois

caixotes de lixo e seis obstáculos. Os obstáculos são colocados nos cantos, três deles no canto superior direito. A Tabela 1 descreve um conjunto de três casos de teste que podem ser gerados neste ambiente de tarefas, e algumas medidas de avaliação de desempenho adequadas para avaliar as interações dos agentes aspiradores de pó com os respectivos casos, ou seja, ambientes instanciados.

Tabela 1 – Conjunto de casos de teste que podem ser gerados no ambiente.

Caso de Teste	Ambiente de Tarefa	Avaliação de Desempenho
CT1	Na área real de um aeroporto A em que os aspiradores de pó robô devem limpar, os lixos são colocados em posições especificadas (p1;...; pn) e a quantidade de lixo em cada posição é (a1; a2;...; an).	A área deve ser limpa em pelo menos t minutos.
CT2	Os usuários da área A jogam lixo repetidamente de maneira aleatória.	A área A deve ser limpa periodicamente.
CT3	Dependendo da hora no aeroporto, a área A pode estar mais ou menos suja, isto é, a quantidade de lixo é função do tempo e da posição.	Aspiradores devem adaptar os intervalos de tempo para limpeza e movimentação para posições relevantes.

Fonte – os autores

Neste tipo de ambiente de tarefas, uma configuração ambiental descrita em um caso de teste deve considerar a quantidade e localização de obstáculos, caixas de lixo, resíduos e estações de carga. Cada um desses fatores deve ser codificado adequadamente no agente Amb como, por exemplo: (a) divida a área A em uma grade formada por NxN células, em que N denota a resolução da grade; (b) coloque objetos, ou seja, obstáculos, resíduos, caixas de lixo estações de carregamento nas células, tal que uma célula que contém um objeto é indicada por 1, enquanto uma célula sem conteúdo é indicada por 0, conforme exemplo ilustrado na Figura 12.

As resoluções dos fatores ambientais podem ser diferentes e sua quantidade pode ser controlada durante o processo de teste. Por exemplo, podemos escolher o número de caixas de lixo e estações de carga menores do que na realidade, enquanto a quantidade de lixo e o número de obstáculos podem ser escolhidos como muito maiores. Diferentes resoluções dos fatores ambientais dificultam a geração de um número suficiente de casos de teste adequados.

Por exemplo, a Figura 11 apresenta um ambiente de tarefas de agente semelhante ao aeroporto dos aspiradores de pó, e em que as diferentes resoluções dos fatores ambientais dificultam a geração de um número suficiente de casos de teste adequados.

De maneira semelhante ao ambiente de tarefas Aeroporto, ilustrado anteriormente, aA

Figura 12 – Codificação de um caso de teste.

1	0	1	0	1	0
0	1	1	0	1	0
0	1	1	0	1	0
1	1	1	0	1	1
0	1	0	1	0	0
1	0	1	0	0	0

Fonte – Próprio autor.

Figura 13 – Cenário "Agents on Mars".

Fonte – (FRANCO, 2014)

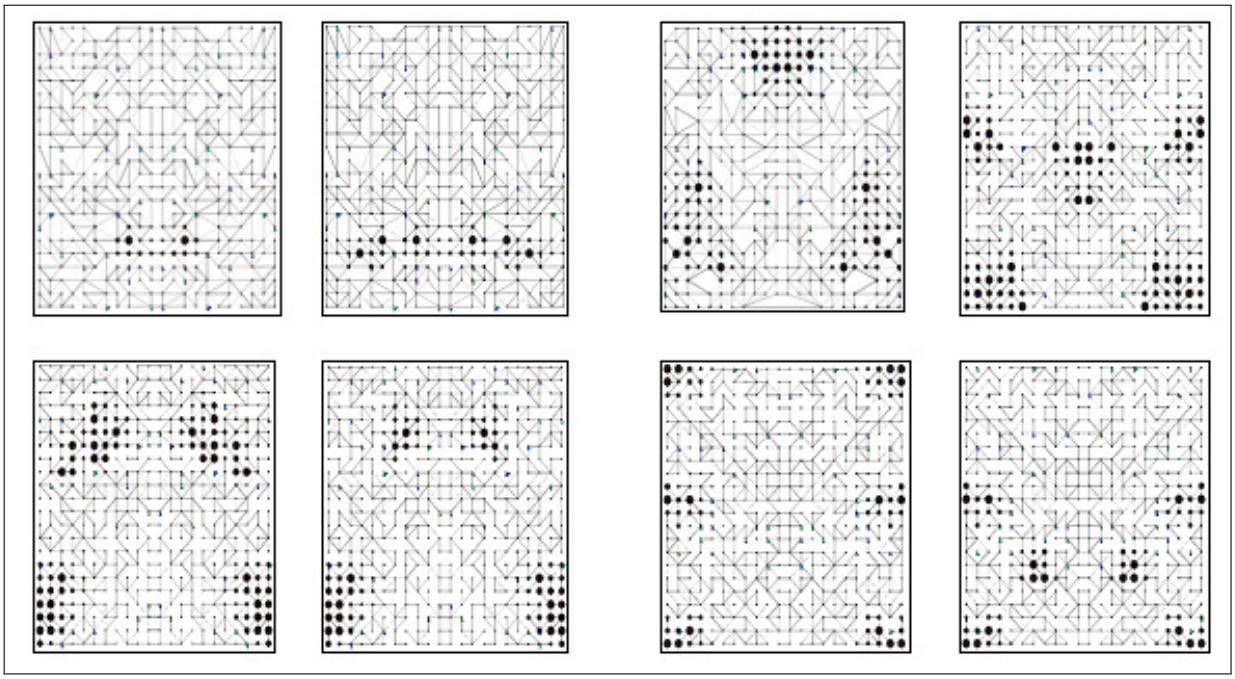
Figura 13 ilustra o cenário Agents on Mars, uma plataforma de teste fornecida por meio de uma competição entre agentes denominada MAPC (*Multi-Agent Programming Contest*) (FRANCO,

2014). O ambiente de tarefas no MAPC é um campo de batalha na Lua, onde dois times de veículos autônomos, com sensores e atuadores especiais, devem explorar e conquistar áreas ricas em recursos naturais, poços contendo água que estão localizados em lugares desconhecidos.

Conforme é possível perceber nas Figuras 11 e 12, a o ambiente de tarefas dos agentes no cenário "Agents on Mars"pode ser representado por um grafo valorado, em que, neste caso, os vértices denotam poços e possíveis localizações para os agentes, e as arestas indicam a possibilidade de passar de um vértice para outro com um custo de energia para o agente. Cada vértice no grafo tem um valor de recurso. Neste grafo, uma zona é um subgrafo coberto por um time. Um time recebe uma recompensa em dinheiro sempre que atinge um marco importante.

Semelhantemente ao exemplo anterior, as resoluções dos fatores ambientais podem ser diferentes e sua quantidade deve ser controlada durante o processo de teste. A Figura 14 ilustra uma diversidade de casos de teste (mapas) que podem aparecer no ***testbed***, por exemplo, com diferentes valores no número e tamanho de áreas valiosas e as distâncias entre elas.

Figura 14 – Casos de teste com diferentes resoluções dos fatores ambientais.



Fonte – (FRANCO, 2014)

Visando lidar com essa diversidade de ambientes de tarefas de agentes, ou seja, quantas "áreas valiosas", o tamanho e como elas estão distribuídas, este trabalho propõe utilizar as ideias sobre padrões ambientais desenvolvidas em (FRANCO; CAMPOS; SICHMAN, 2016). A próxima seção descreve de maneira informal os principais resultados teóricos sobre o assunto adaptados na proposta do gerador de casos de teste GERADOR.

4.3.2 Casos de Teste como Grafos Valorados, a Noção de *Cluster* e de Padrão Ambiental

Cada caso de teste gerado por GERADOR pode ser representado por um grafo valorado, em que os vértices denotam recursos e possíveis "locais"para os agentes, e cada aresta indica a possibilidade do agente Agent mudar de estado, passando de um vértice para outro com um custo para o agente. Formalmente, um caso de teste CT = (V, E) consiste em um conjunto de vértices V e um conjunto de arestas E, que contém pares de elementos distintos de V, uma função de valor de vértice fvv: CT(V) $\longrightarrow$ R e uma função de valor da aresta fve: CT(E) $\longrightarrow$ R, em que R é o conjunto de números reais empregados para avaliar os recursos nos vértices em G e representar o custo da passagem da aresta.

A noção de padrões ambientais em um CT (Caso de Teste) tenta capturar a diversidade de ambientes, ou seja, quantas "áreas valiosas", subgrafos representando os recursos de alto valor, aparecem no grafo G. No trabalho desenvolvido por Franco (2014), essas "áreas valiosas"foram denominadas clusters. Formalmente, um cluster no grafo representativo de um caso de teste CT = (V, E) é qualquer subgrafo C = (V ', E') que satisfaça três condições:

- (1) o valor do recurso associado a cada vértice em um *cluster* C é maior que um valor limite inferior da capacidade do recurso por vértice;
- (2) C é um subgrafo conectado, com apenas um vértice (isolado) com alto valor ou um subgrafo no qual todos os vértices em C não são isolados;
- (3) a soma dos valores dos recursos associados a todos os vértices em um *cluster* C é maior que o valor limite inferior da capacidade dos recursos por *cluster*, que deve ser menor ou igual à capacidade do grafo, ou seja, a soma de todos os valores dos recursos no grafo CT.

Ao considerar a distribuição dos valores dos vértices de um *cluster* C em CT, são possíveis dois casos extremos: *clusters* altamente diversos com o número máximo possível de valores distintos e *clusters* homogêneos com vértices do mesmo valor. A noção de padrão ambiental (EP) associada a um caso de teste CT é definida como uma tupla EP(CT) = <S, N, H, D>, considerando quatro atributos associados à distribuição de todos os *clusters* em CT, de modo que: S representa o tamanho médio dos *clusters* em CT; N representa o número de *clusters* em CT; H (homogeneidade) representa quanto os *clusters* em G têm aproximadamente o mesmo valor; e D (dispersão) representa a distância média aproximada entre os *clusters* em CT.

Por exemplo, os casos de teste na Figura 14 ilustram a noção de *cluster* e oito padrões ambientais diferentes, ou seja, os oito casos extremos associados aos quatro atributos definindo cada EP(CT). Considerando as condições que cada *cluster* deve satisfazer, no exemplo o valor de

limite inferior para cada vértice em um *cluster* é igual a 1 e o valor de limite inferior do *cluster* é 10. Os vértices maiores na cor preta identificam os vértices e arestas que satisfazem as três condições para ser um *cluster*.

As duas primeiras figuras ilustram tamanhos diferentes para os *clusters*. A terceira e quarta figuras ilustram diferentes números de clusters, três *clusters* na terceira e cinco na quarta. A quinta e sexta figuras têm diferentes valores de homogeneidade, os vértices nos *clusters* na quinta figura têm aproximadamente o mesmo valor, e os vértices na sexta figura não. Finalmente, diferentes valores de dispersão são ilustrados na sétima e oitava figura, ou seja, onde os seis *clusters* na sétima figura estão mais dispersos no grafo e os seis *clusters* na oitava figura estão mais próximos uns dos outros.

4.3.3 Geração de Casos de Teste

O programa GERADOR consiste em um artefato de *software* para ajudar ao Projetista do agente Agent na geração de casos de teste específicos de seu interesse, que seriam difíceis de serem gerados durante o processo de busca realizado pelo agente Thestes. Mas, além da interferência do Projetista, o programa também é capaz de propor casos previamente testados e armazenados, considerando desempenhos satisfatórios ou insatisfatórios requisitados pelo Projetista, e casos que tiveram frequência alta de utilização anteriormente.

De acordo com o que foi descrito na seção anterior, pode-se dizer que o programa GERADOR considera duas regras para a geração e reconhecimento do que vem a ser um caso de teste CT= (V, E) para o agente Agent, ou seja:

- (1) pares ordenados da forma (v, val_v) representam casos de teste para Agent,
- (2) casos de teste ligados por pares da forma (e, val_e) também representa casos de teste para Agent,

tal que cada vértice v $\in$ V denota o recurso em um local para ser visitado por Agent, associado a um valor de função de valor de vértice val_v = fvv(v) $\in$ R; e cada aresta e $\in$ E indica a possibilidade do agente Agent passar de um vértice para outro, associado a um valor de função de valor de aresta val_e = fve(e) $\in$ R.

Considerando este formalismo para a geração de casos de teste, três possibilidades do Projetista interagir com o GERADOR para a geração de um caso de teste CT para Agent, ou seja:

- (1) o Projetista poderá especificar manualmente um CT definindo rigorosamente o padrão

ambiental EP(CT), e o programa GERADOR poderá reconhecer se CT é um caso de teste para Agent e, caso afirmativo, realizar a geração do mesmo para ser enviado ao agente MOA e executado em simulação;

- (2) o Projetista poderá especificar de maneira *fuzzy* o o CT, definindo o informalmente o padrão ambiental EP(CT), e o programa GERADOR se incumbirá de realizar a inferência *fuzzy* e gerar um CT adequado para teste de Agent para ser executado em simulação;
- (3) o Projetista poderá requisitar ao GERADOR para executar simulação com um caso CT gerado previamente, em que outros agentes testados foram bem ou mal avaliados;
- (4) o Projetista poderá requisitar ao GERADOR para executar simulação com um caso CT gerado previamente e que teve uma frequência alta de utilização anteriormente.

Nos casos de interação (2)-(4), caso deseje, o Projetista pode indicar a quantidade de casos de teste a serem gerados pelo GERADOR para execução em simulações. A próxima seção descreve o componente *fuzzy* do programa GERADOR, ou seja, que permite ao Projetista especificar o padrão ambiental EP(CT) empregando valores linguísticos para os atributos ambientais para a definição de *clusters* em CT: S (tamanho médio dos *clusters*), N (número de *clusters*), H (homogeneidade dos *clusters*), e D (dispersão dos *clusters*).

4.4 *MONITORING AGENT*

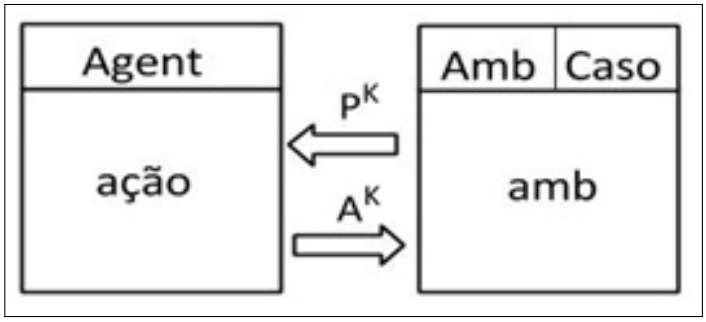
O pressuposto na abordagem implícita no agente (MOA) é que o projetista de um agente racional Agent deve considerar uma medida de avaliação de desempenho para o mesmo, as propriedades do ambiente de tarefas Amb de Agent, as estruturas de agentes disponíveis e conceber o programa Agent visando maximizar seu desempenho nas histórias de Agent em Amb.

4.4.1 Estruturas de Programas de Agentes Artificiais e Medidas de Desempenho

A Figura 15, ilustra um episódio na história das interações entre um agente artificial que está sendo testado e seu ambiente de tarefas. Na figura, Agent representa o programa do agente artificial em teste. Nesta dissertação, pode ser implementado considerando um dos quatro tipos de programas de agentes, ou seja: reativo simples, baseado em modelos, baseado em objetivo e utilidade. Por sua vez, Amb representa um programa ambiente implementado capaz de interagir com o agente Agent por meio de algum protocolo de interação considerando algum caso de testes CT, gerado pelo programa gerador.

Mais especificamente, com relação a figura seja:

Figura 15 – Episódio na história de Agent em Amb.



Fonte – Própria autora.

- **P** = {P1, P2, ...} - representa os estados percebidos do ambiente de Agent – em qualquer instante K o ambiente está em um dos estados do conjunto de estados do ambiente;
- **A** = {A1, A2, ...} – representa a capacidade efetuadora de Agent – um conjunto de ações possíveis de serem executadas pelos atuadores do agente;
- ação : $P^* \longrightarrow$ **A** – representa o comportamento de Agent – uma função que mapeia sequências de estados do ambiente **P** $\in P^*$ em ações A $\in$ **A**, ou seja, intuitivamente, Agent decide que ação executar baseado em sua história, sua experiência até o momento da decisão;
- amb : **P**x**A** $\longrightarrow$ **P** – comportamento (determinístico) de Amb – uma função que mapeia o estado corrente do ambiente P $\in$ **P** e uma ação A $\in$ **A**, em um estado amb(P, A) resultante da execução de A em P;
- Ω: um conjunto de descrições de ambientes de tarefa factíveis de instanciar em Amb e testar Agent;
- Caso de teste CT $\in \Omega$: uma descrição específica de ambiente no conjunto Ω;
- h(CT) $\in$ (PxA)^{NInt}: história de comprimento N_{Int} de Agent em Amb correspondente a um caso de teste CT – uma sequência da forma:

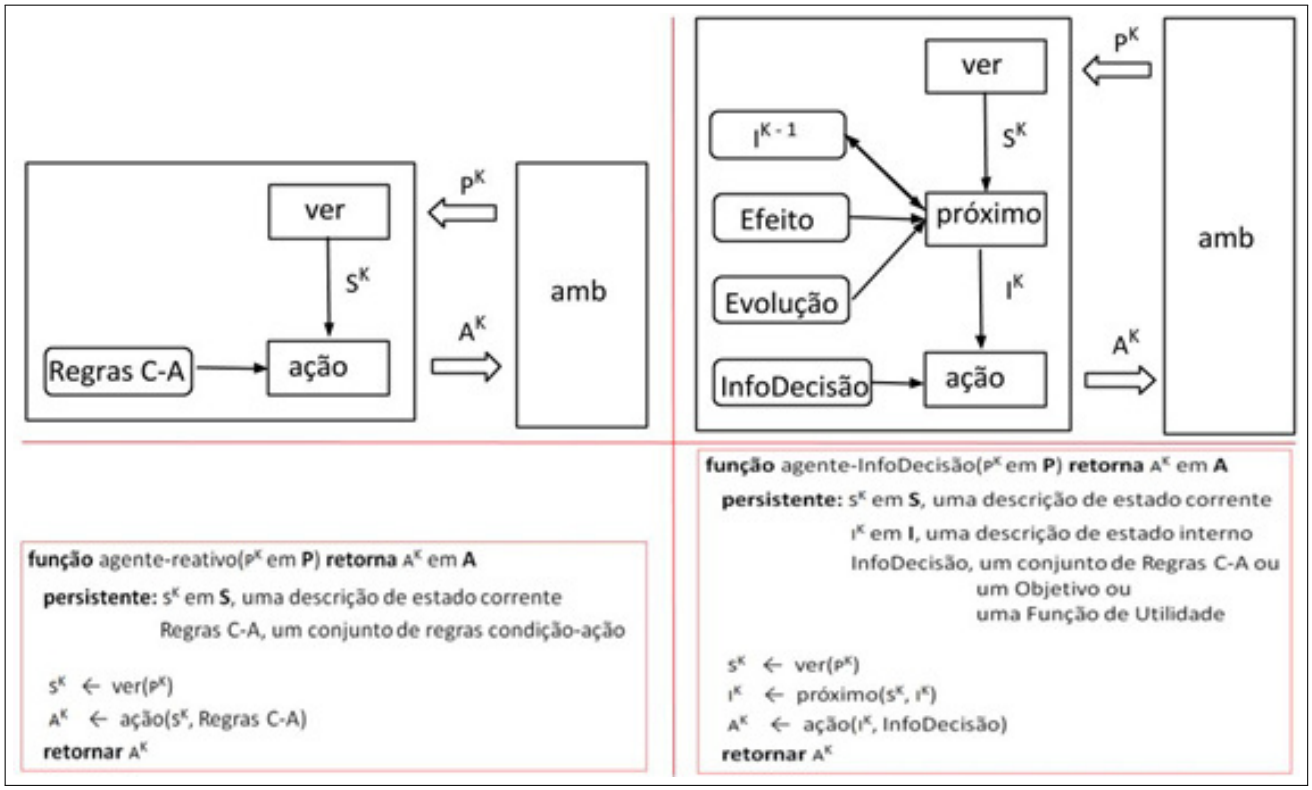
$$(P1, A1) \longrightarrow (P2, A2) \longrightarrow (P3, A3) \longrightarrow \ldots \longrightarrow (P_{Nint}, A_{Nint}),$$

  tal que P1 representa o estado inicial do ambiente, P_{Nint} o último estado do ambiente e A_{Nint} representa a última ação que o agente escolheu para executar em P_{Nint};
- Ep^K(h(CT)) $\in$ PxA: episódio na interação $\underline{K}$, K $\leq N_{Int}$, da história de Agent em Amb correspondente ao caso de teste CT $\in \Omega$;

A Figura 16 ilustra graficamente os esqueletos dos quatro tipos de agentes que podem ser considerados na implementação de Agent. A figura foi gerada a partir de uma síntese das ideias descritas por Russell&Norvig sobre estruturas de programas de agentes (RUSSELL;

NORVIG, 2004) , e de Wooldridge sobre arquiteturas abstratas de agentes (WOOLDRIDGE, 2002). Consiste em um refinamento em duas etapas do subsistema de tomada de decisão de Agent descrito na Figura 1, na seção que descreve o quadro formal associado ao modelo F da organização. À esquerda, o primeiro refinamento dando origem ao subsistema de percepção, ver, de Agent. À direita, o segundo refinamento dando origem ao subsistema de estado interno de Agent, próximo.

Figura 16 – Esqueletos de programas possíveis para Agent.



Fonte – Própria autora.

Mais especificamente, o primeiro refinamento apresenta a arquitetura e o esqueleto de programas de agentes reativos simples baseados em regras condição-ação. O segundo refinamento apresenta a arquitetura e o esqueleto de programas de agentes reativos baseados em modelos, orientado por objetivos ou orientados por metas, dependendo do tipo de informação (InfoDecisão) que o projetista desejar incorporar no processo de tomada de decisão (regras, objetivo ou utilidade). Assim, seja:

- P = {P1, P2, ...} – um conjunto de descrições de estado do ambiente de tarefas de Agent, conforme descrito em um caso de teste Caso;
- ver: P $\longrightarrow$ S – subsistema de percepção de Agent – captura a habilidade do agente observar seu ambiente – função que mapeia percepções de estados do ambiente para um conjunto

de descrições de estados do ambiente;

- I = {I1, I2, ...} – estados internos do ambiente – em qualquer instante Agent mantém em memória uma descrição de estado do ambiente;
- próximo: I x S $\longrightarrow$ I – representa o subsistema de atualização de estado interno de Agent – função que mapeia um estado interno e a descrição de estado atual, que chega da função ver, em um estado interno;
- ação: I $\longrightarrow$ A – representa o subsistema de tomada de decisão de Agent – função que mapeia estados internos em ações.

Em resumo, à esquerda na Figura 15, o programa Agent reativo simples seleciona suas ações baseando-se apenas em sua percepção atual e em um conjunto de regras no formato condição-ação. À direita, o programa Agent reativo baseado em modelos mantém internamente em memória uma descrição de estado de seu ambiente, o qual depende do histórico de suas percepções. O programa Agent orientado por objetivos, além das informações em memória, mantém informação sobre os objetivos que descrevem situações desejadas no ambiente. O programa Agent orientado por utilidade possui uma função utilidade que mapeia uma descrição de estado do ambiente em um grau de felicidade associado.

O subsistema de atualização de estado interno dos programas Agent nos tipos reativos baseados em modelos, orientados por objetivos e por utilidade, e o subsistema de tomada de decisão dos dois últimos tipos, em comum, processam informações de dois tipos, ou seja: sobre como ambiente evolui independentemente das ações do agente (Evolução) e sobre o efeito das ações do agente no ambiente (Efeito).

A Tabela 2 exemplifica uma medida de avaliação de desempenho para o caso em que Agent é um programa agente aspirador de pó que deve limpar um ambiente e maximizar os níveis de limpeza do ambiente e de energia em sua bateria ao final da tarefa. A primeira coluna descreve uma parte das informações nas percepções dos estados do ambiente de Agent (P $\in$ **P**) enviado por Amb. A segunda coluna descreve as ações possíveis nestes estados (A $\in$ **A**). Na terceira e quarta colunas respectivamente, associadas aos objetivos de energia e de limpeza, duas funções escalares (av_E e av_L) para medir o desempenho do agente em cada episódio de sua história no ambiente.

A Figura 17 ilustra seis episódios fictícios de uma história das interações de Agent com Amb, configurado em um caso de teste, e as medidas de desempenho de Agent em termos de energia **($\mathbf{av}_E(\mathbf{Ep}^k)$)** e limpeza **($\mathbf{av}_L(\mathbf{Ep}^k)$)** em cada episódio k. Conforme pode ser percebido nas descrições de estado do ambiente de tarefas (S), Agent percebe apenas localmente o ambiente,

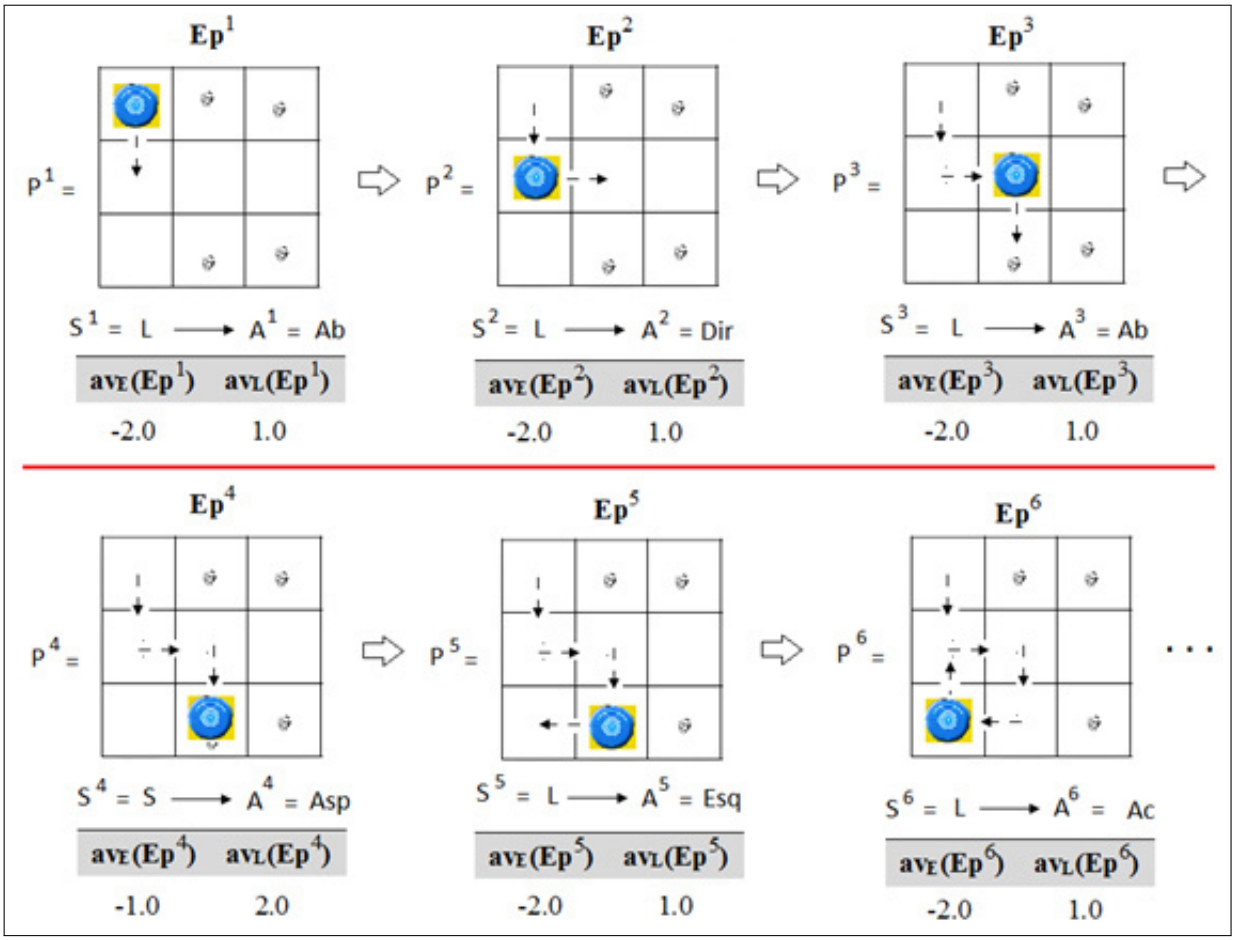
Tabela 2 – Exemplo de Medida de Avaliação de Desempenho.

$\mathbf{P}^k$	$\mathbf{A}^k$	$\mathbf{av}_E(\mathbf{Ep}^k)$	$\mathbf{av}_L(\mathbf{Ep}^k)$
..., Lim, ...	Ope	-1.0	-2.0
..., Lim, ...	Dir, Esq, Aci, Aba	-2.0	1.0
..., Lim, ...	N-ope	0.0	0.0
..., Suj, ...	Ope	-1.0	1.0
...	...	...	...

Fonte – Elaborado pela autora.

isto é, se o local em que está contém sujeira (S^K = Sujo), ou não (S^K = Limpo), não sendo capaz de perceber os locais vizinhos ao seu local no ambiente de tarefas.

Figura 17 – História de Agent em Amb com Seis Episódios.



Fonte – Própria autora.

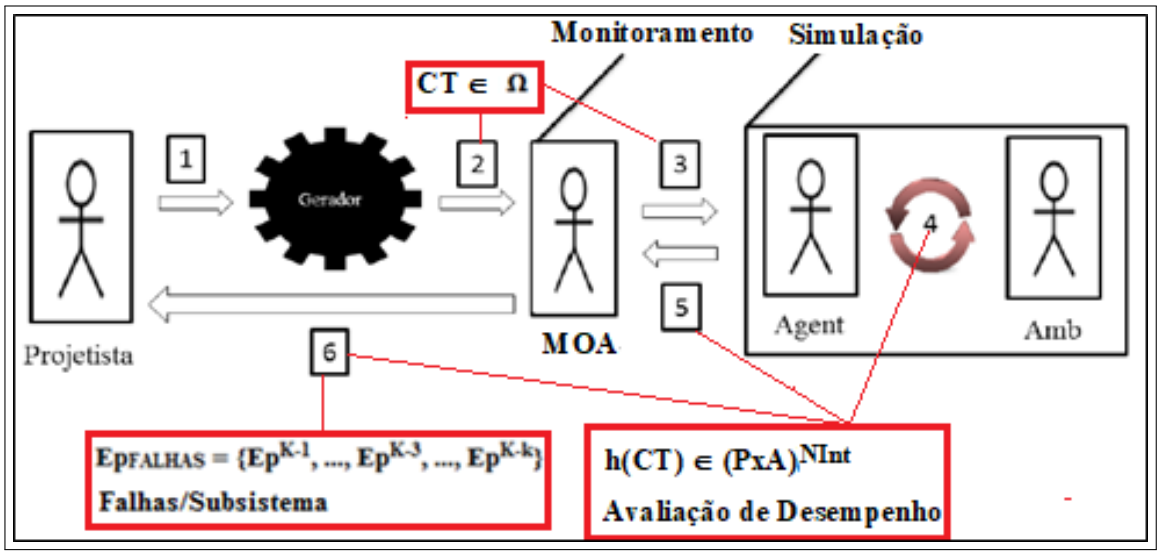
Conforme mencionado na introdução desta seção o agente MOA conhece a medida de avaliação de desempenho estabelecida pelo Projetista, assim como as estruturas de agentes disponíveis, ou seja, o formalismo representando cada um dos módulos de processamento de informação destes agentes. A próxima subseção apresenta os modos como o agente MOA utiliza

este conhecimento para identificar falhas nas histórias das interações dos agentes Agent em Amb.

4.4.2 Função do Programa do Agente MOA

A concepção do agente MOA considera que o programa Agent pode ser idealizado considerando um dos quatro tipos básicos de programas de agentes ilustrados na Figura 16. Quando da escolha de um tipo, as propriedades internas do programa Agent devem ser adequadas às propriedades de seu ambiente de tarefa, concretizadas em um caso de teste CT em Amb. A Figura 18 ilustra um dos modos de interação do Projetista com o GERADOR, e deste com o agente MOA, e as informações, geradas como *feedback* para o Projetista.

Figura 18 – Etapas 3-5 da Figura 16, para o diagnóstico do Agente MOA.



Fonte – Própria autora.

A Figura 18 considera que o agente MOA recebe do gerador de casos de teste GERADOR um caso de teste CT $\in \Omega$. Em seguida, a Etapa 3 o agente MOA incorpora o caso de testes CT no agente Amb e inicia, na Etapa 4, um processo de simulação envolvendo um número de interações do agente Agent com Amb. Mais especificamente, são geradas as seguintes informações para o Projetista:

- Uma história de comprimento N_{Int} de Agent em Amb correspondente ao caso de teste CT, ou seja, h(CT) $\in$ $(PxA)^{NInt}$;
- A avaliação de desempenho de Agent em cada episódio e o valor acumulado ao longo da história;
- O conjunto de episódios na história que são considerados como falhas de Agent, Ep_{FALHAS} = $\{Ep^{K-1}, ..., Ep^{K-3}, ..., Ep^{K-k}\}$; e

- Os subsistemas de processamento de informação em Agent que possivelmente causaram estas falhas.

De maneira a gerar estas informações, o agente MOA foi concebido com a capacidade de tomar decisões nos mesmos ambientes de tarefa do agente sendo testado Agent. Além do mais, enquanto os sensores de Agent percebem parcialmente o ambiente de tarefas CT em Amb, os sensores do agente MOA percebem todo o ambiente descrito em CT. O agente MOA foi concretizado como um agente reativo baseado em utilidades.

4.4.2.1 Primeira Etapa do Processo de Diagnóstico

O processo de diagnóstico de falhas é realizado em duas etapas principais, correspondentes ao processamento de seus dois subsistemas de processamento de informações:

- (E1) de atualização de estado interno – função próximo;
- (E2) de tomada de decisão – função ação.

Mais especificamente, na primeira etapa E1 o processamento do subsistema de atualização de estado interno do agente MOA começa a partir do momento em que, percebido o caso de teste CT $\in \Omega$ enviado pelo GERADOR, o subsistema simula e gera a história de comprimento N_{Int} de Agent em Amb associada ao caso CT, ou seja, h(CT)$\in$ (PxA)^{NInt}. Posteriormente, para cada episódio $Ep^K \in$ h(CT), o subsistema de atualização de estado interno executa três ações principais:

- (A1) gerar um conjunto de $\underline{n}$ episódios ideais possíveis na mesma interação K, ou seja, $Ep_{IDEAIS}{}^K$ = { $Ep_{ideal1}{}^K$, ..., $Ep_{idealn}{}^K$};
- (A2) inserir $Ep_{IDEAIS}{}^K$ na família de conjuntos de episódios ideais em todas as K-1 interações anteriores, Ep_{IDEAIS} = {$Ep_{IDEAIS}{}^1$, ..., $Ep_{IDEAIS}{}^{K-1}$}, obtendo uma nova família Ep_{IDEAIS} = {$Ep_{IDEAIS}{}^1$, ..., $Ep_{IDEAIS}{}^{K-1}$, $Ep_{IDEAIS}{}^K$};
- (A3) se Ep^K não pertencer ao conjunto $Ep_{IDEAIS}{}^K$ então inserir Ep^K no conjunto de episódios com falhas, Ep_{FALHAS} = {Ep^{K-1}, ..., Ep^{K-3}, ..., Ep^{K-k}}, e obter um novo conjunto Ep_{FALHAS} = {Ep^K, Ep^{K-1}, ..., Ep^{K-3}, ..., Ep^{K-k}}.

Mais especificamente, seja Ep^K = Ep(P^K, A^K) $\in$ PxA um episódio na interação $\underline{K}$ da história de Agent em Amb correspondente ao caso CT. Seja também Ep^{*K} = Ep(P^{*K}, A^{*K}) um episódio ideal na mesma interação $\underline{K}$ gerado pelo agente MOA, ou seja, $Ep^{*K} \in Ep_{IDEAIS}{}^K$, isto é, um episódio em que a ação racional A^{*K} foi selecionada por MOA em um ambiente de tarefas CT totalmente observável P^{*K}. Formalmente, o conjunto de episódios ideais em uma interação

K, $Ep_{IDEAIS}{}^{K}$, é formado por todos os n episódios possíveis de serem gerados por um agente racional em K (com o ambiente totalmente observável) que satisfazem pelo menos uma das duas condições abaixo:

- (C1) Para todo objetivo m na medida de avaliação de desempenho: $av_m(Ep(P^{*K}, A^{*K})) \geq av_m(Ep(P^K, A^K))$.
- (C2) A ação A^{*K} (selecionada pelo agente MOA) é melhor que ou é equivalente a ação A^K (selecionada pelo agente Agent) considerando o ponto de vista do Projetista.

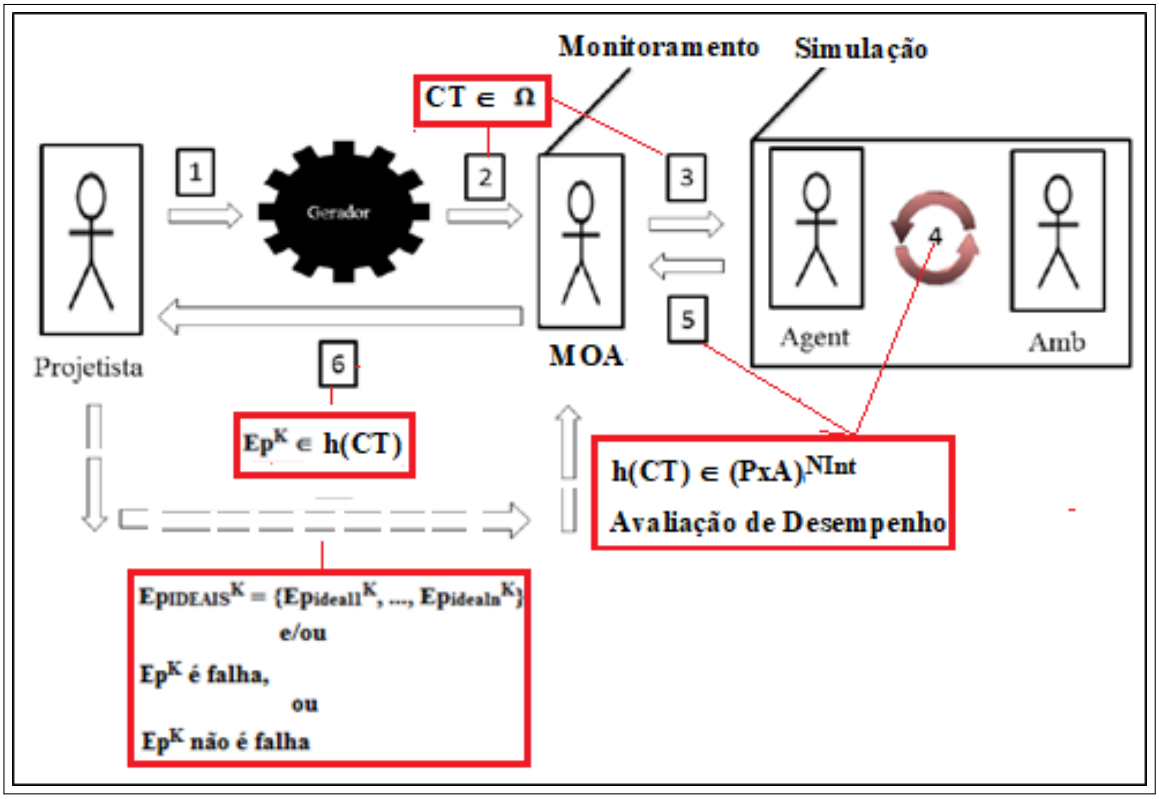
A condição C1 indica que, considerando os valores retornados por cada uma das funções escalares empregadas para medir o desempenho de Agent (como, por exemplo, as duas funções descritas na Tabela 2 para o caso de um agente aspirador de pó), o desempenho do agente MOA domina o desempenho do agente Agent, isto é: ação A^{*K} selecionada por MOA, racionalmente no ambiente totalmente observável, obtém desempenho melhor ou é igual que a ação A^K executada por Agent em um ambiente parcialmente observável.

A condição C2 foi introduzida por duas razões. Primeiro, esta condição permite ao Projetista decidir se uma ação é melhor que outra em situações em que a condição C1 não é satisfeita, ou seja, quando o agente MOA identificar que o seu desempenho não domina o desempenho do agente Agent, isto é: que a ação A^{*K} selecionada por MOA é pior que a ação A^K executada por Agent em pelo menos um dos objetivos na medida de desempenho, ou seja, existe pelo menos um objetivo x em que $av_x(Ep(P^K, A^{*K})) < av_x(Ep(P^K, A^K))$.

A segunda razão para a introdução de C2, a condição permite ao projetista julgar se existe a presença de falhas em um episódio em que o agente MOA está com dúvida se o agente Agent agiu corretamente, pois o mesmo está em um ambiente parcialmente observável. A Figura 19 específica melhor esta primeira situação em que o agente MOA acredita ser necessário interagir com o projetista de Agent, visando tirar dúvidas sobre os episódios ideais e episódios com falha em uma interação K.

Especificamente com relação à segunda razão para a interação do MOA com o Projetista, por exemplo, é possível perceber que as linhas 1, 5 e 6 na Tabela 2 identificam episódios que são falhas, ou seja, aspirar em uma sala limpa, e movimentar-se para uma sala vizinha ou não operar quando a sala corrente está suja. Entretanto, a Tabela não identifica episódios que são falhas em função de movimentos para salas vizinhas que não estão sujas ou retornos desnecessários às salas que já foram visitadas, como é o caso das falhas descritas nos episódios Ep^1, Ep^5 e Ep^6 na Figura 17. Assim, além da interação permitir identificar melhor os episódios que configuram falhas, nestes casos, o Projetista pode também enviar um conjunto de

Figura 19 – Interações do agente MOA com o projetista de Agent.



Fonte – Própria autora.

episódios ideais para o agente MOA.

4.4.2.2 Segunda Etapa do Processo de Diagnóstico

De maneira a colocar em prática esta interação, visando principalmente a atualização de estado interno realizado pela função próximo, o subsistema de tomada de decisão do MOA, realizado pela função ação, incorpora um conjunto de regras condição-ação que podem ser divididas em dois grupos, ou seja:

- (G1): aquelas regras que provocam a interação do MOA com o Projetista, quando for necessário ao subsistema de atualização de estado interno obter informações do projetista, para identificar os episódios ideias e com falha nas execuções das ações A2 e A3; e
- (G2): aquelas regras que utilizam as informações no estado interno atualizado de MOA para realizar o diagnóstico dos subsistemas de Agent que estão causando os episódios com falhas previamente identificados.

As regras condição-ação no grupo G2 só são utilizadas na tomada de decisão a partir do momento em que o subsistema próximo encerra as execuções das ações A1–A3, providenciando finalmente o conjunto Ep_{FALHAS}, isto é, contendo todos os episódios na história h(CT) que

podem ser considerados como falhas de Agent em Amb, e a família de conjuntos de episódios ideais correspondentes nas interações com falhas, Ep_{IDEAIS}. A Figura 20 apresenta um grupo de regras genéricas empregadas por MOA para diagnosticar os subsistemas de processamento de informações de Agent que estão causando a falha em cada episódio em Ep_{FALHAS}.

Figura 20 – Regras genéricas empregadas por MOA.

(1) O agente em teste Agent é do tipo reativo simples

Para todo episódio $Ep^K = Ep(P^K, A^K)$ no conjunto Ep_{FALHAS}, para todo conjunto de episódios ideais $Ep_{IDEAIS}^K = \{Ep(P^K, A_1{}^{*K}), ..., Ep(P^K, A_N{}^{*K})\}$ na família de conjuntos Ep_{IDEAIS}, **se** S^{*K} for o valor desejado na saída de $ver(P^K)$ em um ambiente totalmente observável **então**:

Falha na função ver de Agent **se** $ver(P^K) = S^K \neq S^{*K}$;

Falha na função ação de Agent **se** $ver(P^K) = S^K = S^{*K}$ **e** não existe uma ação ideal $A_X{}^{*K}$ no conjunto de episódios ideais Ep_{IDEAIS}^K tal que $ação(S^K) = A^K = A_X{}^{*K}$.

(2) O agente em teste Agent é do tipo reativo baseado em modelos, orientado por objetivo ou utilidade

Para todo episódio $Ep^K = Ep(P^K, A^K)$ no conjunto Ep_{FALHAS}, para todo conjunto de episódios ideais $Ep_{IDEAIS}^K = \{Ep(P^K, A_1{}^{*K}), ..., Ep(P^K, A_N{}^{*K})\}$ na família de conjuntos Ep_{IDEAIS}, **se** S^{*K} for o valor desejado de $ver(P^K)$ em um ambiente totalmente observável **e** I^* for o valor desejado de $próximo(ver(P^K), I^{*K-1})$ em ambiente parcialmente observável **então**:

Falha na função ver de Agent **se** $ver(P^K) = S^K \neq S^{*K}$;

Falha na função próximo de Agent **se** $próximo(ver(P^K), I^{*K-1}) \neq I^{*K}$

Falha na função ação de Agent **se** $próximo(ver(P^K), I^{*K-1}) = I^{*K}$ **e** não existe uma ação ideal $A_X{}^{*K}$ no conjunto de episódios ideais Ep_{IDEAIS}^K tal que $ação(I^K) = A^K = A_X{}^{*K}$.

Fonte – Própria autora.

As duas primeiras regras são aplicadas aos casos em que o agente testado Agent é do tipo reativo simples. A primeira regra indica que a falha está no subsistema de percepção de Agent, se a sua função ver gerar como saída uma descrição de estado, S^K, que é diferente da descrição desejada, S^{*K}, em um ambiente totalmente observável, conforme percebido por MOA.

A segunda regra indica que a falha está no subsistema de tomada de decisão de Agent, isto é, que apesar de seu subsistema de percepção estar funcionando bem, pois $S^K = S^{*K}$, o agente Agent não conseguiu escolher uma ação racional no episódio sobre consideração, visto que a ação que ele selecionou A^K não faz parte de nenhum dos episódios ideais gerados na interação K.

As três últimas regras são aplicadas quando o agente testado Agent é de um dos três tipos de agentes que possuem um subsistema de atualização de estado interno. Assim como no caso reativo, a primeira regra indica que a falha está no subsistema de percepção de Agent. A segunda regra indica que a falha está no subsistema de atualização de estado interno de Agent, ou seja, a função próximo gera uma descrição de estado interno que é diferente daquilo que se deseja em um ambiente com observabilidade total, I^{*K}, conforme a percepção do agente MOA.

A terceira regra indica que a falha está no subsistema de tomada de decisão de Agent, ou seja, apesar do subsistema de atualização estar funcionando bem, Agent não conseguiu escolher uma ação racional no episódio sobre consideração, isto é, da mesma forma que os agentes reativos, para toda ação A^{*K} em $Ep_{IDEAIS}{}^{K}$, tem-se que $A^{K} \neq A^{*K}$.

5 EXPERIMENTOS

Este capítulo tem como objetivo apresentar os experimentos que foram gerados utilizando a abordagem interativa para o problema de seleção de casos de teste de agentes racionais.

5.1 O AMBIENTE DE TAREFAS E OS AGENTES TESTADOS

O ambiente de tarefas utilizado para validar a abordagem proposta é formado por n x n salas, onde o valor de n é determinado pelo Projetista. Além da quantidade de salas, o ambiente criado pode ser diferenciado pela quantidade de salas sujas e a localização das mesmas. O ambiente pode possuir propriedades diferentes dependendo do tipo de interação do Projetista. Caso o Projetista não realize interações durante a execução dos testes, então o ambiente será do tipo determinístico e estático, isto é, o tempo não altera o estado do ambiente, o próximo estado do ambiente é determinado pelo estado atual e a ação executada. Entretanto, se o Projetista realizar interações durante os testes, o ambiente será estocástico e dinâmico, ou seja, o estado do ambiente pode ser alterado ao longo do tempo ou pelo Projetista. Para a avaliação de desempenho foram considerados critérios para energia e limpeza.

O Quadro 1 ilustra as configurações da medida de avaliação de desempenho para o caso de um programa agente aspirador de pó, em que o intuito é maximizar a limpeza e minimizar a utilização de energia. A primeira coluna apresenta as possíveis percepções de Agent em cada um dos episódios(Limpa(Lim) e Suja(Suj)). A próxima coluna descreve as possíveis ações referentes a cada episódio(Direita(Dir), Esquerda(Esq), Acima(Aci), Abaixo(Aba), Não operar(N-ope) e Operar(Ope)). Na terceira e na quarta coluna são apresentadas, respectivamente, Limpeza e Energia.

Quadro 1 – Medidas de avaliação de desempenho.

Percepção	Ação	Limpeza	Energia
..., Lim, ...	Dir, Esq, Aci, Aba	1.0	-2.0
..., Suj,...	Dir, Esq, Aci, Aba	- 1.0	-2.0
..., Suj,...	N-ope	-1.0	0.0
..., Suj,...	Ope	1.0	-1.0
..., Lim, ...	N-ope	0.0	0.0
..., Lim, ...	Ope	-2.0	-1.0

Fonte – Elaborado pela autora.

Para o teste da abordagem proposta foram criados dois tipos de agentes, reativo

simples e baseado em modelos. Para o tipo reativo simples foram criados dois agentes com diferentes tipos de observabilidade do ambiente, (i) observabilidade parcial, em que Agent só percebe o estado do nodo em que ele se encontra; (ii) observabilidade total, onde Agent possui acesso ao estado de todos os nodos do ambiente em cada instante. O Algoritmo 1 apresenta as regras de condição-ação do agente reativo com observabilidade parcial.

Algoritmo 1: Regras condição-ação do agente reativo simples com observabilidade parcial.

início
...
se percepção = Lim **então** ação = selecionar(Dir, Esq, Aci, Aba); **se** percepção = Suj **então** ação = Ope;
...
fim

Para os agentes do tipo reativo simples as ações são baseadas apenas na percepção, já que os mesmos não possuem um estado interno. O agente percebe o estado do ambiente e apartir da percepção seleciona a ação a ser executada. O Algoritmo 2, apresenta alguns exemplos de regras de condição-ação presentes no programa agente reativo simples com observabilidade total.

Algoritmo 2: Regras condição-ação do agente reativo simples com observabilidade total.

início
...
se percepção = Lim e Sala(Aci) = Suja **então** ação = Aci; **se** percepção = Lim e Sala(Aba) = Suja **então** ação = Aba; **se** percepção = Lim e Sala(Dir) = Suja **então** ação = Dir; **se** percepção = Lim e Sala(Esq) = Suja **então** ação = Esq; **se** percepção = Suj **então** ação = Ope;
...
fim

Foi criado também um agente reativo baseado em modelos com observabilidade parcial, ou seja, o agente percebe apenas uma fração do ambiente. Diferentemente do agente reativo simples, este possui um modelo interno que mantém o estado atual do mundo. O agente armazena informações sobre o estado interno, que dependem do histórico das percepções e percebe os aspectos não observados no estado atual.

Algoritmo 3: Regras condição-ação do agente reativo baseado em modelos com observabilidade parcial.

início
...
se percepção = Suj **então** ação = Ope; **se** percepção = Lim e Sala(Aci) = NãoVisitada **então** ação = Aci; **se** percepção = Lim e Sala(Aba) = NãoVisitada **então** ação = Aba; **se** percepção = Lim e Sala(Esq) = NãoVisitada **então** ação = Esq; **se** percepção = Lim e Sala(Dir) = NãoVisitada **então** ação = Dir; **se** percepção = Lim e TodasSalas = Visitadas
textbfentão ação = escolha(Aci, Aba, Esq, Dir);
...
fim

O subsistema de percepção realiza o mapeamento das informações sobre o estado do ambiente, por conseguinte, o subsistema de atualização de estado interno, mapeia a percepção atual e as informações sobre o estado interno são mantidas pelo agente em um novo estado interno. Por último, o subsistema de tomada de decisão, mapeia a informação sobre estado interno em uma ação correspondente e atualiza novamente o estado interno. Para exemplificar melhor o Algoritmo 3 descreve de forma geral as características das regras de condição-ação.

5.2 FERRAMENTA PARA REALIZAÇÃO DOS EXPERIMENTOS

A Figura 21 apresenta inicialmente as opções disponíveis para o Projetista. A primeira opção é Teste de Agentes, onde o Projetista ao selecionar esta opção pode escolher o Agent a ser testado, o modo de geração que pode ser automático ou manual, a partir da escolha ele poderá seguir com o teste. Inicialmente, o gerador automático, com exceção do modo frequência de geração, gera 10 ambientes, entretanto o Projetista, caso deseje, pode alterar a quantidade de ambientes gerados no modo automático, na Figura 24 e Figura 25 será abordado melhor este assunto. Na opção MOA o Projetista tem acesso as funções do agente de monitoramento. Na opção Gráficos é possível analisar os gráficos de desempenho de Agent de acordo com a avaliação de desempenho. Nos Arquivos de Log, estão disponíveis as histórias de Agent e os valores da avaliação de desempenho, episódio por episódio.

Caso o Projetista selecione a opção Exportar Dados é possível exportar para outro dispositivo as histórias de Agent e os valores da avaliação de desempenho, episódio por episódio. Já na opção Sistema, estão presentes as informações da ferramenta, além das configurações do

banco de dados que podem ser acessadas caso o Projetista tenha a necessidade de pesquisar, excluir ou alterar algum caso de teste que estaja salvo.

Figura 21 – Tela inicial da ferramenta.

Fonte – Própria autora.

Após selecionar as configurações necessárias/desejadas para a realização dos testes, o Projetista pode gerar os ambientes e ter acesso aos botões que possibilitam realizar ações como inciar os testes. A Figura 22 ilustra a tela inicial e contém todas as opções dos botões disponíveis. Os botões possuem as seguintes funcionalidades, respectivamente, iniciar a execução dos testes, pausar, cancelar, aumentar o tamanho do ambiente, diminuir o tamanho do ambiente, alterar estados dos nodos e ajuda, caso o Projetista possua alguma dúvida em relação ao funcionamento da ferramenta.

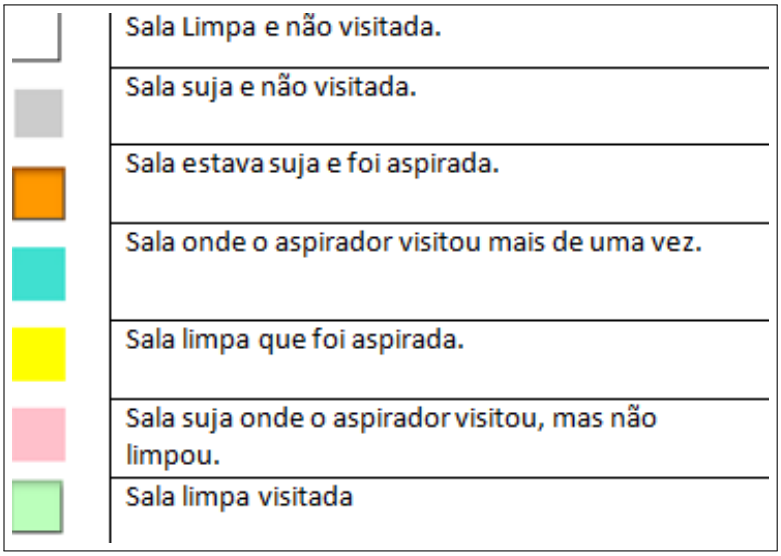
Vale ressaltar que o botão para alterar os estados dos nodos só é habilitado após os testes serem pausados. Durante o desenvolvimento da abordagem, fez-se necessário um mecanismo para auxiliar na identificação das ações do agente, antes, durante e após os testes, para isso foram designadas as seguintes cores para os nodos dependendo do estado dos mesmos. Baseado no exemplo do mundo do aspirador de pó, pode-se considerar os possíveis estados representados na Figura 23.

Ao iniciar os testes, o Projetista poderá definir se ele deseja configurar os parâmetros

Figura 22 – Tela inicial da ferramenta com ambientes gerados.

Fonte – Própria autora.

Figura 23 – Representação do estado dos nodos para o exemplo do mundo do aspirador de pó, de acordo com o estado.



Fonte – Própria autora.

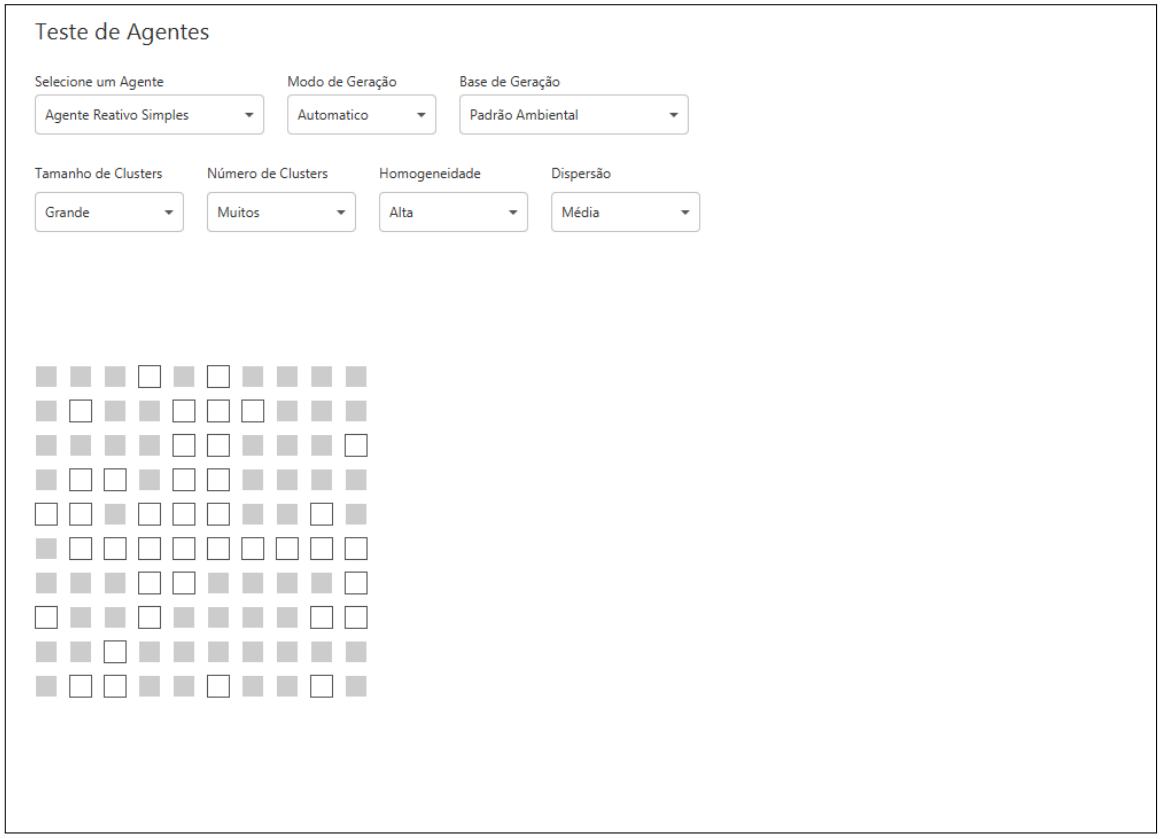
de geração dos casos de teste manualmente ou se o mesmo deseja que o teste seja gerado de maneira automática. Na Situação 1, apresentada na Seção 4.2, o Projetista gera o ambiente de modo automático selecionando as características do padrão ambiental desejado para a realização dos testes.

Na Figura 24, pode-se observar todas as características citadas na Seção 4.3.2, onde os ambientes podem ser gerados de acordo com as características escolhidas pelo Projetista. Ao selecionar a base de geração por padrão ambiental o Projetista possui a opção de selecionar o tamanho dos *clusters*, o número de *clusters*, a homogeneidade e a dispersão.

Todas estas características são representadas por variáveis linguísticas baseadas nos

conceitos da lógica nebulosa, como por exemplo, ao selecionar o tamanho do *cluster* o Projetista pode gerar um ambiente com *clusters*: muito grande, grande, médio, pequeno ou muito pequeno. Para isto foram definidas as partições fuzzy e seus intervalos para cada variável de entrada.

Figura 24 – Configuração com padrão ambiental ao ser gerado.



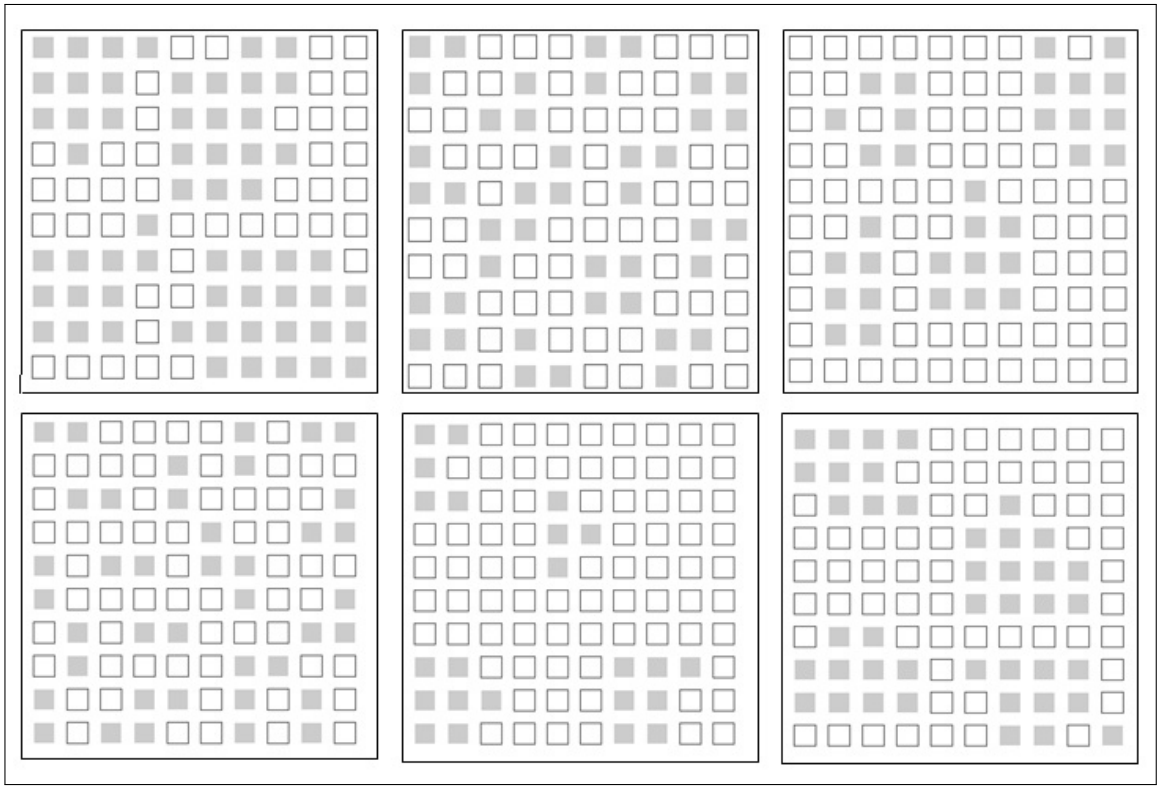
Fonte – Própria autora.

Portanto a ferramenta considera as quatro características do padrão ambiental com graus diferentes de intensidade. Considerando a Figura 25 que apresenta seis ambientes criados com diferentes características de padrão ambiental. O primeiro ambiente foi gerado com as seguintes características:(i) tamanho dos *clusters*: grande; (ii) número de *clusters*: pouco; (iii) homogeneidade: média; (iv) dispersão: média. Cada um dos seis ambientes gerados possui características diferentes.

Foram criados, especificamente para testar agentes do mundo do aspirador de pó, outras opções para a geração de ambientes de maneira manual, neste exemplo, o Projetista deve escolher a quantidade de salas a serem criadas para o ambiente e o modo como será gerado a sujeira. Para a geração da sujeira a abordagem propõe: (i) Personalizar nodos; (ii) utilizando o percentual e; (iii) de maneira aleatória.

Considerando a geração personalizada do ambiente, o Projetista cria o ambiente

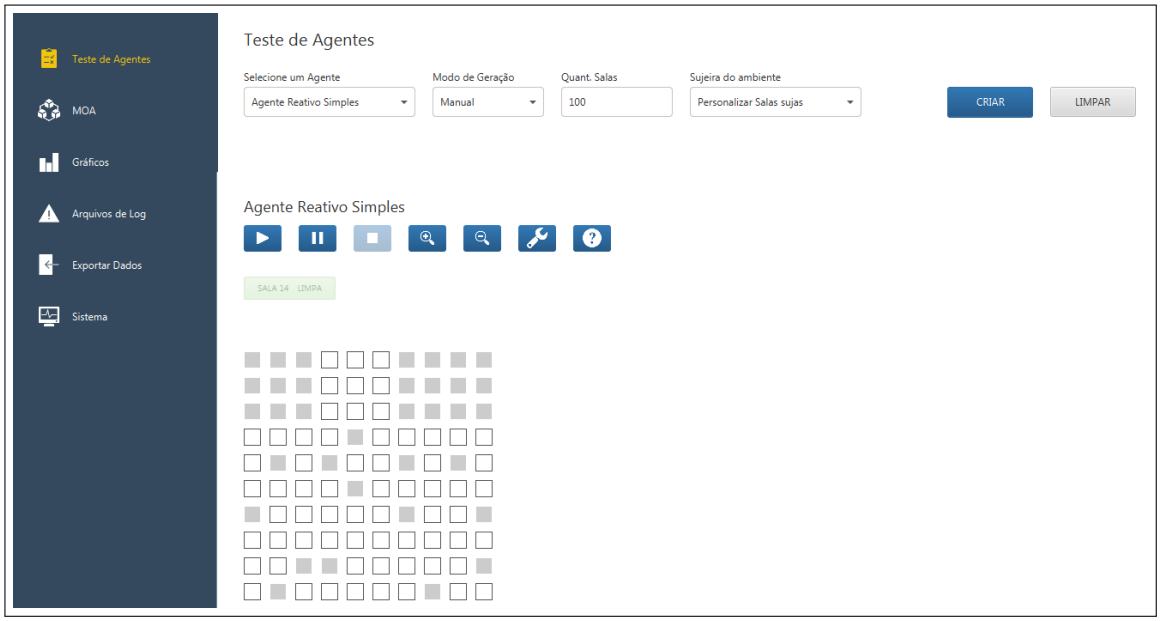
Figura 25 – Exemplos de ambientes com padrão ambiental.



Fonte – Própria autora.

de teste com a intenção de testar uma funcionalidade especifica ou até mesmo para comparar o comportamento dos agentes testados, utilizando o conhecimento tácito para realizar esta atividade.

Figura 26 – Personalizar Nodos.

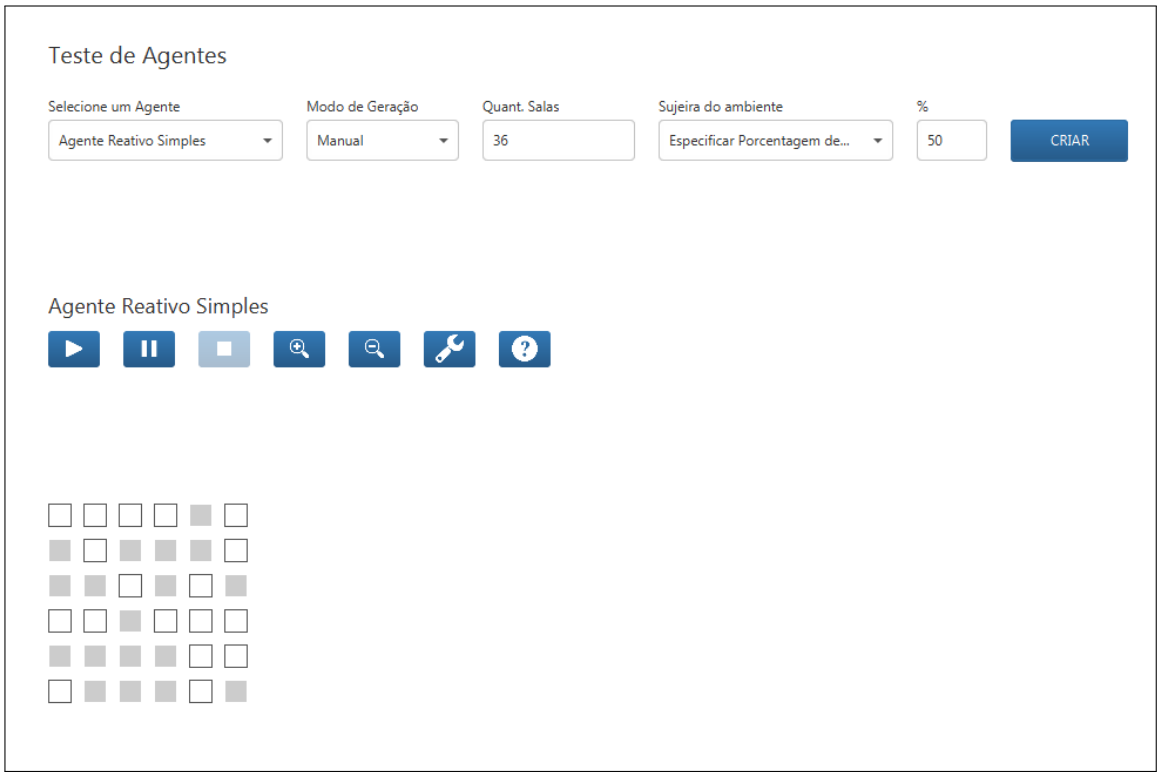


Fonte – Própria autora.

Neste modo de geração é possível pesonalizar os nodos do ambiente, no exemplo do mundo do aspirador de pó, seria inserir a sujeira nas salas. Quando o Projetista seleciona a sala, a mesma possui a cor cinza para diferenciar das limpas que são brancas. Com isso o Projetista consegue gerar o ambiente que deseja e perceber de maneira mais especifíca o comportamento que o mesmo deseja avaliar em Agent. Esta geração pode ser observada na Figura 26.

No modo de geração de sujeira por percentual os casos de teste são gerados com base na quantidade de salas que o Projetista informa e a quantidade de sujeira distribuída aleatoriamente nas salas criadas. Por exemplo, na Figura 27, é apresentado um ambiente, onde o Projetista escolheu a geração manual do caso de teste e definiu um único ambiente com 36 salas com uma porcentagem de sujeira de 50%. Gerando assim, 18 salas sujas neste ambiente. Esta configuração é ideal para gerar uma quantidade exata de sujeira, porém com uma maior agilidade que pelo modo de sujeira personalizado.

Figura 27 – Configuração do percentual de sujeira.

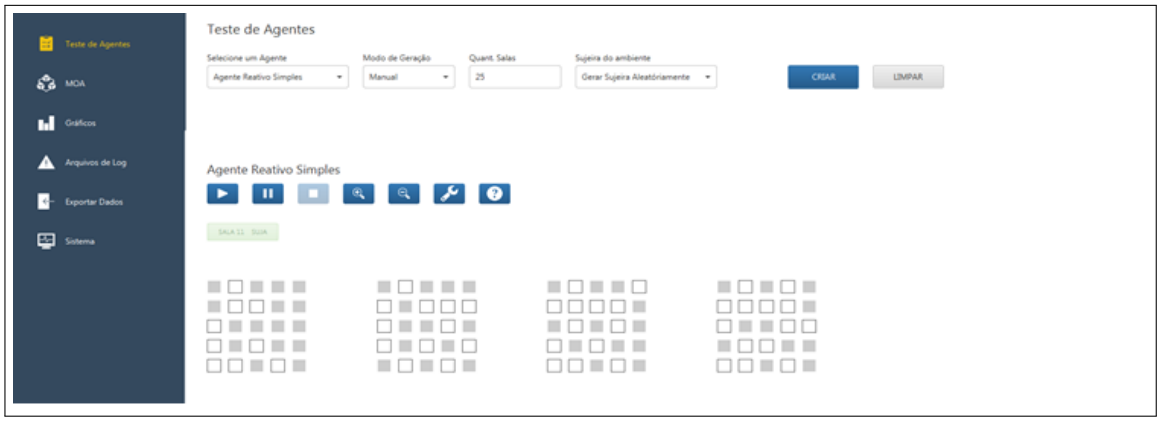


Fonte – Própria autora.

Já no modo aleatório, o Projetista tem a liberdade de gerar o ambiente escolhendo apenas a quantidade de salas e distribuindo de maneira aleatória a sujeira, considerando a quantidade de salas. Contribuindo para que o Projetista possa gerar e perceber o comportamento do agente de forma rápida e efetiva.

A Figura 28, apresenta um dos ambientes gerados utilizando o GERADOR, representando o mundo do aspirador de pó, onde Agent, será testado. Foi selecionado o Agent do tipo agente reativo simples com observabilidade parcial, O ambiente foi gerado com 25 nodos ou salas, com a distribuição da sujeira de modo aleatório e com a geração de apenas quatro ambientes, para que o projetista tenha um maior acompanhamento dos resultados.

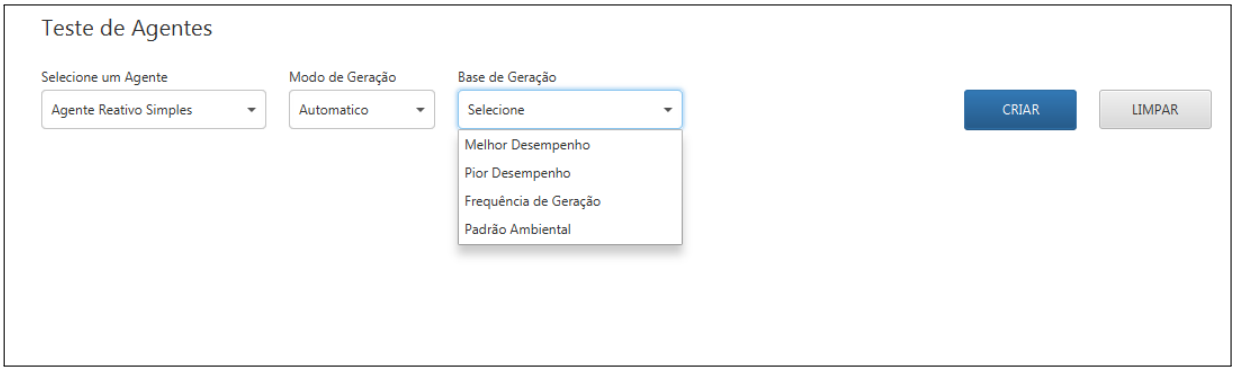
Figura 28 – Ambiente configurado a ser testado.



Fonte – Própria autora.

Se o Projetista desejar gerar o teste automático, o agente GERADOR verificará se foram realizados testes anteriores com o tipo de agente a ser testado, em caso negativo, o agente GERADOR irá gerar os casos de teste com as configurações padrões do sistema, que foram setadas no desenvolvimento da ferramenta, porém se for positivo, ou seja, se houverem outros testes já realizados para o tipo de agente a ser testado o agente será gerado com base nas configurações anteriores. A Figura 29 ilustra a tela inicial para a Situação 2, abordada na Seção 4.2.

Figura 29 – Tela inicial para Situação 2.



Fonte – Própria autora.

Além da geração por padrão ambiental, denominada como Situação 1, há mais três opções de geração automática, (i) pelo melhor desempenho, onde são gerados os casos de teste com os mesmos parâmetros dos melhores desempenho do tipo de agente a ser testado, realizando uma comparação entre os resultados da avaliação dos mesmos; (ii) outra opção é gerar os casos de teste com os mesmos parâmetros dos piores casos de teste, ou seja, quando o tipo de agente a ser testado não foi bem avaliado e; (iii) pela frequência de geração, onde são consultadas todas as configurações já escolhidas, calculando a frequência absoluta das configurações em que são selecionadas as que foram escolhidas com maior frequência, com exeção da quantidade de sujeira, que além da frequência absoluta também é calculada a média aritmética da quantidade de vezes que a mesma foi selecionada. A seguir são apresentados no Quadro 2 um exemplo, neste caso, o GERADOR irá gerar um ambiente com 16 salas .

Quadro 2 – Exemplo: Geração de salas utilizando a opção de frequência de geração anterior.

Frequência de geração	Quantidade de salas criadas
5	36
3	25
12	16
7	9

Fonte – Elaborado pela autora

Vale resaltar que, se houverem frequências iguais, será escolhido de maneira aleatória as quantidades do conjunto de maiores frequências. Isto ocorre em todas as configurações utilizadas.

Quadro 3 – Exemplo: Geração de quantidade de sujeira utilizando a opção de frequência de geração anterior

Frequência de geração	Quantidade de salas sujas
3	6
6	8
2	5
1	10

Fonte – Elaborado pela autora

Após ser calculada a frequência absoluta da quantidade de salas e definido a maior frequência utilizada. São selecionadas as quantidades de sujeira que foram mais utilizadas para que seja calculada a média aritmética. Considerando o Quadro 3 pode-se calcular a quantidade

de sujeira:

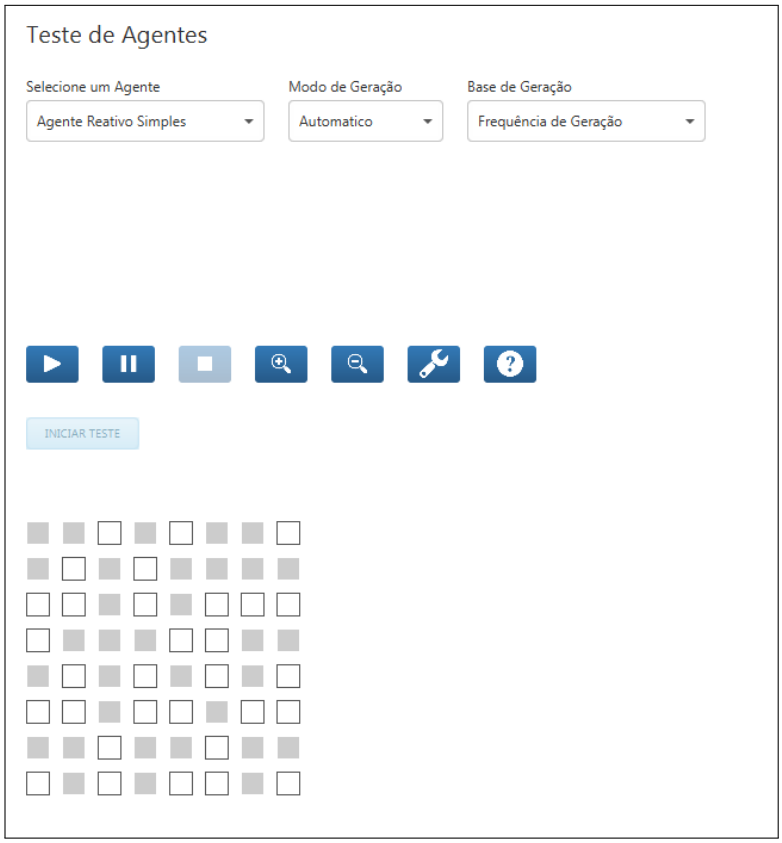
$$qtdSalasSujasGerar = \frac{\sum_{n}^{1}}{frequencia} \tag{5.1}$$

A média calculada será o somatório da quantidade de salas sujas dividido pela frequência. Considerando o Quadro 3 utilizando-se das regras de arredondamento foi calculado o resultado de sete salas sujas no ambiente que possui 16 salas, para um dos casos de teste gerado, os outros casos de teste gerado, terão uma discrepância de uma sala suja ou limpa, de forma aleatória.

Na abordagem proposta, é aconselhável que o Projetista inicie a realização dos testes utilizando o modo manual, para que o mesmo possa gerar as preferências dos testes e conhecer melhor o comportamento do agente a ser testado, por meio da análise dos testes.

Na Figura 30, foi selecionado a geração pela frequência de geração, onde o ambiente gerado possui 31 salas limpas e 33 salas sujas. Pode-se perceber que o Agent selecionado foi o aspirador de pó do tipo reativo simples.

Figura 30 – Configuração dos Parâmetros no Modo Manual.

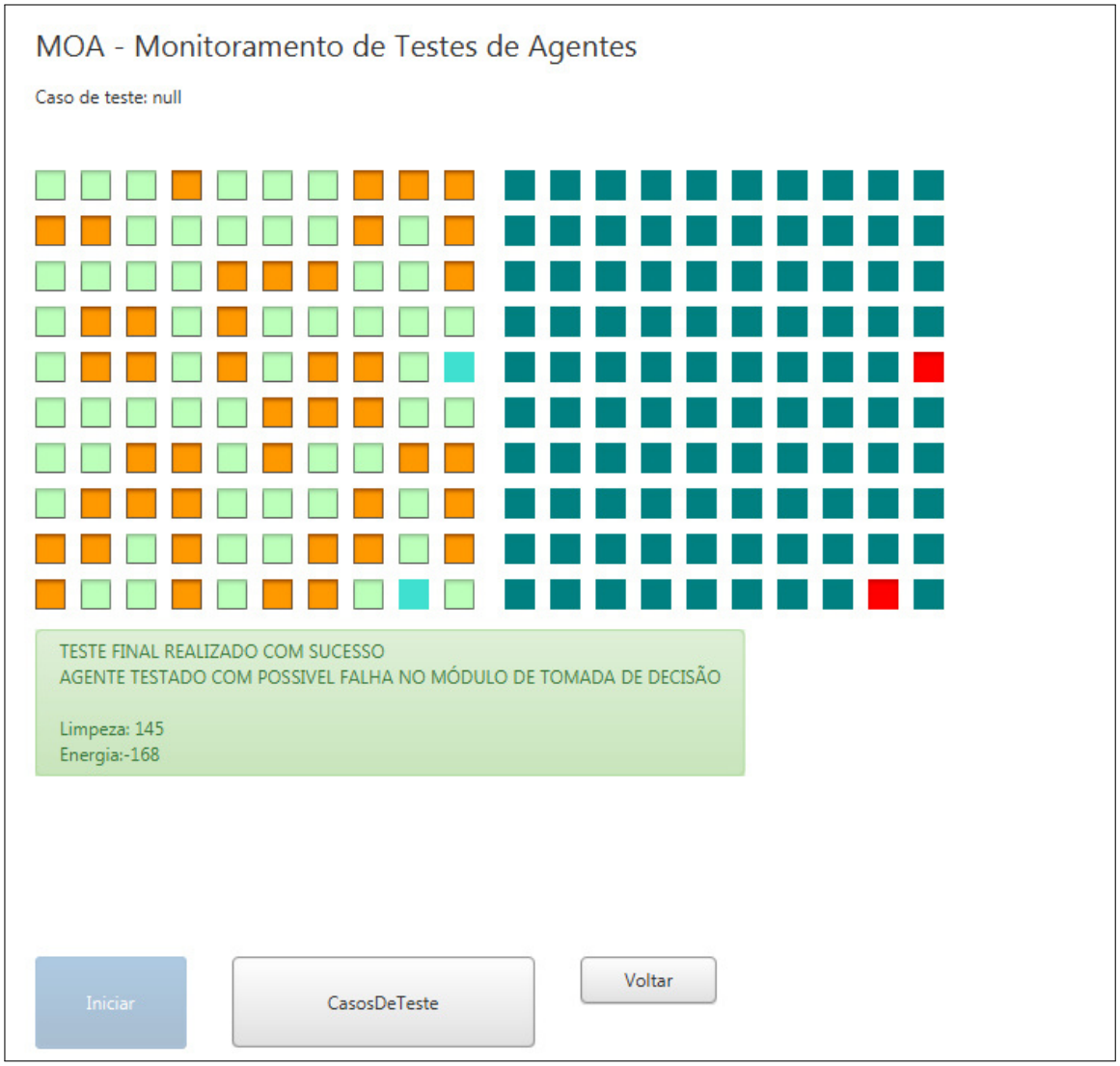


Fonte – Própria autora.

A Figura 31 apresenta um exemplo de monitoramento realizado pelo MOA. Pode-se perceber que, o agente MOA destaca as falhas na cor vermelha e os nodos que estão com o comportamento de acordo com o esperado ficam preenchidos com a cor azul, isto auxilia o Projetista na percepção de erros que podem não ser detectados na execução dos testes.

Além da identificação é possível que o projetista possa analisar com seu conhecimento tácito se realmente ocorreu uma falha que precisa ser corrigida ou se o comportamento inadequado foi gerado pelo tipo do Agent em questão. O MOA também sugere o possível módulo que esta gerando a falha e a pontuação de energia e limpeza do determinado caso de teste.

Figura 31 – Tela de execução do monitoramento com MOA.

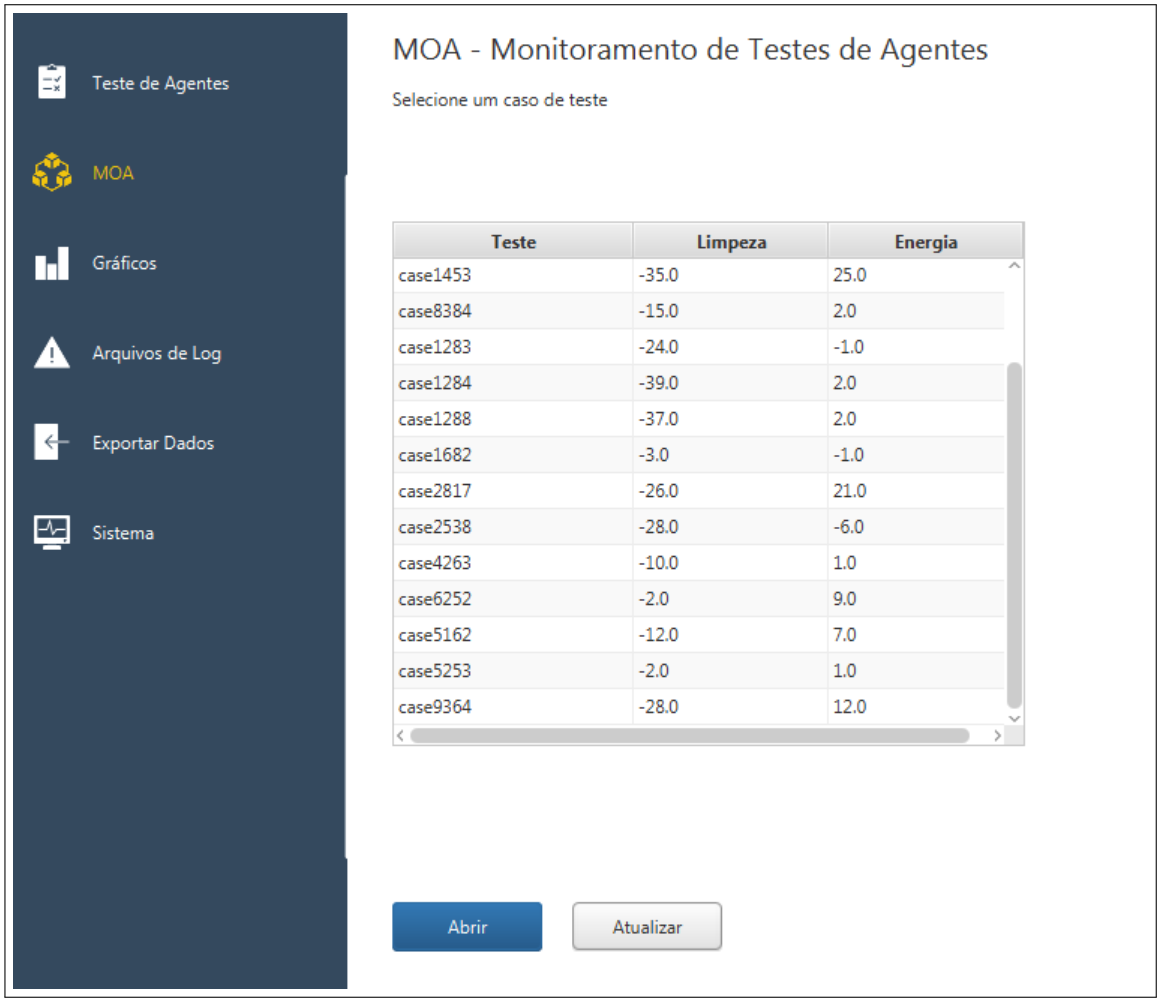


Fonte – Própria autora.

Os casos de testes ficam armazenados e o Projestista pode ter acesso aos mesmos a

qualquer momento, podem ser realizadas operações de manipulação destes arquivos, inclusive o arquivamento destes casos de testes para fins de documentação. A Figura 32 ilustra a visão do projetista ao acessar os arquivos de casos de teste.

Figura 32 – Casos de teste salvos.



Fonte – Própria autora.

5.3 SELEÇÃO DE CASOS DE TESTE PARA O PRIMEIRO MODO DE INTERAÇÃO

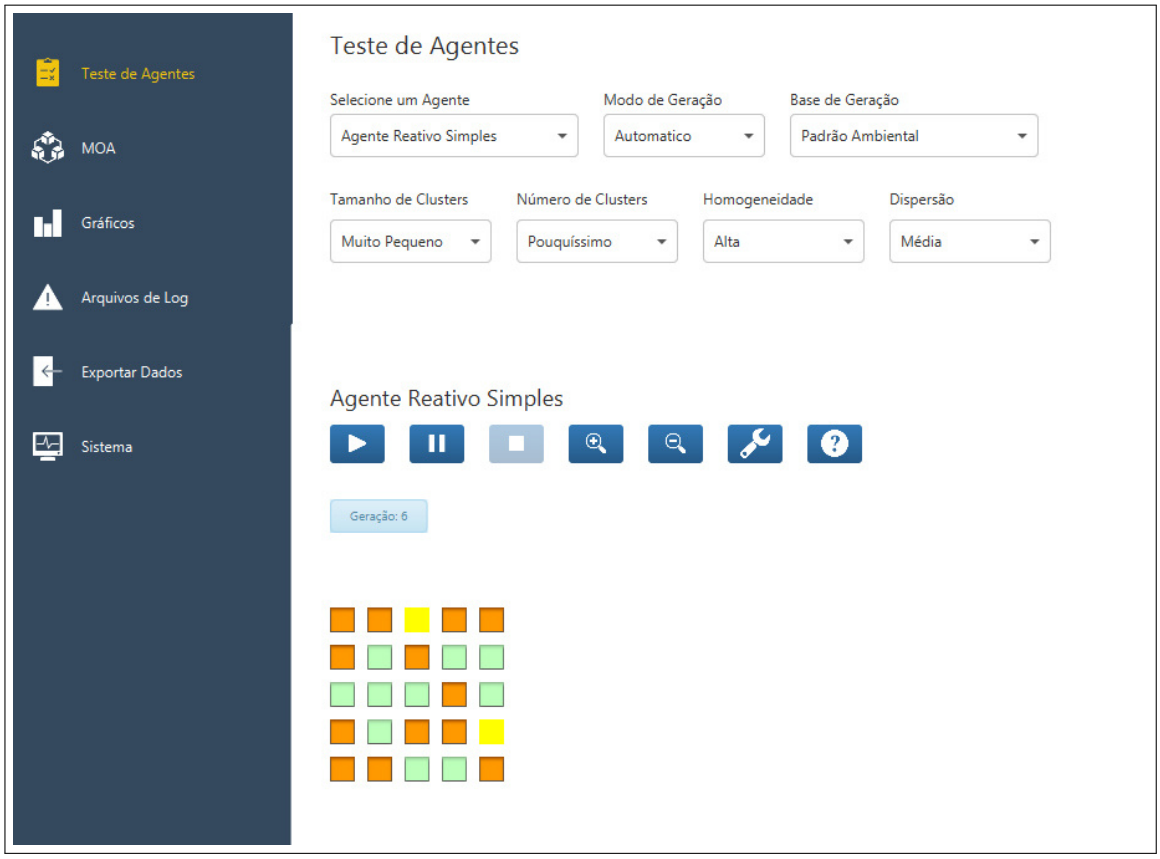
5.3.1 Agente Reativo Simples

Vale salientar que na Situação 1, o Projetista interage criando ambientes de acordo com padrões ambientais que o mesmo necessita. Portanto, foi criado um ambiente obedecendo as características do padrão ambiental, com pouquíssimos *clusters* de tamanho muito pequenos, com alta homogeneidade e média dispersão. Após a configuração e geração do ambiente, os testes foram iniciados, conforme Figura 33 que apresenta os resultados gerados, onde na sexta

geração foram identificadas duas falhas do mesmo tipo em que Agent operou em sala limpa.

Deve-se ressaltar que o Projetista tem a opção de parar a execução para verificar o erro e alterar o ambiente ou continuar realizando a análise e apenas no final dos testes realizar o diagnóstico utilizando o MOA. Neste caso, Agent não teve um bom desempenho em relação as outras gerações dos casos de teste e apresentou um comportamento diferente no ambiente.

Figura 33 – Exemplo de seleção de casos de teste para o modo de interação 1 com agente reativo simples.

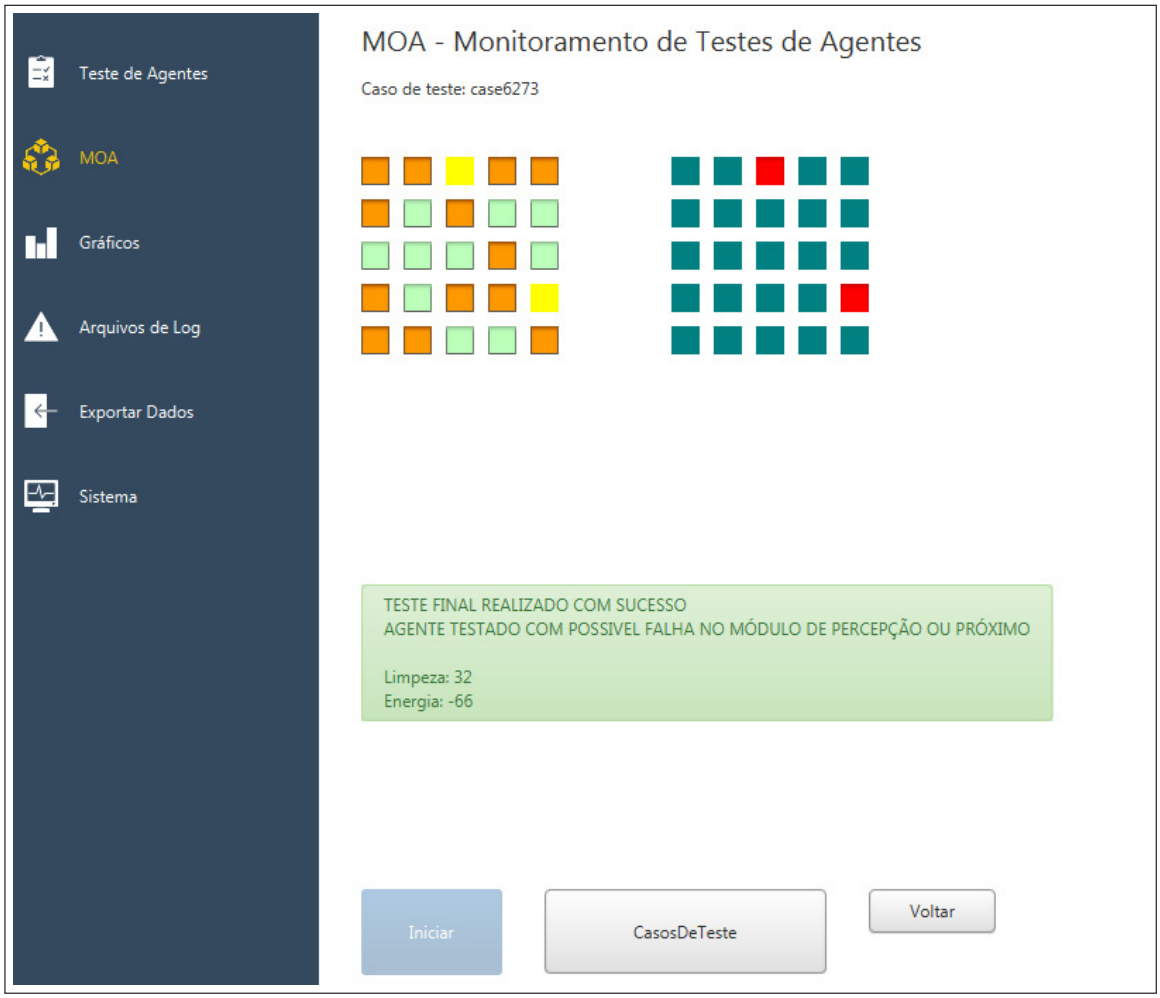


Fonte – Própria autora.

Na Figura 34, o agente MOA destacou a falha encontrada na cor vermelha e diagnosticou que Agent pode possuir falha no módulo de percepção ou no médulo próximo, há uma possiblidade de Agent ter percebido sujeira no ambiente limpo ou o módulo próximo ter sugerido a ação de aspirar um ambiente percebido como limpo.

O Projetista tem a possibilidade de executar novamente o teste utilizando o mesmo cenário, e o mesmo Agent ou outro Agent do mesmo tipo ou de tipo diferente, ou o Projetista poderá somente salvar o caso de teste para consultas ou testes posteriores. Agent utilizou 32 de limpeza e -66 de energia.

Figura 34 – Diagnóstico realizado por MOA de possível módulo com falha do caso de teste da Figura 33.



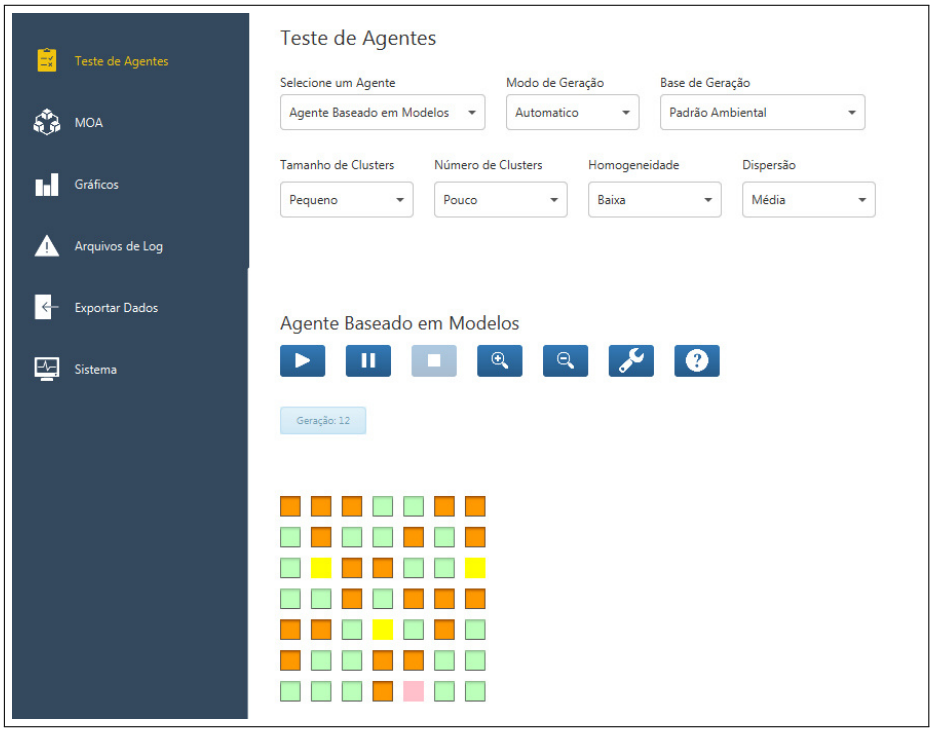
Fonte – Própria autora.

5.3.2 Agente Reativo Baseado em Modelos

Para testar Agent do tipo reativo baseado em modelos foi criado um ambiente com 49 salas com poucos *clusters* de tamanho pequeno, com baixa homogeneidade e média dispersão. Na décima segunda geração Agent cometeu algumas falhas, como pode ser observado na Figura 35. Pode-se perceber que Agent aspirou salas que não haviam sujeira e não operou em um dos nodos que possuia sujeira. O Projetista enviou este caso de teste para o MOA que realizou o diagnóstico, como pode ser observado na Figura 36.

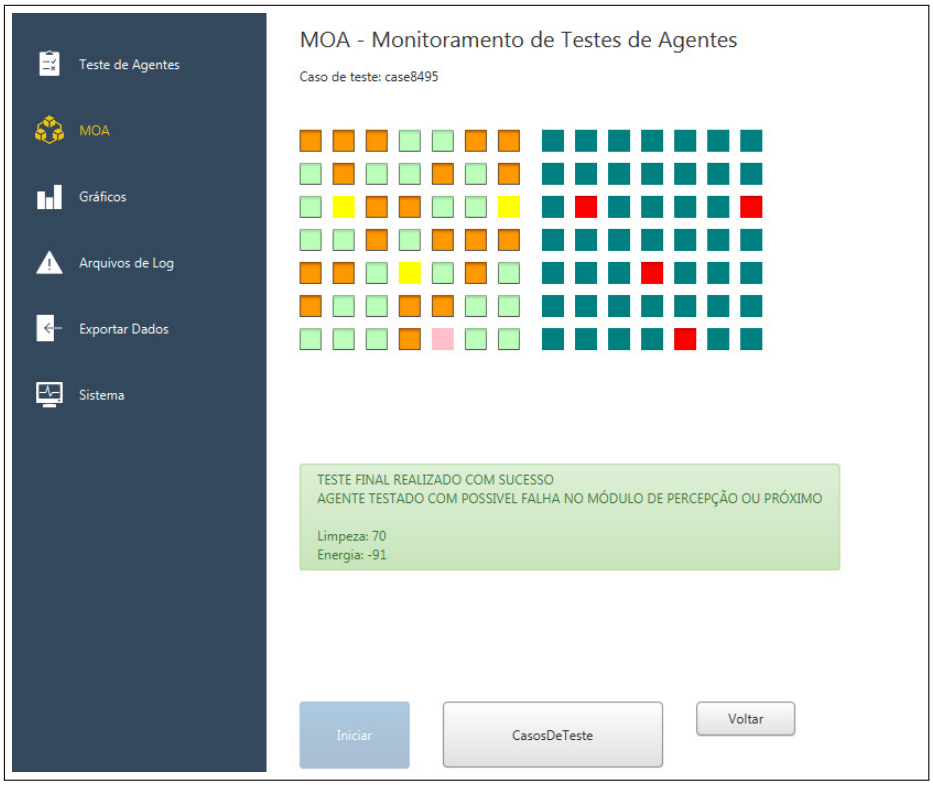
O MOA sugere que possivelmente a falha esta no módulo de percepção ou no próximo. Ou seja, possívelmente o agente não esta percebendo corretamente ou o módulo próximo esta sugerindo ações incorretamente. O Projetista pode realizar as alterações necessárias e continuar testando Agent ou ele pode testar com outros tipos de ambientes e analisar melhor a

Figura 35 – Exemplo de seleção de casos de teste para o modo de interação 1 com agente reativo baseado em modelos.



Fonte – Própria autora.

Figura 36 – Funcionamento do MOA após execução do ambiente da Figura 35.



Fonte – Própria autora.

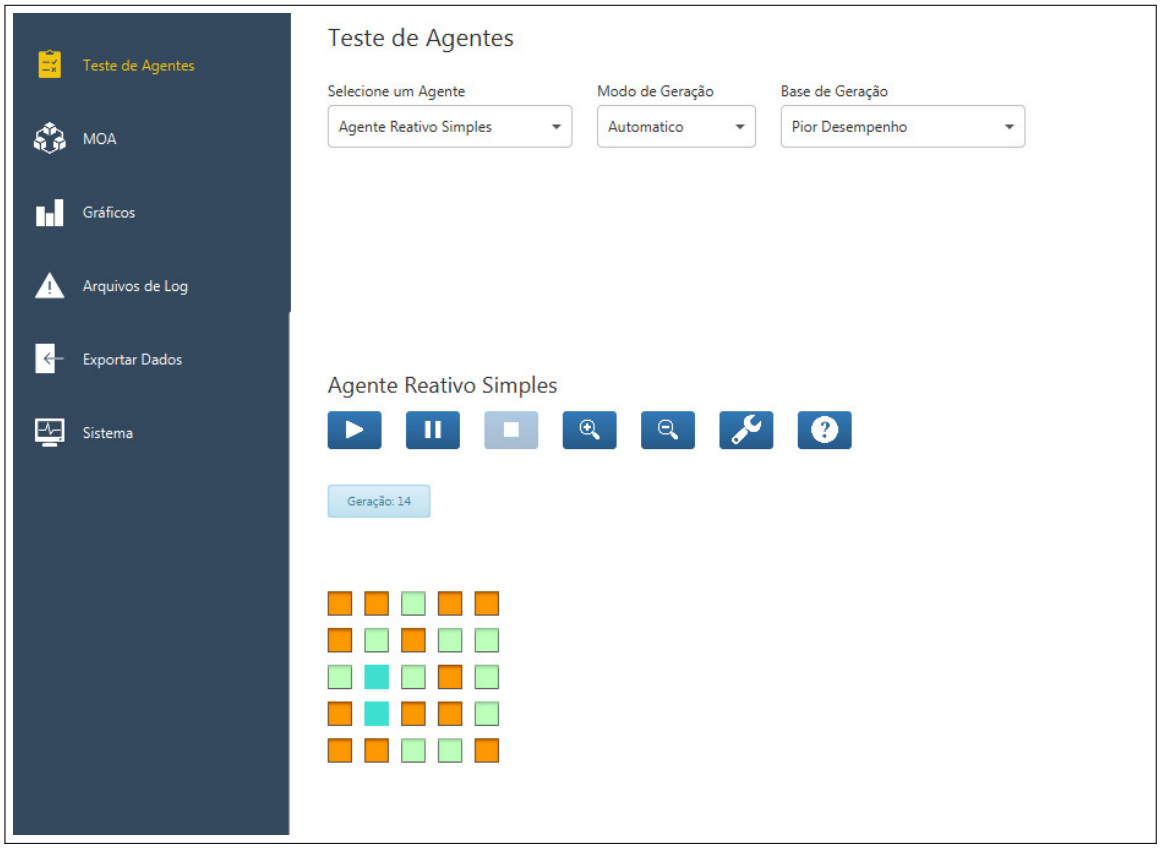
falha que esta sendo gerada.

5.4 SELEÇÃO DE CASOS DE TESTE PARA O SEGUNDO MODO DE INTERAÇÃO

5.4.1 Agente Reativo Simples

Como já abordado anteriormente, na Situação 2 o Projetista pode utilizar um caso de teste salvo para realizar uma nova análise de Agent. A Figura 37 apresenta o caso de teste: 6273 em que foi testado um agente aspirador de pó do tipo reativo simples, e que foi selecionado para testar um outro agent do mesmo tipo com observabilidade parcial.

Figura 37 – Ambiente de teste selecionado a ser executado novamente com outro Agent.

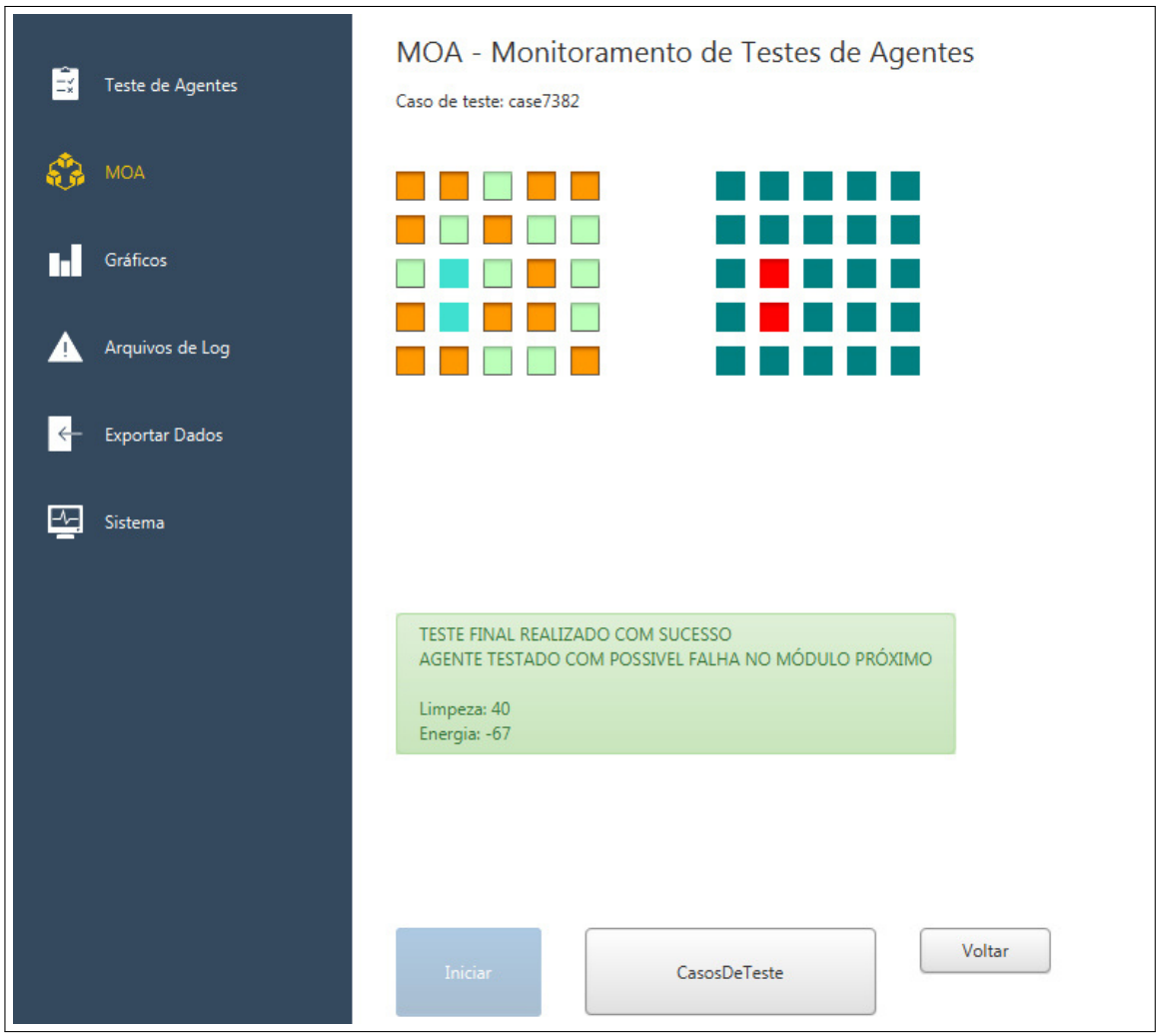


Fonte – Própria autora.

Observa-se que Agent demonstrou falha na décima quarta geração, onde dois nodos foram visitados mais de uma vez. Para confirmar e diagnosticar a falha o Projetista submeteu o caso de teste ao MOA, conforme Figura 40.

O Agent testado apresentou falha ao visitar um nodo mais de uma vez. O MOA sugeriu uma possível falha no módulo próximo, o Projetista pode considerar também que o agente testado não possui estado interno podendo então ser um comportamento considerado

Figura 38 – Ambiente de teste selecionado a ser executado novamente.



Fonte – Própria autora.

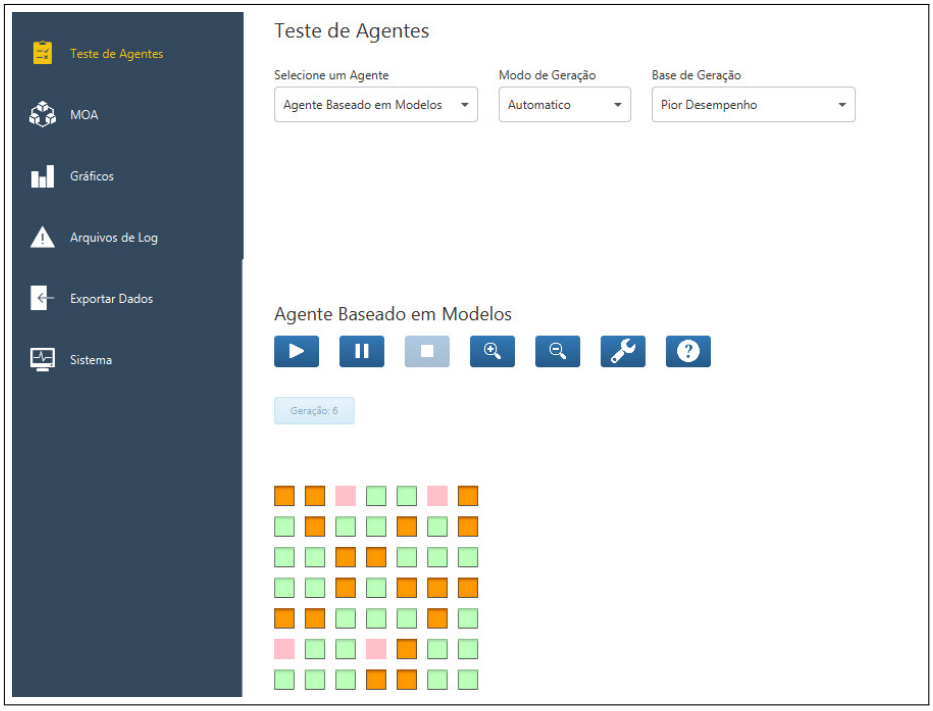
normal para este tipo de agente.

5.4.2 Agente Reativo Baseado em Modelos

A Figura 39 ilustra o teste de um Agent do tipo reativo simples baseado em modelos em que o ambiente selecionado foi o que possuia o pior desempenho para o tipo de Agent, sendo o caso de teste: 8495 o selecionado. O Projetista selecionou este modo com o objetivo de verificar se também ocorrerá a falha anterior com o mesmo tipo de Agent, porém com um outro programa agente.

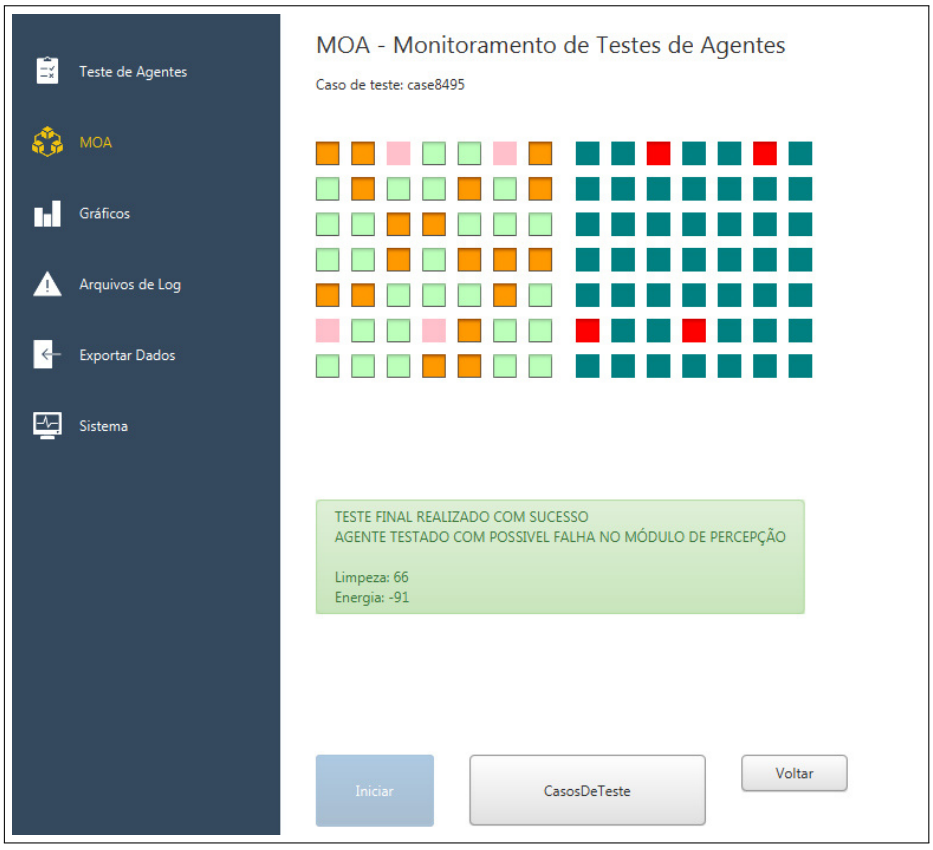
Na Figura 40, pode-se perceber que o MOA realizou o diagnóstico e sugeriu uma possível falha no módulo de percepção, já que Agent visitou quatro nodos com sujeira e não aspirou. Agent gastou 66 pontos de limpeza e 91 de energia para executar a ação.

Figura 39 – Ambiente de teste a ser executado com outro Agent de mesmo tipo.



Fonte – Própria autora.

Figura 40 – Falha exposta ao Projetista.



Fonte – Própria autora.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo serão apresentadas as principais contribuições, trabalhos resultantes, limitações encontradas e sugestões de trabalhos futuros.

6.1 CONTRIBUIÇÕES DO TRABALHO

Considerando que testar agentes é uma atividade complexa, já que os agentes são entidades autônomas, capazes de atingir seus objetivos de maneira independente. A realização de testes adequados favorecem uma melhor avaliação do desempenho do agente na execução das ações e dos planos necessários para atingir seus objetivos em seu ambiente de tarefa. O trabalho é uma extensão e aprimoramento da abordagem desenvolvida por Silveira (2013), inserindo técnicas de otimização interativa para realizar a seleção de conjuntos de casos de testes que sejam úteis para o Projetista.

Para isto, foi criado um método que distribui aleatoriamente a sujeira, porém o Projetista tem a liberdade, mesmo que no modo aleatório, personalizar a sujeira, alterando o estado da sala clicando na sala que o mesmo deseja alterar.

Vale resaltar que em todos os modos de geração de casos de teste o Projetista poderá parar a execução do teste e alterar os parâmetros de sujeira de Amb. A seguir serão apresentadas as vantagens e desvantagens desta abordagem:

Vantagens:

a) É possível desfazer as configuração de ambiente já definidas;
b) O processo de teste fica dinâmico;
c) Gera mais possibilidades para a identificação de falhas.

Desvantagens:

a) Necessita de esforço cognitivo por parte do Projetista na hora de expressar as suas preferências;
b) Requer um conhecimento prévio sobre o sistema agente a ser testado;
c) Só é possível alterar o estado dos nodos que ainda não foram visitadas.

O Projetista pode parar a execução do teste, requerer as modificações, alterar os estados das salas que o mesmo deseja e continuar a execução dos testes. Com isto, por exemplo, o Projetista pode confirmar uma falha gerando uma situação parecida com a anteriormente executada e em que o ambiente gerou uma falha, sem que seja necessário a excução de um novo caso de teste.

Embora tenha sido gerado resultado para o mundo do aspirador de pó, pode-se afirmar que a arquitetura proposta é genérica o suficiente para ser aplicada em diversos outros ambientes com padrões ambientais. Visando consolidar as contribuições, lista-se a seguir as principais provenientes da presente pesquisa:

a) **Padrão ambiental:** considerando um conjunto de características de ambientes de grafos valorados da abordagem de Franco, Campos e Sichman (2016), que possibilita a aplicação em diversos tipos de situação;
b) **Otimização Interativa:** para uma análise empírica mais profunda fez-se necessário a possibilidade do Projetista interagir tanto no início, quanto durante e após a execução dos testes, levando em consideração não apenas a medida de avaliação, mas também a relação entre os componentes internos de Agent e o ambiente de tarefas em que o mesmo está inserido;
c) **Monitoramento:** com o agente MOA é possível designar qual o possível módulo que está gerando falhas.

6.1.1 Publicações Resultantes

1. "ProMon: Agente de Diagnóstico e Monitoramento de Falhas Para Agentes Racionais."Escrito com: Gustavo Augusto Lima de Campos e Francisca Raquel de Vasconcelos Silveira. Apresentado no IX Encontro Unificado de Computação em Teresina em 2016. Prêmio de 1º melhor trabalho.
2. "ProMon: Agente de Diagnóstico e Monitoramento de Falhas Para Agentes Racionais."Escrito com: Gustavo Augusto Lima de Campos e Francisca Raquel de Vasconcelos Silveira. Publicado na REVISTA BRASILEIRA DE COMPUTAÇÃO APLICADA, v. 9, p. 84-96, 2017. Devido ao prêmio do IX Encontro Unificado de Computação em Teresina.

6.2 LIMITAÇÕES

Acredita-se que todos os objetivos delineados por esta pesquisa foram devidamente atingidos ao longo de sua elaboração. Entretanto, algumas limitações podem ser constatadas. São elas:

a) Ausência de outras pesquisas que tenham uma abordagem interativa para testar agentes racionais, impossibilitando a comparação a fim de obter resultados mais maduros;

b) A ausência de experimentos com ambientes reais que possibilite uma análise mais ampla das ações dos agentes testados;
c) A falta de implementação de técnicas de aprendizado de máquina que possam contribuir para a adequação da geração de ambientes para Agent.

6.3 TRABALHOS FUTUROS

O aprimoramento e a continuidade deste trabalho, incluindo as próprias limitações já apresentadas, constituem as oportunidades para trabalhos futuros, que incluem:

a) Desenvolver um campo, onde seja possível visualizar a valor da medida de avaliação de desempenho em tempo de execução, apresentando os comparativos entre os ambientes;
b) Implementar técnicas de aprendizagem de máquina no agente GERADOR para que seja possível a geração de ambientes, não somente por meio de dados salvos, mas utilizando este artifício para melhorar a geração automática;
c) Realizar testes em ambientes reais gerando mais resultados para amadurecimento dos resultados.

REFERÊNCIAS

AMANDI, A. **Programação de agentes orientada a objetos.** 1997. 168f. Tese (Doutorado em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 1997.

ATHAMENA, B.; HOUHAMDI, Z. Structured acceptance test suite generation process for multi-agent system. **Computer and Information Science, Canadian Center of Science and Education,** v. 5, n. 1, p. 55, 2012.

BARTIÉ, A. **Garantia da qualidade de software.** [S.l.]: Gulf Professional Publishing, 2002.

BASTOS, A.; RIOS, E.; CRISTALLI, R.; MOREIRA, T. **Base de conhecimento em teste de software**. São Paulo: [s.n.], 2007.

CARTAXO, E. G. **Geração de casos de teste funcional para aplicações de celulares.** 2006. 82f. Tese (Doutorado em Computação) –Universidade Federal de Campina Grande, Campina Grande, 2006.

FERREIRA, J. d. M. **Uma Investigação em Otimização Interativa Multiusuário para Desenho de Grafos.** 2006. 165f. Tese (Doutorado em Computação) - Universidade Federal de Goiás, Goiânia, 2006.

FERREIRA, T. d. N. **Abordagens interativas usando algoritmo de otimização por colônia de formiga para o problema do próximo release**. 2015. 165f. Dissertação (Mestrado em Computação) — Universidade Estadual do Ceará, Fortaleza, 2015.

FRANCO, M. R. **Avaliação organizacional de times de agentes para o Multi-Agent Programming Contest.** 2014. 175f. Tese (Doutorado em engenharia da Computação) - Universidade de São Paulo, São Paulo, 2014.

FRANCO, M. R.; CAMPOS, G. A.; SICHMAN, J. S. An empirical approach for relating environmental patterns with agent team compositions. In: _____. **Institutions, and Norms in Agent Systems XII**. [S.l.]: Springer, 2016. p. 98–115.

GONÇALVES, E. **Modelagem de arquiteturas internas de agentes de software utilizando a linguagem MAS-ML 2.0**. 2009. 123f. Dissertação (Mestrado em Computação) - Centro de Ciência e Tecnologia, Universidade Estadual do Ceará, Fortaleza, 2009.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. **Search based software engineering:** A comprehensive analysis and review of trends techniques and applications.London: King's College London, 2009.

HEWITT, C. Viewing control structures as patterns of passing messages. **Artificial intelligence,** v. 8, n. 3, p. 323–364, 1977.

HOUHAMDI, Z. Multi-agent system testing: A survey**. International Journal of Advanced Computer,** v. 2, 2011.

INOUE, T.; FURUHASHI, T.; FUJII, M.; MAEDA, H.; TAKABA, M. Development of nurse scheduling support system using interactive ea. **Systems, Man, and Cybernetics,** v. 1, 1999.

IEEE SMC '99 Conference Proceedings. **IEEE International Conference on,** v. 5, p. 533–537 1999.

JORGENSEN, P. C. **Software testing:** a craftsman's approach. [S.l.]: CRC press, 2016. 81p.

JUNIOR, H. d. S. C.; FILHO, L. R. V. M.; ARAÚJO, M. A. **Uma ferramenta interativa para visualização de código fonte no apoio à construção de casos de teste de unidade**. [S.l.]: SAST, 2015. 35p.

LAM, D. N.; BARBER, K. S. Debugging agent behavior in an implemented agent system. In: _____. **Programming Multi-Agent Systems**. [S.l.]: Springer, 2004. p. 104–125.

MAGEDANZ, T.; ROTHERMEL, K.; KRAUSE, S. Intelligent agents: An emerging technology for next generation telecommunications In: FIFTEENTH ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER SOCIETIES, 96.,. 1996. [S.l.], **Anais...** [S.l.]: Networking the Next, 1996. v. 2, p. 464–472.

MARCULESCU, B.; FELDT, R.; TORKAR, R.; POULDING, S. An initial industrial evaluation of interactive search-based testing for embedded software. **Applied Soft Computing,** v. 29, p. 26–39, 2015.

MIETTINEN, K. **Nonlinear Multiobjective Optimization, volume 12 of International Series in Operations Research and Management Science**. [S.l.]: Kluwer Academic Publishers, 1999.

MYLOPOULOS, J.; CASTRO, J. Tropos: A framework for requirements-driven software development. **Information Systems Engineering: State of the Art and Research Themes, Lecture Notes in Computer Science**, v. 2, 2000.

NASCIMENTO, H. A. D. d. **User Hints for Optimization Processes**. 2003. 154f. Tese (Doutorado em Computação) — University of Sydney, Sydney, 2003.

NGUYEN, C. D. **Testing techniques for software agents.** 2008. 165f. Tese (Doutorado em Computação) - DIT University, 2008.

NGUYEN, C. D.; MILES, S.; PERINI, A.; TONELLA, P.; HARMAN, M.; LUCK, M. Evolutionary testing of autonomous software agents. **Autonomous Agents and Multi-Agent Systems,** v. 25, n. 2, p. 260–283, 2012.

NGUYEN, C. D.; PERINI, A.; BERNON, C.; PAVÓN, J.; THANGARAJAH, J. Testing in multi-agent systems. In: INTERNATIONALWORKSHOP ON AGENT-ORIENTED SOFTWARE ENGINEERING, 1., 2009. [S.l.], **Anais...** [S.l.:s.n.], 2009. p. 180–190.

NGUYEN, C. D.; PERINI, A.; TONELLA, P. Goal-oriented testing for mass. **International Journal of Agent-Oriented Software Engineering, Inderscience Publishers,** v. 4, n. 1, p. 79–109, 2009.

NGUYEN, C. D.; PERINI, A.; TONELLA, P.; MILES, S.; HARMAN, M.; LUCK, M. **Evolutionary testing of autonomous software agents**. [S.l.:s.n.], 2009.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial**. 3. ed. [S.l.]: Elsevier Brasil, 2014. v. 1.

PARMEE, I.; HALL, A.; MILES, J.; NOYES, J.; SIMONS, C. et al. Discovery in design: Developing a people-centred computational approach. In: INTERNATIONAL DESIGN CONFERENCE, 9., 2006. Dubrovnik, **Anais...** Dubrovnik:[s.n.], 2006.

RIOS, E. **Teste de software**. [S.l.]: Alta Books, 2006.

RUSSELL, S. J.; NORVIG, P. **Inteligencia Artificial:** un enfoque moderno. [S.l.: s.n.], 2004.

SILVEIRA, F. R. d. V.; CAMPOS, G. A. L. de; CORTÉS, M. Monitoring and diagnosis of faults in tests of rational agents based on condition-action rules. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 17., 2015. [S.l.]. **Anais...** [S.l.: s.n.], 2015. p. 585–592.

SILVEIRA, F. R. V. **Uma abordagem fundamentada em agentes racionais para o teste de agentes racionais**. 2013. 123f. Dissertação (Mestrado em Computação) - Centro de Ciência e Tecnologia, Universidade Estadual do Ceará, Fortaleza, 2013.

SIMONS, C. L.; SMITH, J.; WHITE, P. Interactive ant colony optimization (iaco) for earlylifecycle software design. **Swarm Intelligence,** v. 8, n. 2, p. 139–157, 2014.

TAKAGI, H. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. **Proceedings of the IEEE, IEEE,** v. 89, n. 9, p. 1275–1296, 2001.

TECUCI, G. **Building intelligent agents:** an apprenticeship multistrategy learning theory, methodology, tool and case studies. [S.l.]: Morgan Kaufmann, 1998.

TONELLA, P.; SUSI, A.; PALMA, F. Using interactive ga for requirements prioritization. In: SECOND INTERNATIONAL SYMPOSIUM, 5., 2010. [S.l.], **Anais...** [S.l.:s.n.], 2010. p. 57–66.

WATKINS, J. **Testing it:** an off-the-shelf software testing process. [S.l.]: Cambridge University Press, 2000.

WEISS, G. **Multiagent systems:** a modern approach to distributed artificial intelligence. [S.l.]: MIT press, 1999.

WOOLDRIDGE, M. **An introduction to multiagent systems.** [S.l.]: John wiley & sons ltd., 2002.

XIMENES, J. N. d. S.; CAMPOS, G. A. L.; SILVEIRA, F. R. d. V. Promon: agente de diagnóstico e monitoramento de falhas para agentes racionais. **Revista Brasileira de Computação Aplicada,** v. 9, n. 1, p. 84–96, 2017.

ZINA, H. Test suite generation process for agent testing. **Indian Journal of Computer Science and Engineering**, Citeseer, v. 2, n. 2, 2011.

GLOSSÁRIO

A

Agente: Entidade computacional capaz de perceber o ambiente através de sensores e agir através de atuadores, adaptando-se a mudanças e sendo capaz de criar e perseguir metas.

aprendizagem: Capacidade de aprender e adaptar-se de acordo com o ambiente em que os agentes interagem.

Atuadores: Dispositivos que produzem movimentos/ações.

D

Desempenho: Determina o nível de sucesso do agente, refletindo o resultado desejado no ambiente.

G

GERADOR: Agente gerador de casos de teste.

Grafos: Estrutura matemática que equivale a um conjunto de objetos em que alguns pares estão relacionado em certo sentido.

M

MOA: Agente responsável por receber um caso de teste gerado pelo Projetista e/ou por GERADOR, simular e monitorar as interações de Agent em Amb, visando descobrir os módulos de processamento da informação de Agent que estão causando falhas em Amb, entre outras finalidades.

R

Racionalidade: Que se encontra em conformidade com a razão, ou seja, compreensível logicamente.

S

Sensores: Dispositivos que respondem a um estímulo de maneira específica e que pode ser transformado em ação.

Subsistemas: Módulos de processamento de informação componentes.

U

Utilidade: Função que mapeia um estado para um número real que representa o grau de satisfação com este estado, podendo determinar o peso adequado para cada objetivo.