UNIVERSIDADE FEDERAL FLUMINENSE

MARCELO MARQUES DA ROCHA

EvaML e EvaSIM: Proposta de Linguagem Baseada em XML e Simulador para o Robô EVA

NITERÓI

2022

MARCELO MARQUES DA ROCHA

EvaML e EvaSIM: Proposta de Linguagem Baseada em XML e Simulador para o Robô EVA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: CIÊNCIA DA COMPUTAÇÃO

Orientadora:
DÉBORA CHRISTINA MUCHALUAT SAADE

NITERÓI

2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

R672e Rocha, Marcelo Marques da
EvaML e EvaSIM: Proposta de Linguagem Baseada em XML e Simulador para o Robô EVA / Marcelo Marques da Rocha ; Débora Christina Muchaluat Saade, orientador. Niterói, 2022.
168 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2022.

DOI: http://dx.doi.org/10.22409/PGC.2022.m.07166477709

1. Robótica. 2. Sistemas Multimídia. 3. Linguagem de programação (Computador). 4. Simulador. 5. Produção intelectual. I. Muchaluat Saade, Débora Christina, orientador. II. Universidade Federal Fluminense. Instituto de Computação. III. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

MARCELO MARQUES DA ROCHA

EvaML e EvaSIM: Proposta de Linguagem
Baseada em XML e Simulador para o Robô EVA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: CIÊNCIA DA COMPUTAÇÃO

Aprovada em setembro de 2022.

BANCA EXAMINADORA

Profa. DÉBORA CHRISTINA MUCHALUAT SAADE - Orientadora, UFF

Profa. AURA CONCI, UFF

Profa. CLAUDIA MARIA LIMA WERNER, UFRJ

Niterói
2022

Todas as coisas foram feitas por Ele, e sem Ele nada do que foi feito se fez.

João 1:3

Agradecimentos

Gostaria de agradecer ao Deus de Abraão, de Isaque e de Jacó. Ao Deus da minha salvação, minha força, meu refúgio e minha inspiração. O Deus que esteve comigo por tantas madrugadas, me conduzindo. Esse Deus que, por tantas vezes, me fez compreender o que eu não entendia e me fez enxergar aquilo que eu não conseguia ver. Obrigado Deus por ter permitido que eu chegasse até aqui!

Quero agradecer à minha esposa, Rachel Rocha, por todo amor, carinho e compreensão durante essa jornada. Agradeço e peço desculpas à minha querida filha, Luiza Rocha, por tantas vezes ter ficado ausente devido ao tempo dedicado a este trabalho. Espero que um dia, você possa compreender que tudo que fiz, foi por você!

Agradeço aos meus pais, Luis e Vera, por sempre terem feito de tudo para que eu pudesse alcançar meus objetivos. Sou grato ao meu irmão Márcio Rocha, por ter me mostrado, desde cedo, a beleza e a importância da matemática.

Gostaria de fazer um agradecimento especial à minha prezada orientadora Débora Saade! Que me direcionou e conduziu no desenvolvimento deste trabalho com maestria e generosidade. Obrigado pela paciência, por todo aprendizado e por todas as oportunidades que você me proporcionou. Vamos em frente!!!

Agradeço aos amigos que fiz durante o mestrado e que tantas vezes me ajudaram, Eyre Montevecchi, Marina Josué, Fábio Barreto, Raphael Abreu e Pedro Valentim. Aos meus companheiros de viagem, Rômulo Vieira e Flávio Miranda. E ao meu amigo e dupla em tantos trabalhos de disciplinas, Augusto Parisot.

Gostaria de agradecer também à CAPES e ao INCT-MACC pelo apoio financeiro parcial deste trabalho.

Resumo

As tecnologias robóticas não têm sido usadas somente nas fábricas, mas em vários ambientes no dia a dia das pessoas. Os robôs são usados como dispositivos assistivos e a robótica é um método capaz de ampliar as capacidades físicas e cognitivas dos humanos. Uma nova classe de robôs, os robôs socialmente assistivos (SARs - *Socially Assistive Robots*) surge da interseção de duas outras classes, a classe dos robôs assistivos, que prestam algum tipo de assistência, e a classe dos robôs sociais interativos, que se comunicam com o usuário. O objetivo dos SARs não é apenas fornecer algum tipo de assistência ou se comunicar, mas fornecer estímulos ao usuário por meio da interação com o robô. Esta dissertação de mestrado propõe a extensão da linguagem de programação visual de um SAR chamado EVA com a adição de dois novos componentes para o aprimoramento da capacidade de interação multimodal do robô, oferecendo reconhecimento de expressões faciais dos usuários e integração de efeitos sensoriais de luz. Além disso, esta dissertação propõe e avalia um jogo sério para crianças com TEA (Transtorno do Espectro Autista) utilizando o robô EVA com as novas funcionalidades implementadas neste trabalho. Esta dissertação propõe também a linguagem EvaML, uma linguagem específica de domínio (DSL – *Domain Specific Language*) baseada em XML para facilitar a especificação de sessões interativas com o robô EVA. Esta dissertação apresenta também a proposta de um simulador para o robô EVA, o EvaSIM, capaz de executar suas sessões interativas e auxiliar no desenvolvimento de programas para o robô. O paradigma GQM (*Goal Question Metric*) foi usado para estruturar a avaliação das soluções desenvolvidas nesta dissertação. Os novos componentes que estenderam a capacidade de interação multimodal do robô e o jogo sério proposto foram analisados em relação a aceitação da tecnologia. A linguagem EvaML foi analisada em relação à sua clareza, eficácia, facilidade de uso percebida e teve sua usabilidade avaliada segundo nove dimensões cognitivas do framework CDN (*Cognitive Dimensions of Notations*). Todos os objetivos foram alcançados ao final da etapa de avaliação, observando-se bons resultados. Também foram observados bons resultados na avaliação da usabilidade da linguagem EvaML segundo os critérios definidos pelo framework CDN. Além disso, os usuários avaliaram positivamente a linguagem com afirmações relacionadas à facilidade de uso. Para o simulador EvaSIM, também foi

realizada uma avaliação estruturada segundo o paradigma GQM. O simulador foi analisado em relação à sua utilidade percebida, facilidade de uso percebida, clareza e teve sua usabilidade avaliada através das questões do SUS (*System Usability Scale*), do ponto de vista dos usuários. Todos os objetivos foram alcançados ao final da etapa de avaliação, observando-se bons resultados. Utilizando-se a escala de aceitabilidade relativa à pontuação SUS, o EvaSIM teve sua usabilidade geral, classificada como *aceitável* e, utilizando-se uma escala de adjetivos, o EvaSIM foi classificado como *"melhor imaginável"*.

Palavras-chave: Robôs socialmente assistivos, Interação multimodal, Jogo sério, Linguagem específica de domínio, Simulador.

Abstract

Robotic technologies have not only been used in factories, but in various environments in people's daily lives. Robots are used as assistive devices and robotics is a method capable of expanding the physical and cognitive capabilities of humans. A new class of robots, named Socially Assistive Robots (SARs), arises from the intersection of two other classes, the class of assistive robots, which provide some kind of assistance, and the class of interactive social robots, which communicate with the user. The purpose of SARs is not just to provide some kind of assistance or to communicate, but to provide stimuli to the user through interaction with the robot. This master dissertation proposes the extension of the visual programming language of an SAR called EVA with the addition of two new components to improve the robot's multimodal interaction capability, offering recognition of users' facial expressions and integration of light sensory effects. In addition, this dissertation proposes and evaluates a serious game for children with ASD (Autism Spectrum Disorder) using the EVA robot with new features implemented in this work. This dissertation also proposes the EvaML language, a domain specific language (DSL) based on XML to facilitate the specification of interactive sessions with the EVA robot. In addition, this dissertation also presents the proposal of a simulator for the EVA robot, the EvaSIM, capable of executing its interactive sessions and assisting in the development of programs for the robot. The GQM (Goal Question Metric) paradigm was used to structure the evaluation of the solutions developed in this dissertation. The new components that extended the robot's multimodal interaction capability and the proposed serious game were analyzed in relation to the acceptance of the technology. The EvaML language was analyzed regarding its clarity, effectiveness, perceived ease of use and its usability was evaluated according to nine cognitive dimensions of the CDN framework (Cognitive Dimensions of Notations). All goals were achieved at the end of the evaluation stage, and good results were observed. Promising results were also observed in the evaluation of the usability of the EvaML language according to the criteria defined by the CDN framework. In addition, users positively rated the language with statements related to ease of use. For the EvaSIM simulator, an evaluation structured according to the GQM paradigm was also performed. The simulator was analyzed in relation to its perceived usefulness,

perceived ease of use, clarity and its usability was evaluated through the SUS (System Usability Scale) questions, from the users' point of view. All goals were achieved at the end of the evaluation stage, and good results were observed. Using the acceptability scale relative to the SUS score, EvaSIM had its general usability classified as *acceptable* and, using an adjective scale, EvaSIM was classified as *"best imaginable"*.

Keywords: Socially Assistive Robots, Multimodal Interaction, Serious Game, Domain-Specific Language, Simulator.

Lista de Figuras

1 Componentes de hardware do robô EVA 27

2 Uma sessão de terapia de estimulação cognitiva com três participantes sendo conduzida pelo robô Eva (CRUZ-SANDOVAL; MORALES-TELLEZ et al., 2020) 28

3 Arquitetura de software do robô EVA 29

4 Um script na VPL 30

5 Expressões sintetizadas pelo robô EVA 31

6 Controlando efeitos sensoriais de luz 34

7 Reconhecimento de expressões faciais 35

8 Fluxograma do estágio 1 do jogo 37

9 Fluxograma do estágio 2 do jogo 37

10 Fluxograma do estágio 3 do jogo 38

11 Criança jogando o jogo de imitação (Versão incompleta das peças do robô) 39

12 Criança jogando o jogo de imitação (Versão completa do robô) 40

13 Respostas do grupo dos especialistas paras as questões relacionadas à Utilidade Percebida (PU) e Facilidade de Uso Percebida (PEOU) 43

14 Respostas do grupo dos iniciantes paras as questões relacionadas à Utilidade Percebida (PU) e Facilidade de Uso Percebida (PEOU) 43

15 Seções de um documento EvaML 47

16 Árvore de elementos de um documento EvaML 47

17 The EVA Robot Memory Array View 52

18 Processo de *parsing* e exemplo de linha de comando 56

19 Processo de validação de um documento EvaML usando XML Schema . . . 58

20 Elementos da interface gráfica de usuário do EvaSIM 61

21 Simulando o reconhecimento de expressões faciais 63

22 Simulando o reconhecimento de voz 63

23 Indicando o movimento da cabeça do robô 64

24 Emulador de terminal 64

25 Arquitetura geral do EvaSIM 65

26 Um exemplo de script VPL 66

27 (a) Estrutura de um script VPL vazio. (b) Um exemplo de um nó do tipo *Condition*. (c) Um exemplo de uma *aresta* (link) 67

28 Processamento e Simulação de um script da VPL no EvaSIM 68

29 Estrutura hierárquica do modelo GQM desenvolvido nesta dissertação . . . 79

30 Tempo de duração das etapas 1, 2 e 3 da atividade com a EvaML 86

31 Nível de dificuldade dos comandos da EvaML do ponto de vista dos experientes em XML 89

32 Nível de dificuldade dos comandos da EvaML do ponto de vista dos iniciantes em XML 90

33 Respostas do grupo dos experientes em XML para as questões relacionadas às dimensões cognitivas (CDN) 92

34 Respostas do grupo dos iniciantes para as questões relacionadas às dimensões cognitivas (CDN) 92

35 Respostas dos experientes com o EVA para as questões Q9 a Q12 99

36 Respostas dos iniciantes com o EVA para as questões Q9 a Q12 99

37 Respostas dos experientes com o EVA para o questionário SUS 101

38 Respostas dos iniciantes com o EVA para o questionário SUS 102

39 Pontuação SUS - Resultados para os dois grupos e para o todo 103

40 Escalas Adjetiva e de Aceitabilidade associadas à pontuação SUS 104

41 Modelos para a impressão 3D 132

42 (a) Visão frontal da placa Matrix Voice com os 18 LEDs RGB, (b) Montagem do display de toque de 5.5" e (c) Matrix Voice acoplada ao Raspberry PI 133

43 Placa ArbotiX-M, servomotor AX-12A e um cabo FTDI-USB 134

44 Fixação dos servomotores no suporte na parte interna do Eva 134

45 Monitor de porta serial da IDE do Arduino 136

46 Saída do programa que mostra o ID de um servomotor na janela de monitoramento da porta serial 136

47 Robô EVA 140

48 Instalando o plugin XML no VSCode 141

49 Instalando o Python no Windows 142

Lista de Tabelas

1 Questionário TAM 41

2 Questionário TAM - Resultados 42

3 Elementos de um documento EvaML (elemento raiz e elementos principais) 48

4 Elementos de um documento EvaML (comandos) 50

5 Vozes para o serviço de texto para fala do IBM Watson 51

6 Tipos de comparação e operações lógicas do comando <**case**> 56

7 Parâmetros do *Parser* EvaML 57

8 Comandos da VPL simulados pelo EvaSIM 65

9 Objetivos determinados para os experimentos realizados com a linguagem EvaML e o simulador EvaSIM. 73

10 Questões do objetivo G1. 75

11 Questões relacionadas às Dimensões Cognitivas (CDN) do objetivo G1 . . 76

12 Métricas para questões do G1. 77

13 Questões do objetivo G2. 77

14 Métricas para as questões do G2. 78

15 Questionário SUS para o objetivo G2 79

16 Questão Q1 (Clareza) para o objetivo G1 - Resultados 84

17 Questões Q5 a Q7 para o objetivo G1 - Resultados 84

18 Questão Q8 para o objetivo G1 - Resultados 87

19 Questões relacionadas às Dimensões Cognitivas para o objetivo G1 - Resultados 91

20 Linguagem EvaML (notas finais) - Resultados 96

21 Questões Q9 a Q12 para o objetivo G2 - Resultados 98

22 Questões do SUS (Q13 a Q22) para o objetivo G2 - Resultados 103

23 Pontuação SUS - Resultados 104

Lista de Abreviaturas e Siglas

API *Application Programming Interface*

ARM *Advanced RISC Machine*

CDN *Cognitive Dimensions of Notations*

DOF *Degree of Freedom*

DOM *Document Object Model*

DSL *Domain Specific Language*

DTD *Document Type Definition*

DTT *Discrete Trial Training*

EVA *Embodied Voice Assistant*

GPL *General Purpose Language*

GQM *Goal Question Metric*

GUI *Graphical User Interface*

IBM *International Business Machines*

IDE *Integrated Development Environment*

IHR Interação Humano-Robô

IPC *Inter-Process Communication*

JSON *JavaScript Object Notation*

LED *Light Emitting Diode*

PEOU *Perceived Ease of Use*

PU *Perceived Usefulness*

RGB *Red, Green e Blue*

SAR *Socially Assistive Robots*

STT *Speech to Text*

SUS *System Usability Scale*

TAM *Technology Acceptance Model*

TCP *Transmission Control Protocol*

TEA Transtorno do Espectro Autista

TTS *Text to Speech*

VPL *Visual Programming Language*

XML *Extensible Markup Language*

Sumário

1 Introdução **12**

1.1 Objetivos e Contribuições 15

1.2 Estrutura da Dissertação 17

2 Trabalhos Relacionados **18**

2.1 Robôs Socialmente Assistivos para Terapias de TEA 18

2.2 Efeitos Sensoriais em Terapias em Saúde 19

2.3 Jogos Sérios Aplicados em Terapias para Autistas 20

2.4 Linguagens para Robôs 21

2.5 Simuladores de Robôs 23

3 Robô EVA **26**

3.1 Arquitetura do Robô 26

3.1.1 Componentes de Hardware 26

3.1.2 Componentes de Software 28

3.2 Funcionalidades Principais do Robô EVA 30

3.2.1 Expressões do Olhar Usando o Display 30

3.2.1.1 Reconhecimento de Voz 31

3.2.2 Fala 31

3.2.3 Execução de Arquivos de Áudio 31

3.2.4 Animação dos LEDs RGB 32

3.2.5 Movimentação da Cabeça 32

4 Proposta de Jogo Sério para Crianças com TEA **33**

4.1 Estendendo as Capacidades de Interação Multimodal do EVA 33

4.1.1 Controle de Efeitos Sensoriais de Luz 34

4.1.2 Reconhecimento de Expressões Faciais 34

4.1.3 Um Jogo Sério para Crianças Autistas em 3 Estágios 35

4.2 Avaliação da Proposta 39

4.3 Considerações Finais 44

5 Linguagem EvaML **45**

5.1 Elementos de um Documento EvaML 46

5.2 Comandos da Linguagem EvaML 50

5.2.1 Processo de Parsing de um Script EvaML 55

5.2.2 Processo de Validação de um Script EvaML 57

6 Simulador EvaSIM **60**

6.1 Simulando o Reconhecimento de Expressões Faciais 62

6.2 Simulando o Reconhecimento de Voz 62

6.3 Simulando a Movimentação da Cabeça do Robô 63

6.4 Executando arquivos JSON 64

6.4.1 O Funcionamento do EvaSIM 65

6.4.2 Mapeamento de dados JSON *(JSON Data Mapping)* 67

6.4.3 Processamento de Código XML EvaSIM *(EvaSIM XML Code Processing)* 70

7 Avaliação **72**

7.1 Metodologia 72

7.1.1 G1 - Questões e Métricas 73

7.1.2 G2 - Questões e Métricas 75

7.1.3 Modelo GQM 77
7.2 Experimentos com Usuários 77
7.2.1 Experimento com a Linguagem EvaML 78
7.2.2 Experimento com o Simulador EvaSIM 81
7.3 Resultados 83
7.3.1 G1 - Análise dos Resultados 83
7.3.2 G2 - Análise dos Resultados 96
7.4 Considerações Finais 104

8 Conclusão 106
8.1 Limitações 108
8.2 Trabalhos Futuros 109

REFERÊNCIAS 111

Apêndice A – XML SCHEMA DA LINGUAGEM EVAML 120

Apêndice B – MONTAGEM FÍSICA DO ROBÔ EVA 132
B.1 Montagem do Robô 132
B.2 Configurando os IDs dos Servomotores AX-12A 135

Apêndice C – ROTEIRO DO TESTE DE AVALIAÇÃO DA EVAML E DO EVASIM 139

Apêndice D – QUESTIONÁRIO: EVAML E EVASIM - TESTE DE USABILIDADE 148

1 Introdução

As tecnologias robóticas não têm sido usadas somente nas fábricas, mas em vários ambientes no dia a dia das pessoas. A interação homem-robô vem sendo empregada para melhorar a qualidade de vida e está se tornando cada vez mais comum (SHIBATA, 2012). Robôs são usados como dispositivos assistivos do dia a dia das pessoas, e a robótica é um método capaz de ampliar as capacidades físicas e cognitivas dos humanos. Existem robôs, por exemplo, que auxiliam nos trabalhos domésticos, há aqueles que assistem pessoas com algum tipo de deficiência nos braços, ajudando-as a se alimentarem. Contudo, têm surgido avanços em um novo campo da robótica, onde o objetivo não é prover assistência, mas prover estímulos aos humanos, através da interação com o robô (SHIBATA, 2004).

Uma nova classe de robôs, os robôs socialmente assistivos (SARs - *Socially Assistive Robots*) surge da interseção de duas outras classes, a classe dos robôs assistivos, que prestam algum tipo de assistência, e a classe dos robôs sociais interativos, que se comunicam com o usuário. O objetivo dos SARs não é apenas fornecer algum tipo de assistência ou se comunicar, mas fornecer estímulos ao usuário por meio da interação com o robô. Espera-se que os robôs assistivos se tornem onipresentes, transformando a vida cotidiana, e sejam amplamente utilizados em terapias de saúde. O aumento da prevalência de pessoas idosas com demência, associado à maior demanda por serviços de saúde, representa um desafio para a sociedade. O uso de tecnologias inovadoras, como a robótica, pode vir a ser uma das soluções para essa situação (NESTOROV et al., 2014). O projeto de SARs é um campo multidisciplinar que precisa incorporar as diversas necessidades de pacientes e terapeutas para serem empregados em terapias em saúde.

Projetado especificamente para fins terapêuticos, o robô PARO (*Seal Robot*) atende ao crescimento percebido das necessidades dos idosos, incluindo aqueles que necessitam de cuidados a longo prazo, de receber um estímulo social e emocional de maneira significativa e positiva. Ao mesmo tempo, o uso do robô dá mais liberdade aos cuidadores para a execução de outras tarefas (BISCHOFF; GRAEFE, 2002).

Em (CRUZ-SANDOVAL; FAVELA, 2019a), um estudo com pacientes com doença de Alzheimer foi realizado e mostrou que o SAR chamado EVA foi eficaz em envolver os participantes da terapia. Estudos mostram que o uso de tecnologias como os SARs pode enriquecer as intervenções de saúde, facilitar a comunicação entre terapeutas e pacientes e apoiar a coleta de dados e avaliação no diagnóstico de pacientes (KIENTZ et al., 2013; SANTOS et al., 2021).

Os SARs também vêm sendo utilizados para auxílio no diagnóstico e tratamento de crianças com TEA (Transtorno do Espectro Autista) (SANTATIWONGCHAI et al., 2016). Em (SANTOS et al., 2021) há uma proposta de plataforma de *coaching* robótico para treinamento social, motor e cognitivo. O robô utilizado nesse trabalho é o robô NAO, que tem sido amplamente utilizado na terapia social com crianças com TEA. O NAO é um robô humanoide, isto é, parece um humano, e pode fornecer estímulos sonoros e visuais. Todas essas caraterísticas são favoráveis à interação com crianças com TEA, que tendem a preferir estímulos simplificados para evitar focar em detalhes.

Em (FACHANTIDIS; SYRIOPOULOU-DELLI; ZYGOPOULOU, 2020), são apresentados resultados positivos da interação de crianças com TEA com um robô, indicando uma maior incidência de contato visual, proximidade e interação do que as interações entre crianças e um humano. Em (ROBINS et al., 2010; LEE, Jaeryoung et al., 2012; LEE, J. et al., 2012), há evidências que apoiam a afirmação de que crianças autistas preferem robôs a humanos. Os resultados desses estudos indicam que os robôs podem gerar diversos efeitos positivos nesse tipo de terapia e que a interação humano-robô pode ajudar a solucionar os problemas que ocorrem na interação humano-humano. Os déficits nas habilidades sociais e de comunicação em pacientes com autismo abrangem uma série de outras deficiências, como dificuldade em reconhecer a linguagem corporal ou demonstrar contato visual com quem conversam e problemas em expressar suas próprias emoções e entender as dos outros.

A interação multimodal e os efeitos sensoriais podem ser muito úteis quando somados às capacidades dos robôs. Além da interação por voz, que é frequentemente usada por SARs como o EVA, a interação por vídeo pode ser fornecida para ajudar a capturar a emoção do usuário, por exemplo. Além disso, efeitos sensoriais de luz podem ser usados para criar terapias imersivas e torná-las mais atrativas para os usuários, principalmente crianças (ROCHA; VALENTIM et al., 2022; JOSUÉ et al., 2020).

Os avanços na robótica têm permitido que robôs ganhem grande complexidade e portanto estão sendo utilizados em tarefas cada vez mais desafiadoras. Programar esses robôs

utilizando uma linguagem de propósito geral (GPL - *General Purpose Language*) torna-se uma tarefa árdua. A dificuldade dessa tarefa é devido ao fato de que mesmo simples programas necessitam obter valores vindos dos sensores, executando algum algoritmo de controle e enviando algum comando para os atuadores no robô.

Segundo Mernik, Heering e Sloane (2005), quando é preciso optar entre uma linguagem específica de domínio (DSL - *Domain Specific Language*) ou uma GPL, o que está em jogo é a escolha entre generalidade e expressividade. As DSLs apresentam ganhos de expressividade e facilidade de uso se comparadas com as GPLs. As DSLs também apresentam ganhos em produtividade e custo de manutenção. A utilização de uma DSL demanda do usuário uma menor quantidade de domínio e conhecimento.

Nos últimos anos, dezenas de linguagens de programação foram projetadas e usadas para programar softwares de robôs (YANG et al., 2015). Algumas delas são linguagens de propósito geral, o que significa que elas não são especialmente projetadas para construir softwares de robôs, por exemplo, Java, C++, Python, etc. (CHEN et al., 2005). Esse tipo de linguagem dificulta o entendimento, pois sendo uma linguagem feita para programadores, a sua utilização na construção de programas para robôs por pessoas que não possuem conhecimento profundo em programação de computadores, se torna bastante difícil. Outra dificuldade que se enfrenta, quando se utiliza uma linguagem de propósito geral na programação de robôs, é a falta de elementos que possam acessar os componentes de hardware e recursos específicos de cada robô.

Segundo Tousignant, Van Wyk e Gini (2011) as linguagens específicas de domínio estão começando a ganhar popularidade na comunidade robótica, porque prometem simplificar o processo de desenvolvimento de programas grandes e complexos que são necessários para robôs. Linguagens específicas para robôs são tipos de linguagens de programação dedicadas a resolver questões específicas, isto é, gerenciamento do hardware do robô, controle e planejamento da movimentação do robô, percepção do ambiente, controle de sensores etc. Na verdade, linguagens de programação para robô são diretamente influenciadas pelas características do robô e evoluem junto com o robô e suas aplicações. Uma linguagem que facilite o desenvolvimento de programas para um robô pode facilitar sua divulgação e uso na prática, o que é bastante desejável.

Quando se programa para um robô físico, nem sempre o hardware do robô está disponível para testes dos programas desenvolvidos. Assim sendo, uma ferramenta de simulação do robô é muito útil. Um simulador permite testar algoritmos desenvolvidos para o robô físico, sem a necessidade de ter o dispositivo montado. Os simuladores são uma solução

comum para contornar problemas relacionados a cenários de teste do mundo real (PERICO et al., 2016). Os simuladores de robôs têm sido utilizados como ferramentas no treinamento educacional com evidências de que os resultados obtidos com o uso do robô físico, durante o treinamento, são semelhantes aos do simulador (KURNIAWAN et al., 2018).

1.1 Objetivos e Contribuições

Este trabalho utiliza o robô EVA (*Embodied Voice Assistant*) (CRUZ-SANDOVAL; FAVELA, 2019b) como plataforma de desenvolvimento de SARs. O EVA é *open source* e fornece a maioria dos recursos necessários para construir um SAR totalmente funcional e acessível, além de permitir a criação de interações para contextos e populações específicas usando uma linguagem de programação visual.

Com intuito de oferecer terapias imersivas para crianças autistas, este trabalho aprimora as capacidades do robô EVA de reconhecer as emoções do usuário por meio do reconhecimento de expressões faciais e também de oferecer efeitos sensoriais de luz para tornar a terapia mais atraente para os usuários. A ideia é usar o robô interativo como a primeira etapa de uma terapia de reconhecimento de emoções para crianças com TEA. Esta dissertação propõe uma sessão de terapia de regulação de emoção para TEA, através de um Jogo Sério (JS) onde a criança reconhece as emoções do robô. Durante o jogo, o robô EVA reconhece a expressão facial da criança para verificar seu progresso no aprendizado. Para avaliar a proposta, uma criança neurotípica de 6 anos foi convidada a jogar o jogo, com o consentimento de seus pais, e foram gravados vídeos da sessão do jogo. Esses vídeos foram avaliados por médicos e psicólogos especialistas em terapias para TEA usando o questionário baseado no modelo de aceitação de tecnologia (TAM - Technology Acceptance Model) (MARANGUNIĆ; GRANIĆ, 2015). Foi feita uma análise estatística para comparar os resultados da avaliação entre terapeutas de TEA experientes e iniciantes.

Além da extensão das capacidades de interação multimodal para o robô EVA, este trabalho propõe uma DSL para especificação de sessões interativas para o robô (ROCHA; VALENTIM et al., 2022). A escolha do EVA deve-se ao fato dele ser um projeto de robótica *open-source* criado para servir como ferramenta de apoio a pesquisas na área de IHR (Interação Humano-Robô). A linguagem proposta chama-se EvaML e ela é uma linguagem baseada em XML (*Extensible Markup Language*) (W3C, 2008). Por ser baseada

em XML, ela tem como um dos objetivos principais ser mais legível que as linguagens de propósitos gerais. A EvaML possui elementos específicos que permitem controlar os recursos de hardware e de interação multimodal do robô EVA.

Embora o EVA use hardware de baixo custo e software de código aberto, nem sempre é prático ter um robô físico à mão, principalmente durante o projeto iterativo de terapias. Portanto, é difícil testar uma sessão de terapia robótica se você não tiver o robô físico montado com todos os seus componentes de hardware sendo difícil também treinar pessoas (técnicos ou pesquisadores/entusiastas em IHR), ensinando-os a programar uma aplicativo ou sessão de terapia interativa usando a ferramenta de programação do robô. Com essa motivação, este trabalho propõe uma ferramenta de simulação que permita testar scripts desenvolvidos para o robô EVA, o simulador chamado EvaSIM.

Para avaliar a linguagem EvaML e o simulador EvaSIM, esta dissertação utiliza a metodologia GQM (*Goal Question Metric*) (CALDIERA; ROMBACH, 1994), que define objetivos, questões e métricas para as avaliações a serem feitas. As propostas foram avaliadas com usuários experientes e iniciantes no uso do robô EVA e também com usuários experientes e iniciantes no uso de XML.

Em resumo, esta dissertação tem como objetivos principais:

- proposta de uma linguagem baseada em XML para o desenvolvimento de sessões interativas para o robô EVA, que facilite o desenvolvimento dessas sessões e
- proposta de um simulador para o robô EVA capaz de executar suas sessões interativas e auxiliar no desenvolvimento de programas para o robô.

Tendo como objetivos secundários:

- extensão de capacidades de interação multimodal do robô EVA, com reconhecimento de expressões faciais dos usuários e integração de efeitos sensoriais de luz,
- proposta de um jogo sério para terapias de crianças com TEA,

Destacam-se as seguintes contribuições da dissertação:

- Linguagem EvaML baseada em XML para o desenvolvimento de sessões interativas para o robô EVA.

- Software simulador EvaSIM para auxiliar no desenvolvimento de programas para o robô.
- Extensão da linguagem de programação visual do robô com a adição de dois novos componentes para o aprimoramento da capacidade de interação multimodal do robô, oferecendo reconhecimento de expressões faciais dos usuários e integração de efeitos sensoriais de luz.
- Proposta e avaliação de um jogo sério para crianças com TEA utilizando o robô EVA com as novas funcionalidades implementadas neste trabalho.

1.2 Estrutura da Dissertação

O restante do texto desta dissertação está estruturado da seguinte forma. No Capítulo 2 são discutidas as pesquisas relacionadas a este trabalho encontradas na literatura.

O Capítulo 3 apresenta o robô EVA e seus componentes de hardware e software, descrevendo suas principais funcionalidades.

O Capítulo 4 apresenta em detalhes a proposta de duas novas funcionalidades do robô que foram desenvolvidas durante a implementação de um jogo sério para terapias com crianças TEA.

No Capítulo 5 é apresentada a proposta da linguagem EvaML, baseada em XML, para desenvolvimento de sessões interativas para o robô EVA. A proposta do simulador do robô, chamado EvaSIM, é apresentada no Capítulo 6.

O Capítulo 7 descreve a avaliação realizada, explicando a metodologia utilizada, experimentos realizados e a análise dos resultados obtidos.

O Capítulo 8 conclui o trabalho, apresentando suas limitações e assinalando trabalhos futuros.

2 Trabalhos Relacionados

2.1 Robôs Socialmente Assistivos para Terapias de TEA

SARs tornaram-se ferramentas populares em intervenções com pacientes com TEA. Robôs têm sido usados em escolas de educação especial e centros de atendimento para pessoas autistas. Em (JAVED; PARK, 2019), um framework foi desenvolvido para envolver crianças com TEA em interações sensoriais. Eles usaram dois robôs, o primeiro chamado Romo e o segundo era um pequeno robô humanoide. Eles eram capazes de produzir estímulos visuais e sonoros. O robô Romo pode expressar suas emoções através de três componentes: movimento, expressões faciais e efeitos sonoros. A estrutura foi usada como ferramenta para terapias de regulação emocional e funcionou da seguinte forma. O paciente interagia com o robô por meio de um jogo, que rodava em um notebook. O jogo usa três estados principais, que são o estado de emoção do usuário, o estado de emoção do personagem pinguim e um estado de emoção objetivo predeterminado fixo para o qual o pinguim orienta o usuário. Para construir um relacionamento com o usuário, o pinguim primeiro aloca dinamicamente um estado objetivo temporário que se aproxima do estado do usuário, então ele gradualmente move esse estado objetivo temporário para mais perto do objetivo predeterminado para facilitar a regulação emocional. Assim, com base na interação contínua com o personagem pinguim, o algoritmo tenta guiar o usuário para o estado alvo desejado. O participante é então orientado, progressivamente, pelo comportamento social do robô (movimento/expressão facial/efeitos sonoros) até a expressão facial final. Os resultados de um estudo com usuários confirmaram a viabilidade do framework como uma ferramenta para regular a emoção.

Em (SANTATIWONGCHAI et al., 2016), um robô móvel foi desenvolvido e usado para investigar seu potencial em estimular interações sociais em crianças com autismo. Esses estímulos foram fornecidos por meio de jogos ou atividades terapêuticas. Durante as sessões com o robô, especialistas em autismo observaram algumas capacidades em pacientes que nunca haviam sido vistas em outras terapias, como a capacidade de desenvolver

novas estratégias ao jogar um jogo com o robô. Isso indica que o robô poderia auxiliar esses especialistas a encontrar novas capacidades em crianças com autismo, criando terapias mais eficazes, adaptadas a cada paciente.

Uma proposta de uso de um robô, que é capaz de medir o comportamento social de uma criança com TEA durante uma interação, foi feita em (HIROKAWA et al., 2016). Essa proposta utiliza um dispositivo vestível para detectar o sorriso da criança. Embora o dispositivo permita detectar um sorriso mesmo sem a criança estar de frente para o robô, o uso de um dispositivo vestível pode ser desconfortável, principalmente para crianças com TEA que tendem a ter hipersensibilidade tátil.

Bevill et al. (2016) apresentam um sistema robótico interativo que fornece comportamentos emocionais e sociais para terapia multissensorial para crianças com TEA. O sistema utiliza dois robôs, um dos quais é capaz de mostrar emoções através de expressões faciais e o outro demonstra suas emoções usando linguagem corporal e gestos. A proposta dos autores é que os robôs percorram um cenário labiríntico e durante o circuito encontrem objetos que estimulem os cinco sentidos, ou seja, visão, audição, olfato, paladar e tato. A ideia é que as crianças possam aprender com os robôs a lidar com a sobrecarga sensorial.

Um estudo usando dois robôs foi realizado em (FEIL-SEIFER; MATARIC', 2008). O objetivo era examinar o comportamento dos robôs e se eles afetavam ou não o comportamento de crianças com TEA. Foram encontradas evidências de que o comportamento dos robôs afeta o comportamento social da criança, tanto na interação humano-humano quanto na interação humano-robô.

2.2 Efeitos Sensoriais em Terapias em Saúde

O uso de efeitos sensoriais também tem sido aplicado em terapias de autismo. Josué et al. (2020) apresentam alguns casos de uso de aplicações que utilizam efeitos sensoriais nestas terapias. Um deles é a *Magic Room*, que é um sistema que visa proporcionar integração sensorial para crianças com TEA através do uso de efeitos sensoriais. Outro caso de uso proposto foi a criação de um ambiente multissensorial imersivo onde o terapeuta seria capaz de manipular efeitos sensoriais e o paciente ficaria imerso naquele ambiente. O objetivo da aplicação era permitir a realização de terapias de relaxamento em salas multissensoriais para pacientes com doença de Alzheimer.

Em (PARES et al., 2005), há uma proposta de ambiente físico adaptativo que permite

que crianças com autismo severo interajam com sucesso com estímulos multimodais. Esse ambiente gera estímulos de vários tipos (visuais, auditivos e vibrotáteis) em tempo real.

Zubrycki e Granosik (2016) apresentaram um sistema de dispositivos sensoriais para apoiar terapeutas de crianças com TEA. Uma Caixa de Sensores (*Sensor Box*) podia ser manuseada por crianças e era capaz de gerar uma série de estímulos sensoriais, como sons e efeitos de luz.

Uma superfície multissensorial elástica de grande escala que permite usuários fazer música ao tocar em cima da tela foi desenvolvido em (CIBRIAN et al., 2017). Nessa superfície, os usuários podiam tocar sons de diferentes instrumentos musicais.

Diferente dos estudos mencionados anteriormente, este trabalho estende as capacidades de interação multimodal de um SAR chamado EVA, integrando efeitos sensoriais e reconhecimento de expressão facial em uma única plataforma baseada em robô. Javed e Park (2019), Santatiwongchai et al. (2016), Hirokawa et al. (2016), Bevill et al. (2016) e Feil-Seifer e Mataric' (2008) propõem o uso de robôs para terapias com crianças com TEA, sem oferecer a possibilidade de capturar a expressão facial da criança por meio de vídeo. Em (PARES et al., 2005; CIBRIAN et al., 2017; ZUBRYCKI; GRANOSIK, 2016) efeitos sensoriais são usados para terapias de TEA, incluindo efeitos de luz, mas eles não usam uma plataforma baseada em robô.

2.3 Jogos Sérios Aplicados em Terapias para Autistas

Oei e Patterson (2013) apontam para uma ligação entre jogar jogos de ação e a melhoria cognitiva e perceptiva. No entanto, essa melhoria não está associada apenas aos jogos de ação. Diferentes tipos de jogos podem melhorar diferentes aspectos cognitivos. Em (PAULA et al., 2020) foi feita uma implementação do jogo do *Stroop* para o sistema brasileiro de TV digital utilizando-se efeitos sensoriais de luz. Um jogo sério (JS) foi desenvolvido para treinar a atenção seletiva de idosos através do efeito *Stroop*. O jogo exibe na tela o nome de uma cor escrita com outra cor, então, o usuário através dos botões do controle remoto da TV, seleciona a cor correta como resposta. O efeito sensorial de luz é utilizado para auxiliar o usuário idoso na escolha da resposta correta.

Através do uso de jogos é possível proporcionar interatividade, aumentar a atividade mental e promover a interação social entre diversos usuários. O trabalho de Sandoval Bringas et al. (2016) descreve o desenvolvimento de um JS que pode ser usado por crianças com TEA para melhorar a comunicação e a interação social. Em (BARAJAS; AL

OSMAN; SHIRMOHAMMADI, 2017) foi proposto um JS para crianças com TEA com o objetivo de investigar os efeitos de um jogo como terapia lúdica. Por meio de um estudo empírico, os jogos convencionais (jogos de blocos não computacionais) foram comparados com o JS proposto. Evidências mostraram uma melhora na interação social das crianças, no processo de colaboração entre elas e uma diminuição nas brincadeiras solitárias.

O desenvolvimento emocional de uma criança envolve a capacidade de compreender seus próprios sentimentos, bem como aqueles de quem a cercam. Para uma criança com TEA, o processo de compreensão e expressão de sentimentos é muito difícil. Um JS foi desenvolvido em (TAN; HARROLD; ROSSER, 2013) com o objetivo de ajudar crianças com TEA, por meio de um processo de imitação, a reconhecer diferentes expressões faciais. Inspirado em (TAN; HARROLD; ROSSER, 2013), esta dissertação propõe uma sessão interativa para o robô EVA, desenvolvendo um jogo sério para terapias de regulação emocional de crianças com TEA.

2.4 Linguagens para Robôs

Frequentemente, pessoas como educadores, artistas e pesquisadores desejam poder programar e controlar robôs de uma maneira facilitada. No entanto, as ferramentas atuais para programação destes dispositivos podem ser difíceis de aprender, especialmente para pessoas sem formação técnica. Nilles et al. (2018) propõem o sistema IMPROV, uma linguagem de programação para descrição, em alto nível, do controle de movimentação de robôs, com visualização imediata do movimento resultante em um robô físico ou simulado. O IMPROV gera um código que funciona com o ROS (*Robot Operating System*), uma estrutura de software de robô de código aberto amplamente usada na academia e na indústria e integrada a muitos robôs disponíveis comercialmente. Os comandos no sistema IMPROV são compilados para mensagens ROS. A linguagem é inspirada em técnicas coreográficas, e permite ao usuário compor e transformar os movimentos no espaço e no tempo.

O trabalho de Baillie (2005) propõe a linguagem URBI com o intuito de torná-la a linguagem de baixo nível padrão para o controle de robôs. URBI é uma linguagem baseada em uma arquitetura cliente/servidor. Um servidor URBI roda no robô e um cliente envia comandos para o servidor a fim de controlar o robô. A comunicação entre o cliente e o servidor pode ser feita através de uma conexão TCP/IP ou se o cliente estiver executando no mesmo dispositivo do robô, a comunicação pode ser feita diretamente por

comunicação entre processos (IPC - *Inter-Process Communication*).

O trabalho de Pinciroli e Beltrame (2016) apresenta a linguagem BUZZ, destinada à programação de grupos de robôs. A sintaxe da linguagem mistura construções imperativas e funcionais e é inspirada em linguagens dinâmicas como JavaScript, Lua e Python. Em (VASQUEZ; MATIA, 2019) os autores desenvolveram uma linguagem de programação para um robô social chamado DORIS. O objetivo da proposta era a criação de uma linguagem simples, em que programadores pudessem programar facilmente e que fosse capaz de integrar os diferentes módulos da arquitetura do robô.

Tousignant, Van Wyk e Gini (2011) apresentam XROBOTS, um protótipo de DSL para a programação de robôs móveis. Ela é uma linguagem baseada em máquinas de estado hierárquico. Estados são tratados como entidades de primeira classe na linguagem, podendo ser passados como argumentos para outros estados parametrizados.

Uma das plataformas de robótica comerciais mais utilizadas na literatura é o robô NAO, fabricado pela Aldebaran Robotics[1]. O robô vem com uma ferramenta de programação visual chamada *Choregraphe* e um kit de desenvolvimento de software. As ferramentas que complementam o robô NAO são relativamente fáceis de entender para alguém com conhecimentos sobre programação. Por outro lado, é muito difícil para as pessoas que não têm conhecimento adequado sobre o assunto. Herwidodo, Zaini et al. (2015) propõem a utilização da própria língua indonésia como linguagem de controle para o NAO. Isso é feito através de um interpretador que interpreta os comandos na língua indonésia e os converte para a linguagem do robô.

Este trabalho propõe a linguagem EvaML, uma linguagem baseada em XML para a especificação de sessões interativas para a plataforma de robótica *open-source* EVA. Diferente das proposta de Nilles et al. (2018), a linguagem EvaML permite controlar todas as funcionalidades do robô e pode, através de variáveis específicas, ter acesso a todos os sensores do robô. Os trabalhos de (BAILLIE, 2005; PINCIROLI; BELTRAME, 2016; VASQUEZ; MATIA, 2019) apresentam propostas de linguagens de programação para robôs que são uma mistura de linguagens como C, Pascal, JavaScript e Python, enquanto esta dissertação apresenta a proposta de uma linguagem baseada em XML. Por ser baseada em XML, EvaML apresenta os elementos de controle do robô com mais legibilidade, diferente do trabalho de (TOUSIGNANT; VAN WYK; GINI, 2011), onde os elementos da linguagem são representados como máquinas de estado.

Em (HERWIDODO; ZAINI et al., 2015) a proposta apresentada destina-se ao robô

[1]https://www.solidworks.com/pt-br/story/aldebaran-robotics

NAO que é uma plataforma de robótica comercial e de custo alto. A nossa proposta é destinada a uma plataforma de robótica *open-source* e de baixo custo.

2.5 Simuladores de Robôs

Segundo Perico et al. (2016), os simuladores são uma solução comum para contornar problemas relacionados a cenários de teste do mundo real. Simuladores de robôs têm sido utilizados como ferramentas no treinamento educacional com evidências de que os resultados obtidos com o uso do robô físico, durante o treinamento, são semelhantes aos do uso do simulador (KURNIAWAN et al., 2018).

Vários campos da engenharia utilizam simuladores, e a robótica não é uma exceção. Existem muitos simuladores de robôs, alguns são de código aberto e outros são modelos comerciais. Os simuladores de robôs são comumente usados para o desenvolvimento de algoritmos sem que haja a necessidade de ter o robô montado fisicamente, facilitando a realização de experimentos, sem o risco de danificar o robô, e sendo também usados como ferramentas de apoio, ensinando as pessoas a programá-los.

Os simuladores desempenham um papel importante no projeto e desenvolvimento de sistemas robóticos. Através da simulação é possível verificar a viabilidade do projeto e desenvolver experimentos que podem ajudar a prever possíveis problemas. De acordo com (AYALA et al., 2020), os simuladores de código aberto mais usados e altamente cotados são *Webots* e *Gazebo*, e entre os simuladores de código fechado, destaca-se o simulador *V-REP*[2].

Na Coréia, a empresa Yujin Robotics[3] criou os robôs iRobi e iRobiQ, que são robôs educacionais e domésticos. Truong et al. (2011) apresentaram uma proposta de interface para facilitar o desenvolvimento de conteúdo para um simulador baseado nesses robôs, que foi criado considerando o alto custo de robôs educacionais reais. O simulador é voltado para desenvolvedores de conteúdo educacional, como professores.

No trabalho de Ishimura et al. (2003), foi proposto um ambiente de simulação open-source capaz de acomodar qualquer tipo de robô. O trabalho fornece o ambiente de simulação como um ambiente de testes primário para programadores de robôs iniciantes da plataforma de robótica AIBO[4] da Sony, acelerando a curva de aprendizado. Mais um

[2]Desde novembro de 2019, o V-REP foi substituído pelo CoppeliaSim versão 4.0. Veja https://www.coppeliarobotics.com/

[3]http://www.yujinrobot.com/

[4]https://us.aibo.com/

projeto de simulador robótico com a proposta para facilitar o aprendizado de programação para o robô Pololu 3PI[5] foi apresentado em (LIMA et al., 2018). Em (CARPIN et al., 2007), os autores apresentam uma proposta de um simulador de código aberto de alta fidelidade que pode ser usado tanto em pesquisa quanto em educação, o USARSim. Ele se baseia no Unreal Engine 2.0, um mecanismo de jogo disponível comercialmente produzido pela Epic Games para comercializar jogos de tiro em primeira pessoa.

No trabalho de Gena et al. (2021), há uma proposta de aplicativo que visa ampliar a funcionalidade da versão simulada do robô NAO (GOUAILLIER et al., 2009; POT et al., 2009), agregando ao simulador as características necessárias para o desenvolvimento de interações sociais. Os autores apontam que uma aplicação interessante do simulador seria utilizá-lo como um robô virtual que pudesse ser utilizado para dar continuidade às terapias na casa do paciente, sem o uso do robô físico.

Existem vários simuladores de robôs que foram testados em vários estudos como em (AYALA et al., 2020; KUMAR; REEL, 2011). Eles observaram que o uso de simuladores tem dois objetivos principais. O primeiro é auxiliar no processo de prototipagem e construção de robôs, fazendo uso de *frameworks* como a *engine* de física ODE (*Open Dynamics Engine*). O segundo objetivo principal é ajudar os usuários de robôs, seja na área industrial, educacional ou de saúde, a aprender a desenvolver programas que possam controlar esses robôs. Muitos simuladores requerem grandes recursos computacionais como placas 3D dedicadas, alguns não são *open-source*, muitos deles não são multiplataforma, rodando apenas em um sistema operacional específico, outros são *open-source* mas difíceis de configurar e usar, alguns são *open-source* mas são direcionados para plataformas de robótica comerciais.

Este trabalho apresenta um software simulador de código aberto chamado EvaSIM para o SAR EVA, que também é uma plataforma de robótica de código aberto. O simulador é multiplataforma e fácil de instalar e usar. É um software que pode rodar em hardware com baixo poder computacional, pois não utiliza gráficos pesados. Enquanto os simuladores robóticos são importantes para desenvolvedores de robôs tradicionais em termos de segurança, para um robô social é particularmente importante investigar como o robô é percebido e se sua comunicação verbal e não verbal (i.e, expressão facial, corporal, movimento e fala) é claramente identificada pelo usuário. O EvaSIM pode funcionar como uma ferramenta complementar ao robô EVA físico, podendo ser utilizado para treinamento de técnicos, entusiastas de IHR (Interação Humano-Robô) e profissionais de

[5]https://www.pololu. com/category/279/original-3pi-robot

saúde no desenvolvimento de sessões de terapias interativas multissensoriais para o robô físico, facilitando a prototipagem rápida de intervenções centradas no usuário. O próximo capítulo apresenta o robô EVA.

3 Robô EVA

3.1 Arquitetura do Robô

O robô EVA (*Embodied Voice Assistant*) é um robô social e foi desenvolvido originalmente por pesquisadores do CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada), em Baja California no México. O EVA foi desenvolvido como parte da tese de doutorado de Cruz Sandoval (2020) e depois foi estendido em outros trabalhos (MITJANS, 2020) e (ROCHA; VALENTIM et al., 2022).

3.1.1 Componentes de Hardware

A Figura 1 apresenta a imagem do robô EVA e seus componentes de hardware. Esta versão do robô inclui uma interface por voz, uma tela sensível ao toque, um anel de LEDs RGB integrado em seu tórax e servomotores que permitem que sua cabeça se mova com dois graus de liberdade (2DOF - *Degree Of Freedom*). O robô pode girar a cabeça para a esquerda, para a direita, para cima e para baixo. Os itens apresentados a seguir descrevem cada um dos componentes.

- *Tela de Toque*: O EVA tem uma tela de 5.5 polegadas que dá ao robô a capacidade de sintetizar expressões faciais, sendo elas: *neutra*, *raiva*, *tristeza* e *alegria*.
- *Raspberry PI 4*: Esta é a placa principal do robô, que roda a distribuição Linux para ARMv6 (Raspbian). As demais placas que complementam as funcionalidades do EVA são conectadas ao Raspberry.
- *ArbotiX-M*: Os dois servomotores que são responsáveis pelo movimento da cabeça do robô são controlados por este dispositivo[1]. A ArbotiX-M é uma placa baseada em Arduino e pode ser programada através da IDE do Arduino.

[1] https://www.trossenrobotics.com/p /arbotix-robot-controller.aspx

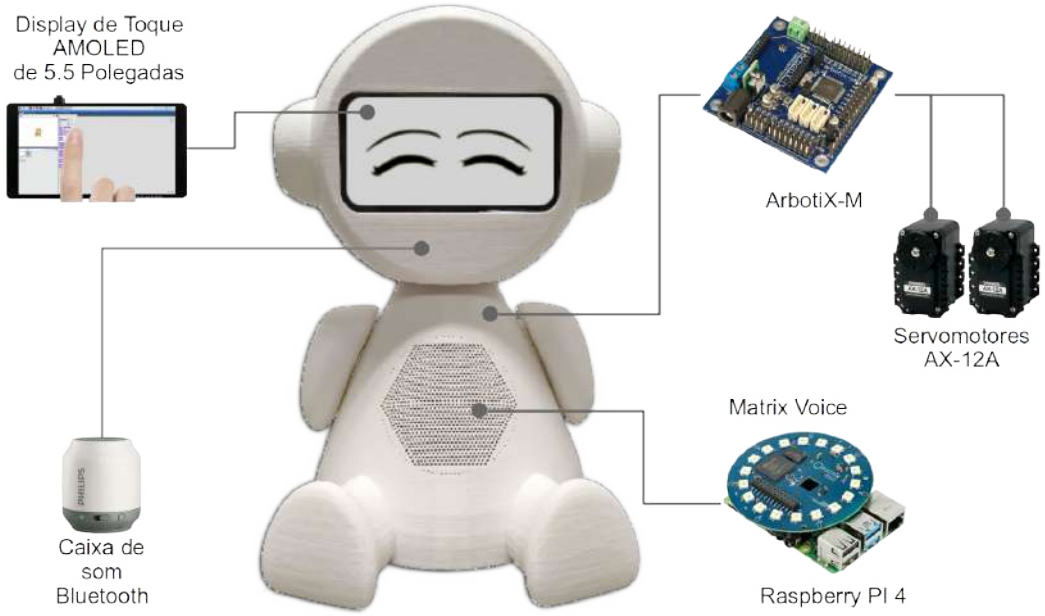

Figura 1: Componentes de hardware do robô EVA

- *Dois Servomotores AX12-A*: O robô usa dois servomotores Dynamixel AX-12A[2] que fornecem 2 graus de liberdade (DoF - *Degree of Freedom*) para a cabeça do EVA.
- *Caixa de Som Bluetooth*: O robô pode falar, usando recursos de *Text-To-Speech*[3] (TTS), pode emitir sons e tocar música usando uma caixa de som *Bluetooth*.
- *Matrix Voice*: Esta placa[4] tem duas funções distintas no projeto do robô EVA. A primeira é funcionar como dispositivo de captura de áudio, captando a voz do usuário, que futuramente será convertida para texto. Para o processo de captura de áudio, ela conta com oito microfones integrados. A placa também é usada como elemento de comunicação não verbal. Algumas animações com cores específicas são executadas usando a matriz de dezoito LEDs RGB da placa.

Como dito anteriormente o robô EVA foi criado por pesquisadores do CICESE, no México. Em Cruz-Sandoval e Favela (2019a), os criadores do EVA apresentam os resultados de um estudo utilizando o robô na condução de uma sessão de terapia não farmacológica com pacientes idosos com demência. A Figura 2 apresenta uma sessão de terapia de estimulação cognitiva com três pacientes em uma casa de repouso para idosos.

[2] https://www.robotis.us/dynamixel-ax-12a/
[3] Conversão de texto para áudio usando o serviço na nuvem do IBM Watson
[4] https://www.matrix.one/products/voice

Figura 2: Uma sessão de terapia de estimulação cognitiva com três participantes sendo conduzida pelo robô Eva (CRUZ-SANDOVAL; MORALES-TELLEZ et al., 2020)

3.1.2 Componentes de Software

O EVA possui uma aplicação web para desenvolvimento de scripts. Ela trabalha com um banco de dados para salvar e carregar os scripts criados e também mantém uma pasta no sistema de arquivos para armazenar arquivos de áudio, que podem ser tocados durante as interações. A aplicação, através de uma porta serial, controla os servomotores que movem a cabeça do robô. O sistema web foi desenvolvido usando a pilha de software MEAN, que inclui as tecnologias: *MongoDB*, *Express*, *AngularJS* e *NodeJS*. A linguagem C++ foi utilizada para o desenvolvimento do firmware de controle dos servomotores, que roda na placa ArbotiX-M, e para o código de controle das animações dos LEDs RGB, que roda na placa Matrix Voice. A Figura 3 mostra os componentes de software do robô.

A plataforma EVA inclui um componente para desenvolver sessões interativas (MITJANS, 2020). O usuário pode criar scripts de interação usando uma linguagem de programação visual (VPL - *Visual Programming Language*), definindo o fluxo do script usando sequências, condições e loops. A VPL possui vários elementos que podem ser usados para criar scripts de interação, alguns deles são: um elemento para definir a linguagem a ser usada pelos componentes de fala-para-texto e texto-para-fala, um elemento para controlar a expressão do olhar do robô, outro componente para reconhecimento de voz, um componente para reconhecimento das expressões faciais do usuário usando a *webcam*, e um componente capaz de controlar efeitos sensoriais de luz usando a lâmpada inteligente.

A Figura 4 mostra uma interação que foi construída usando a VPL. Por ser uma lin-

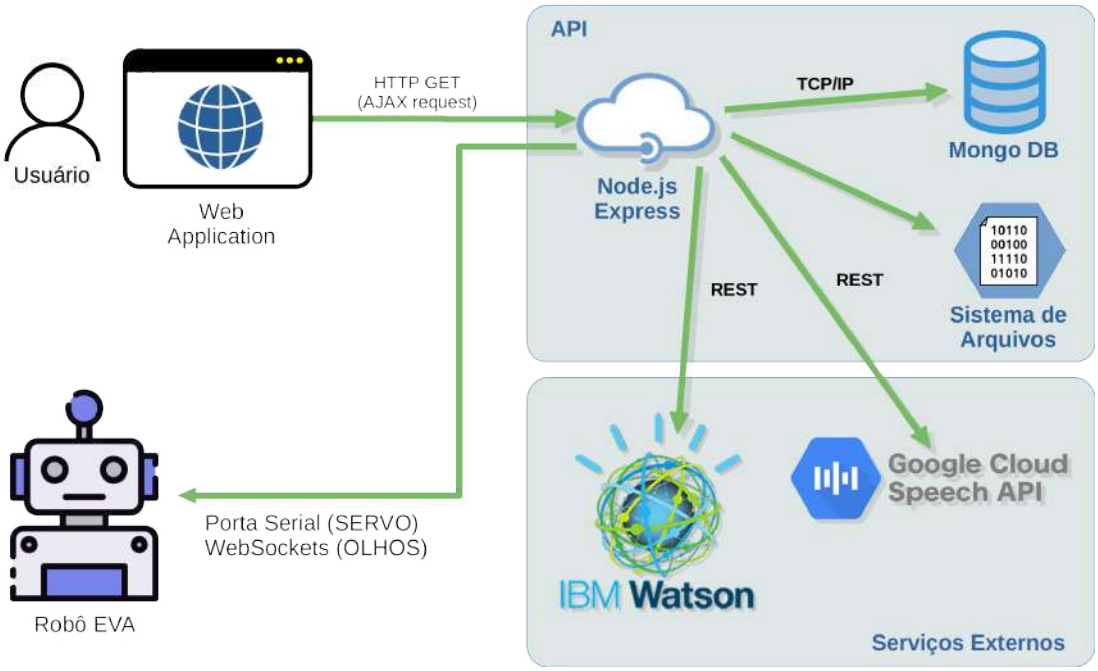

Figura 3: Arquitetura de software do robô EVA

guagem de programação visual, a VPL não possui detalhes sintáticos como uma linguagem de programação convencional. Ela foi criada com o objetivo de facilitar a construção de scripts de interação por pessoas que não são especialistas em programação. Usando esta ferramenta gráfica, construída com o framework web GoJS[5], um script pode ser desenvolvido de forma simples, apenas arrastando e soltando os elementos de controle da linguagem na interface gráfica.

O fluxo de execução de um script na VPL ocorre de cima para baixo e no caso de elementos condicionais, da esquerda para a direita. Um script na VPL é representado por um grafo onde os nós são os elementos da linguagem (capacidades do robô) e as arestas indicam a sequência do fluxo de execução. Através do uso de elementos condicionais é possível alterar o curso da execução do script dependendo de uma condição. Como pode ser visto na Figura 4, o script representado na imagem possui cinco elementos distintos. O elemento *Voice* é utilizado para configurar o idioma e timbre de voz utilizados nos processos de texto-para-fala e fala-para-texto. O elemento *Random* gera um número natural aleatório dentro de um intervalo especificado em seus parâmetros. O elemento *Condition* avalia uma condição, no exemplo avalia o número gerado aleatoriamente pelo comando anterior e, dependendo do resultado, o fluxo de execução do script pode seguir diferentes caminhos. O elemento que segue o caminho da esquerda é o *Light*, que controla a lâmpada

[5]https://gojs.net/latest/index.html

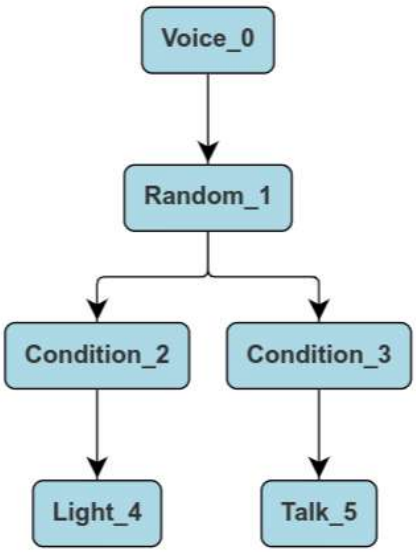

Figura 4: Um script na VPL

inteligente, seu estado (ligado ou desligado) e sua cor. O elemento que segue o caminho da direita é o *Talk*, que faz o EVA falar o texto especificado como parâmetro. Cada elemento da VPL possui um conjunto de parâmetros que são configurados no momento de sua inserção no script. O Apêndice B descreve o processo de montagem física do robô EVA.

3.2 Funcionalidades Principais do Robô EVA

Em sua versão básica, o robô possui capacidades de comunicação verbal e não verbal. Ele pode falar, gerando um áudio a partir de texto e também pode reconhecer a fala. Para o processo de TTS (*Text-To-Speech*), texto para fala, o robô usa os recursos cognitivos do IBM-Watson[6] e para o processo STT (*Speech-To-Text*), fala para texto, ele usa os serviços da API do Google[7]. O robô pode expressar emoções através do olhar e essas emoções podem ser enfatizadas com a movimentação da sua cabeça. É possível executar animações com os LEDs que ficam na placa Matrix Voice, que se encontra no tórax do robô.

3.2.1 Expressões do Olhar Usando o Display

O robô EVA pode expressar emoções através do seu olhar utilizando o display de 5.5 polegadas. Ele pode renderizar quatro tipos de expressões. A expressão que representa a emoção *neutra*, expressão padrão do robô, é apresentada assim que o robô é ligado.

[6] https://www.ibm.com/watson/developercloud/text-to-speech.html
[7] https://cloud.google.com/speech/

A segunda expressão é a expressão de *felicidade*. Ele pode expressar também *tristeza* e *raiva*. A Figura 5 mostra a imagem das quatro expressões do robô.

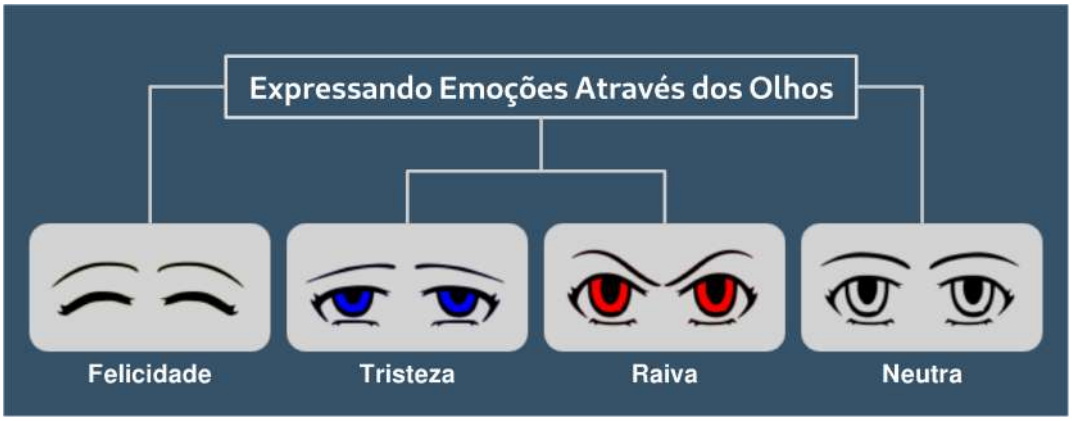

Figura 5: Expressões sintetizadas pelo robô EVA

3.2.1.1 Reconhecimento de Voz

A interação por voz é um recurso importante no processo de interação homem-robô. O EVA pode capturar o áudio da fala do usuário, usando os microfones que se encontram na placa Matrix Voice, que fica no seu tórax. Após a captura do sinal de áudio, o arquivo é enviado para a API do Google que transforma o conteúdo da fala em um arquivo de texto. A string retornada é então processada pelo robô e utilizada no processo de comunicação com o usuário.

3.2.2 Fala

Utilizando os recursos cognitivos do IBM Watson, o robô consegue transformar texto em fala. O conjunto de comandos da VPL do robô contém com um comando que recebe como parâmetro um texto e envia esse texto para a nuvem, que então retorna o áudio do texto sintetizado. O serviço do IBM Watson permite que seja definido o timbre de voz da fala a ser gerada. Existem opções de vozes masculinas e femininas. A escolha do timbre de voz está atrelada diretamente ao idioma usado no processo texto-para-fala do robô.

3.2.3 Execução de Arquivos de Áudio

Como foi mostrado na Figura 1, o EVA conta com uma caixa de som *Bluetooth*. Esse dispositivo permite que o robô possa tocar arquivos de áudio no formato *wav*. Para ser usado pelo robô, isto é, poder ser acessado pelo comando responsável por tocar áudio, o

arquivo de som deve estar dentro da pasta "sonidos" que se encontra no diretório raiz da aplicação de controle do robô dentro do Raspberry PI.

3.2.4 Animação dos LEDs RGB

Como um recurso de comunicação não verbal, o robô utiliza a matriz com dezoito LEDs RGB que se encontra na sua placa Matrix Voice. A linguagem visual do robô possui um comando que permite a execução de um conjunto de animações predefinidas usando esses LEDs. Algumas dessas animações estão associadas a outras ações do robô e são executadas junto com os comandos relacionados. Por exemplo, ao falar, o robô executa uma animação com os LEDs na cor azul, e ao escutar, o robô executa uma animação com os LEDs na cor verde.

3.2.5 Movimentação da Cabeça

O robô usa dois servomotores Dynamixel AX-12A[8] que fornecem 2 graus de liberdade para a cabeça do EVA. O comando de controle do movimento da cabeça do robô pode fazer com que ele mova a cabeça para cima, para baixo, para a esquerda e para a direita. Para cada movimento, há uma opção que realiza o movimento com menor ou com maior amplitude. Há também a opção de movimentação chamada "sim" que faz com que o robô execute um movimento de sobe e desce com a cabeça, como se estivesse respondendo "sim" com a cabeça. Da mesma maneira, há o movimento de "não". A movimentação da cabeça do robô serve como um componente capaz de aumentar a expressividade do robô, por exemplo quando o robô demonstra uma expressão de raiva através do olhar, essa expressão pode ser acentuada fazendo com que a cabeça do robô se incline para baixo.

Este capítulo apresentou um visão geral sobre o robô EVA. O próximo capítulo apresenta a proposta de extensão das capacidades de interação multimodal do robô e o jogo sério desenvolvido para terapia de crianças com TEA.

[8] https://www.robotis.us/dynamixel-ax-12a/

4 Proposta de Jogo Sério para Crianças com TEA

4.1 Estendendo as Capacidades de Interação Multimodal do EVA

Como mencionado anteriormente, o robô EVA é capaz de realizar uma sessão personalizada por meio de scripts de interação. Esses scripts podem ser facilmente criados usando uma VPL projetada exclusivamente para o EVA (MITJANS, 2020). A VPL não se preocupa com os detalhes sintáticos de uma linguagem de programação convencional, ela foi criada para facilitar a construção de scripts de interação permitindo que pessoas não especializadas em programação criem seus próprios scripts.

A linguagem possui vários componentes, incluindo aqueles que controlam o fluxo de execução do script, contadores, temporizadores, controles condicionais e controles de fala e escuta do robô. Executando o script de interação, o robô pode interagir com o usuário, obtendo informações por meio da escuta e se expressando por meio da fala, expressões faciais, movimentação da cabeça e animações com os leds no seu tórax.

Esta dissertação estende a linguagem gráfica do robô, adicionando a ela dois novos componentes como elementos de primeira classe da linguagem. A VPL foi estendida para incluir efeitos sensoriais de luz nos scripts de terapia, para que o EVA possa se comunicar com uma lâmpada inteligente usando uma interface Wi-Fi para ativá-la ou desativá-la. A linguagem do robô também foi estendida para capturar a emoção do usuário por meio do reconhecimento de expressões faciais, portanto, uma câmera integrada é usada para capturar a imagem do usuário e analisar sua expressão facial para identificar se ele está feliz, triste, irritado ou neutro (indiferente). A Seção 4.1.1 apresenta a proposta de inclusão de efeitos sensoriais de luz e a Seção 4.1.2 discute a proposta de reconhecimento de expressões faciais (ROCHA; VALENTIM et al., 2022).

4.1.1 Controle de Efeitos Sensoriais de Luz

Esta dissertação propõe a extensão da plataforma EVA com um novo componente adicionado como elemento de primeira classe à sua linguagem de programação visual. Esse elemento é chamado de *Light*, e dá ao robô a capacidade de controlar efeitos sensoriais de luz usando uma lâmpada inteligente. A Figura 6 mostra o esquema de conexão e controle do robô sobre a lâmpada inteligente. Após a lâmpada estar configurada corretamente na rede wifi, o robô, através de uma conexão TCP, se conecta à lâmpada. Através do comando *Light*, o robô pode ligar e desligar a lâmpada, como também pode definir a sua cor usando uma paleta de cores RGB com 24 bits de cor. O comando de controle e seleção de cor é enviado do robô para a lâmpada como uma string JSON.

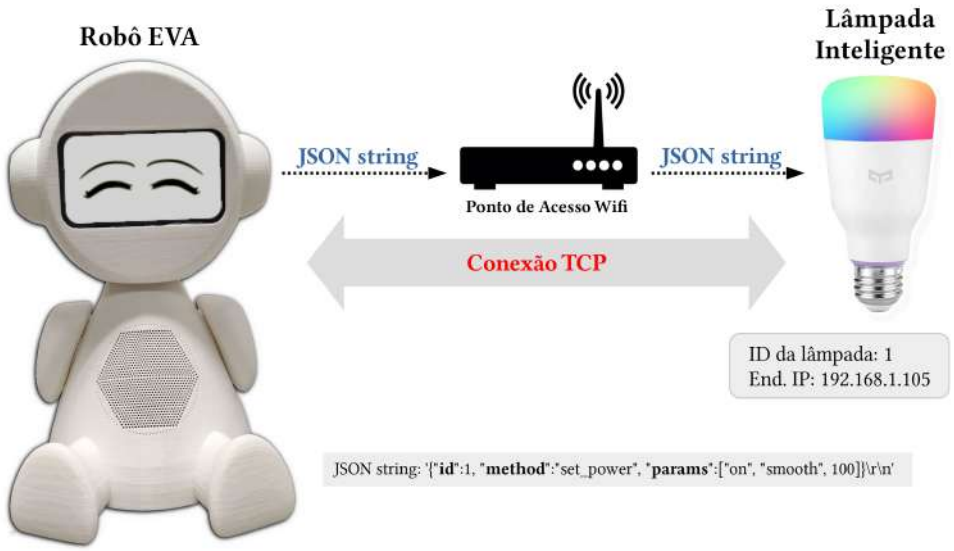

Figura 6: Controlando efeitos sensoriais de luz

O vídeo disponível neste link[1] ilustra as capacidades do robô EVA.

4.1.2 Reconhecimento de Expressões Faciais

A capacidade de identificar a emoção do usuário por meio do reconhecimento de expressões faciais e utilizar essas informações dentro do aplicativo é uma facilidade que amplia o poder de interação do robô com o usuário. Para isso, uma segunda extensão é proposta à VPL dando ao robô a capacidade de reconhecer expressões faciais. Esse novo componente é denominado *UserEmotion*. Ele usa os serviços de um módulo externo desenvolvido em Python (VALENTIM; BARRETO; MUCHALUAT-SAADE, 2020). Tanto o módulo Python quanto o software EVA rodam no mesmo dispositivo, um Raspberry PI 4.

[1] https://bit.ly/3kNt4u3

A comunicação entre o módulo e o robô é feita através de uma conexão TCP. Quando iniciado, o módulo de reconhecimento facial cria um servidor TCP que fica aguardando a conexão com o robô. Após a conexão ser estabelecida o módulo de reconhecimento facial aciona a *webcam*, iniciando assim os processos de captura de imagem e inferência da expressão facial. Ao final do procedimento, o módulo retorna ao cliente TCP (EVA), uma string contendo a expressão identificada. O módulo de reconhecimento facial pode retornar as seguintes expressões: "NEUTRAL", "ANGRY", "DISGUST", "FEAR", "SURPRISE", "HAPPY" e "SAD". Para obter a expressão facial do usuário, através do componente *UserEmotion*, o robô envia uma solicitação ao módulo de reconhecimento facial Python. O processo funciona conforme ilustrado na Figura 7.

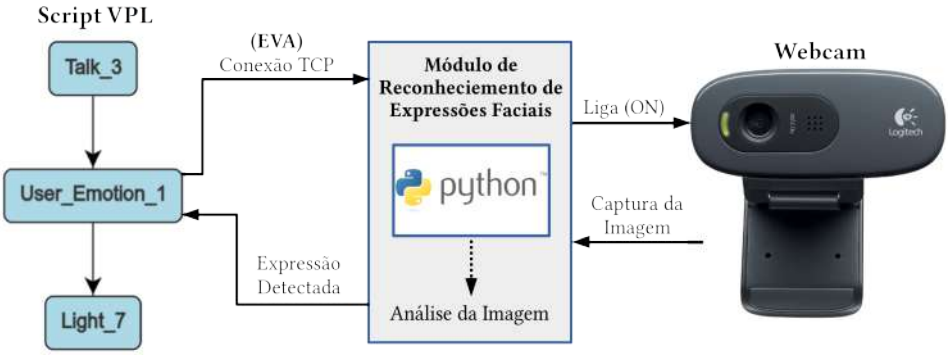

Figura 7: Reconhecimento de expressões faciais

Foram feitos testes para avaliar a acurácia do módulo de expressão facial no reconhecimento das seguintes emoções do usuário: "HAPPY", "ANGRY" e "SAD". Estes três tipos de expressões faciais foram escolhidas, pois são as que atualmente podem ser expressas pelo EVA e foram utilizadas na sessão de terapia que foi implementada conforme será discutido na seção seguinte. Para o teste de acurácia, um script foi implementado na linguagem visual do EVA para reconhecer 30 expressões faciais. Dois usuários testaram o programa com 5 expressões de cada tipo "HAPPY", "ANGRY"e "SAD". Considerando 30 eventos de reconhecimento de expressões faciais, o EVA reconheceu corretamente 24, obtendo uma acurácia de 80%. Considerando apenas a expressão "HAPPY", a acurácia do EVA foi de 100%. Os testes foram realizados por dois adultos. Assim, o resultado foi considerado satisfatório podendo ser utilizado na prática.

4.1.3 Um Jogo Sério para Crianças Autistas em 3 Estágios

Para servir como objeto de avaliação da proposta de extensão das capacidades de interação multimodais do EVA, apresentada nas Seções 4.1.1 e 4.1.2, esta dissertação desenvolveu

um jogo sério utilizando a linguagem de programação VPL do robô EVA. O jogo tem como objetivo auxiliar nas terapias de regulação emocional para crianças com TEA.

Durante a fase de design, a ideia do jogo foi apresentada a um terapeuta especialista em TEA. O terapeuta recomendou que fossem enfatizadas apenas as respostas corretas dadas pela criança. O EVA não deveria dizer, por exemplo, "sua resposta está errada" para a criança. Por outro lado, o EVA deveria elogiar a criança quando a resposta dada fosse a correta. Este procedimento é utilizado em intervenções em Análise do Comportamento Aplicada denominadas *Discrete Trial Training* (DTT) (SMITH, 2001). A DTT é uma prática que utiliza instruções com tentativas repetidas de ensino, onde cada tentativa apresenta uma resposta e um reforço a essa resposta, se for o caso. É um treinamento totalmente estruturado, com início e fim claramente definidos. Esse tipo de intervenção experimental discreta aumenta a probabilidade de respostas corretas do paciente com base no aprendizado livre de erros. A interação entre terapeuta e criança deve ser realizada em um ambiente sem distrações, com instruções claras, concisas e objetivas e com o reforço imediato para cada resposta correta.

Foi criado um jogo com três estágios. O primeiro estágio foi chamado de *jogo das cores*, no qual a criança é solicitada a identificar as cores que são apresentadas pelo robô usando os efeitos sensoriais de luz. No segundo estágio, denominado *jogo das emoções*, o robô apresenta diversas expressões faciais, enquanto a criança tenta identificá-las. Nesses dois primeiros estágios a criança interage com o robô via voz. No terceiro estágio, chamado *jogo da imitação*, a criança precisa imitar as emoções do robô com suas próprias expressões faciais.

Cada etapa inclui três perguntas. Se a criança der uma resposta incorreta, ela terá uma nova chance de tentar acertar a resposta correta. O estágio 1 segue o fluxograma mostrado na Figura 8.

Usando o recurso texto para fala do robô, um texto de saudação é transformado em fala e o jogo começa com o robô cumprimentando a criança. Em seguida, o robô escuta, aguardando a resposta da criança e através do recurso fala para texto o nome do participante é capturado, transformado em *string* e armazenado em uma variável. Em seguida, o robô, utilizando o componente *Light* proposto neste trabalho, apresenta a primeira cor, acendendo a lâmpada e pedindo à criança que identifique a cor apresentada. Nesse ponto, usando o recurso de teste condicional do VPL, a resposta da criança, já transformada em *string*, é comparada com a *string* que representa a resposta correta. Caso a criança acerte, o robô a parabeniza e verifica se atingiu o número máximo de

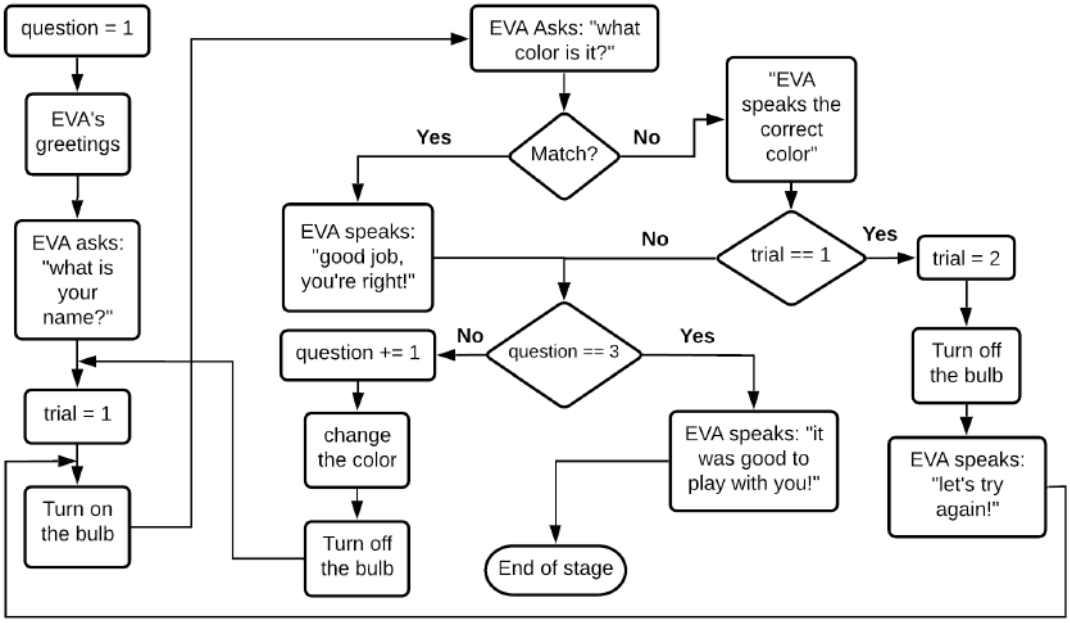

Figura 8: Fluxograma do estágio 1 do jogo

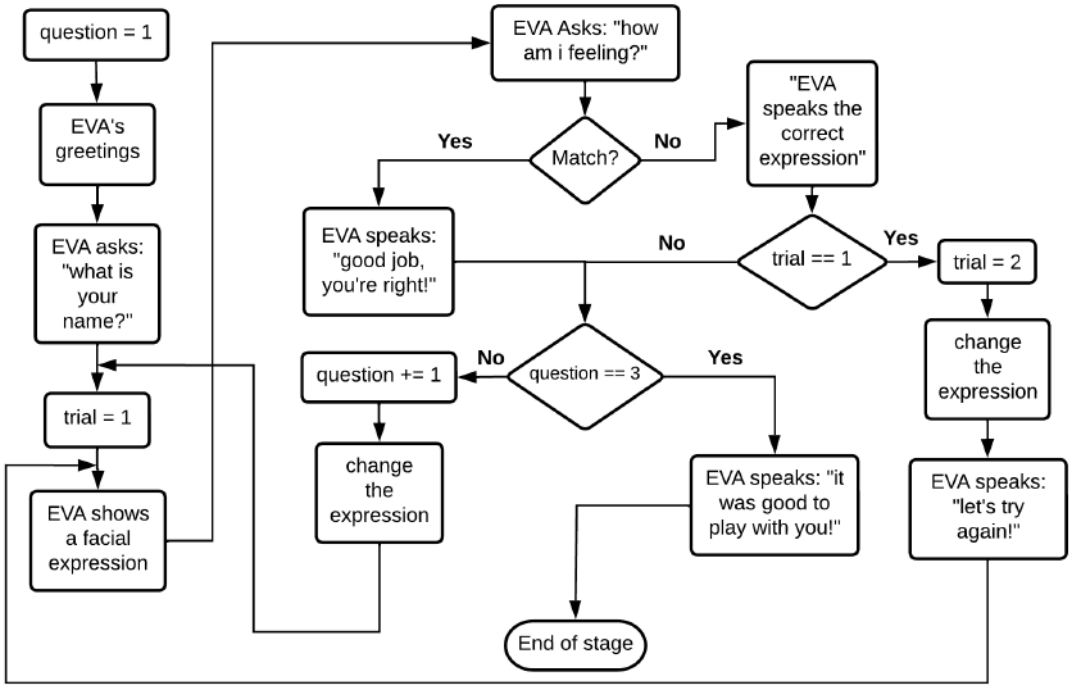

Figura 9: Fluxograma do estágio 2 do jogo

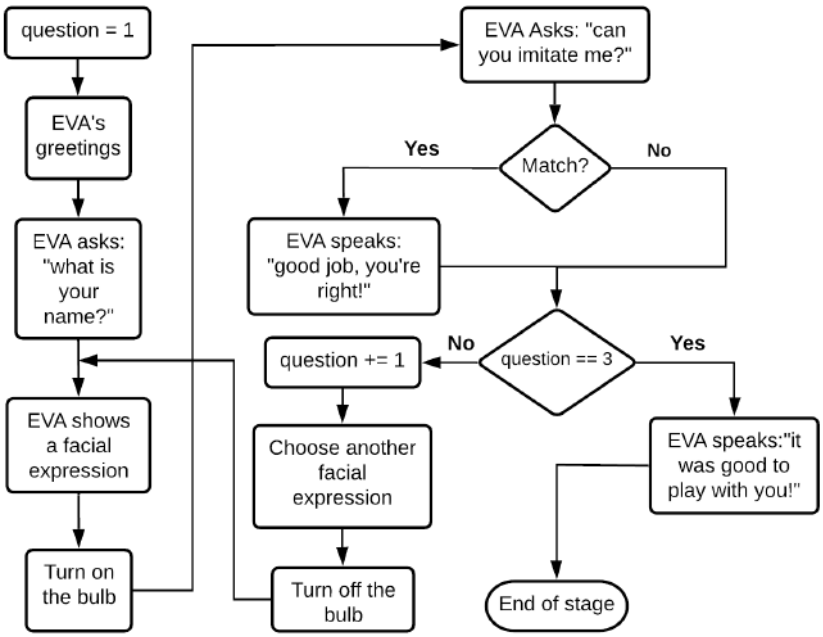

Figura 10: Fluxograma do estágio 3 do jogo

questões, caso contrário, uma nova cor é selecionada e o processo de apresentação é repetido apresentando uma nova cor. Se a criança cometer um erro, o robô fala o nome correto da cor e dá outra oportunidade para a criança tentar novamente. Após três perguntas, o robô dá uma saudação final e o jogo termina.

O fluxograma do estágio 2 pode ser visualizado na Figura 9. Agora, a criança precisa reconhecer as emoções do robô. Portanto, ao invés de apresentar as cores e acender a lâmpada, o robô apresenta suas emoções através do olhar. Três emoções são representadas: *felicidade, raiva e tristeza.*

No estágio 3, a lógica do jogo muda conforme mostrado na Figura 10. O robô mostra uma emoção (através do olhar) e pede à criança que o imite. Diferentemente das etapas anteriores em que o robô recebia a resposta via interação por voz, neste estágio, o robô obtém a resposta através do reconhecimento da expressão facial da criança, utilizando o componente *userEmotion* proposto neste trabalho. O robô capta a expressão da criança e verifica se ela deu a resposta correta. O EVA parabeniza a criança quando sua imitação é reconhecida corretamente. Três expressões faciais são apresentadas em uma partida.

4.2 Avaliação da Proposta

Para avaliar a proposta apresentada, utilizou-se o modelo TAM (Modelo de Aceitação de Tecnologia) (MARANGUNIĆ; GRANIĆ, 2015). Este instrumento é amplamente utilizado para avaliação de tecnologia aplicada à saúde (HU et al., 1999). O TAM é baseado em um questionário que avalia a Intenção de Uso da tecnologia avaliando a Utilidade Percebida (PU - *Perceived Usefulness*), "o grau em que uma pessoa acredita que o uso de um determinado sistema melhoraria seu desempenho no trabalho"; e Facilidade de Uso Percebida (PEOU - *Perceived Ease of Use*) (TAN; HARROLD; ROSSER, 2013), "o grau em que uma pessoa acredita que usar um determinado sistema seria livre de esforço"(DAVIS, 1989).

Figura 11: Criança jogando o jogo de imitação (Versão incompleta das peças do robô)

Conforme sugerido por Salleh et al. (2017), o jogo foi testado por uma criança neurotípica brasileira de 6 anos com o consentimento de seus pais para participar do experimento. O robô falava em português na sessão com a criança. Como nem todas as partes físicas do robô estavam disponíveis na época do teste, uma TV foi conectada ao Raspberry Pi para mostrar os olhos do robô à criança. Alguns vídeos da criança jogando cada etapa do jogo com EVA[2] [3] [4] foram gravados, com o consentimento dos pais. A Figura 11 mostra a criança jogando o estágio 3 do jogo com configuração inicial apresentada nos vídeos. Já na Figura 12, pode-se ver a mesma criança, jogando o *jogo da imitação* com a versão do robô completamente montado com todas as suas peças.

O questionário TAM foi adaptado para avaliação com 8 perguntas relativas a métrica

[2] https://youtu.be/ScyzGQNp_w0
[3] https://www.youtube.com/watch?v=KnyPr1Bc3Lo
[4] https://www.youtube.com/watch?v=PU8BLwTkGaw

Figura 12: Criança jogando o jogo de imitação (Versão completa do robô)

PU (Utilidade Percebida) e 5 perguntas relativas à métrica PEOU (Facilidade de Uso Percebida), conforme detalhado na Tabela 1. Foi pedido aos usuários que assistissem aos vídeos e respondessem o questionário. As questões para avaliar a utilidade percebida (PU) e a facilidade de uso percebida (PEOU) utilizaram a escala *Likert* variando de (1) discordo fortemente a (5) concordo fortemente.

Um grupo de 48 usuários adultos, com idades entre 19 e 61 anos, incluindo profissionais de saúde e estudantes, como médicos, psiquiatras, psicólogos, enfermeiros, psicopedagogos, educadores, fonoaudiólogos e terapeutas ocupacionais, avaliou a proposta. Todos deram seu consentimento para participar do experimento, caso contrário não poderiam assistir aos vídeos e responder ao questionário. Consideraram-se os profissionais com dois ou mais anos de experiência no cuidado de crianças com TEA como especialistas e aqueles com menos de dois anos de experiência, incluindo estudantes, como iniciantes (SAMONTE et al., 2020). Quatro participantes foram excluídos do experimento por não terem especificado seu curso de graduação. Portanto, o grupo final foi dividido em 24 especialistas em TEA e 20 iniciantes em TEA. A Tabela 2 mostra os resultados da mediana (M_d), da moda (M_o), o nível de discordância[5] (ND), o nível de neutralidade[6] (NN) e o nível de concordância[7] (NC) para as questões de Utilidade Percebida (PU1 a PU8) e Facilidade Percebida de Uso (PEOU1 a PEOU5), para cada grupo, o dos especialistas e o dos iniciantes. Os resultados com medianas menores que 3,0 indicam que os parti-

[5]Nível de discordância (ND) é a porcentagem do número de participantes que discordam parcialmente ou discordam fortemente.

[6]Nível de neutralidade (NN) é a porcentagem do número de participantes que nem discordam e nem concordam, ou seja, se colocam como neutros.

[7]Nível de concordância (NC) é a porcentagem do número de participantes que concordam parcialmente ou concordam fortemente.

Tabela 1: Questionário TAM

Questão	Utilidade Percebida
PU1	O uso da tecnologia me ajuda a fazer meu trabalho
PU2	O robô interativo EVA seria útil para me auxiliar em terapias para TEA
PU3	O robô interativo EVA atrapalharia minhas sessões de terapia para TEA
PU4	O jogo de reconhecimento de cores seria útil para me auxiliar em terapias para TEA
PU5	O jogo para reconhecer emoções seria útil para me auxiliar em terapias para TEA
PU6	O jogo para imitar as emoções do robô seria útil para me auxiliar em terapias para TEA
PU7	O robô interativo EVA pode me ajudar a aumentar a eficácia das terapias para TEA no reconhecimento de emoções
PU8	O robô interativo EVA pode ser útil em outros tipos de terapias
Questão	**Facilidade de Uso Percebida**
PEOU1	Uma criança com TEA interagiria facilmente com o robô por meio da voz
PEOU2	O efeito de luz torna a sessão de terapia com crianças com TEA MENOS atrativa
PEOU3	Efeitos sonoros tornam a sessão de terapia com crianças com TEA MAIS atrativa
PEOU4	Acho que seria fácil usar o robô nas minhas sessões de terapia com crianças com TEA
PEOU5	Eu precisaria de muito esforço para usar o robô em minhas sessões de terapia

cipantes discordam fortemente/parcialmente com a questão. Resultados com medianas iguais a 3,0 indicam que os participantes não discordam e nem concordam com a questão, demonstrando um posição de neutralidade. Resultados com medianas maiores ou iguais a 4,0 indicam que os participantes concordam fortemente/parcialmente com a questão. As Figuras 13 e 14 apresentam as respostas obtidas para cada uma das questões do questionário TAM para os dois grupos, o grupo dos especialistas e o grupo dos iniciantes, respectivamente. Cada região colorida das barras contém a indicação da quantidade de participantes que escolheram a opção indicada.

Com base nos resultados obtidos para as questões do tipo PU, pode-se concluir que a maioria dos especialistas concorda que o EVA seria útil para ajudá-los durante as sessões de terapia (PU1, PU2, PU3 e PU8). Observe que PU3 é uma questão negativa onde um maior nível de discordância é desejado e obtido (ND = 70,83% e $M_d = 2$). Os especialistas, com um nível de concordância maior que 83% e todos com medianas iguais a 5, também concordam que as extensões propostas neste trabalho são úteis e os ajudariam a realizar seu trabalho de forma mais eficaz (PU4, PU5, PU6 e PU7).

Considerando os resultados das questões do tipo PEOU, a maioria dos especialistas concorda que seria fácil usar o EVA em suas sessões de terapia de TEA (PEOU4) sem

Tabela 2: Questionário TAM - Resultados

Questão	**Especialistas (24)**					**Iniciantes (20)**					**Mann-Whitney**	
	M_d	M_o	**ND**	**NN**	**NC**	M_d	M_o	**ND**	**NN**	**NC**	U	p-exato
PU1	5	5	4,17%	0,00%	95,83%	4	4	5,00%	0,00%	95,00%	206,5	0.216
PU2	5	5	12,50%	0,00%	87,50%	5	5	10,00%	5,00%	85,00%	238,5	0,486
PU3	2	1	70,83%	8,33%	20,83%	2	2	65,00%	5,00%	30,00%	196	0,154
PU4	5	5	8,33%	8,33%	83,33%	4	4 e 5	10,00%	0,00%	90,00%	218	0,307
PU5	5	5	4,17%	0,00%	95,83%	4	4 e 5	10,00%	0,00%	90,00%	186	0,104
PU6	5	5	4,17%	0,00%	95,83%	5	5	10,00%	0,00%	90,00%	227	0,385
PU7	5	5	12,50%	0,00%	87,50%	4	5	10,00%	5,00%	85,00%	202	0,19
PU8	5	5	8,33%	0,00%	91,67%	5	5	5,00%	0,00%	95,00%	222	0,341
PEOU1	4	4	12,50%	12,50%	75,00%	4	4	15,00%	5,00%	80,00%	223	0,35
PEOU2	2	4	54,17%	0,00%	45,83%	3	1	50,00%	25,00%	25,00%	207	0,223
PEOU3	**4**	**4**	**25,00%**	**4,17%**	**70,83%**	**5**	**5**	**5,00%**	**10,00%**	**85,00%**	**169,5**	**0,048**
PEOU4	4	4	12,50%	8,33%	79,17%	5	5	5,00%	10,00%	85,00%	205	0,209
PEOU5	2	1	70,83%	8,33%	20,83%	2	2	65,00%	10,00%	25,00%	213,5	0,267

muito esforço (PEOU5). Por outro lado, alguns especialistas indicaram que os pacientes com TEA podem ter dificuldade de interagir com o EVA por meio da voz (PEOU1), dependendo do nível de comprometimento da comunicação social que possuem. Com relação à questão PEOU2, sobre a utilização de efeitos sensoriais de luz tornar as sessões de terapia menos atrativas, os especialistas mostraram que discordam da afirmação, porém, com um nível de discordância de apenas 54,17% contra 45,83% de nível de concordância. Analisando as respostas dos iniciantes, com base nos níveis de discordância ($ND = 50{,}00\%$), neutralidade ($NN = 25{,}00\%$) e concordância ($NC = 25{,}00\%$), pode-se ver que apesar do nível de discordância ser maior que o nível de concordância, não houve maioria no grupo para essa questão. Com base no valor de mediana ($M_d = 3$), concluímos que os iniciantes não discordam e nem concordam sobre a utilização de efeitos sensoriais de luz tornar as sessões de terapia menos atrativas. Por outro lado, considerando os efeitos sonoros, 25,00% dos especialistas demonstrou ter algum tipo de preocupação com seu uso (PEOU3).

Foi feita uma análise estatística usando o teste não paramétrico de *Mann-Whitney* (MCKNIGHT; NAJAB, 2010) para comparar as respostas do TAM entre os grupos de especialistas e iniciantes para cada questão. A Tabela 2 mostra os valores de U e os resultado de p-exato unicaudais nas suas últimas colunas. Foi encontrada diferença estatística apenas na questão PEOU3 ($U = 169{,}5$; $p < 0{,}05$), onde os iniciantes constataram que os efeitos sonoros tornam a sessão de terapia mais atrativa que os especialistas. Para o

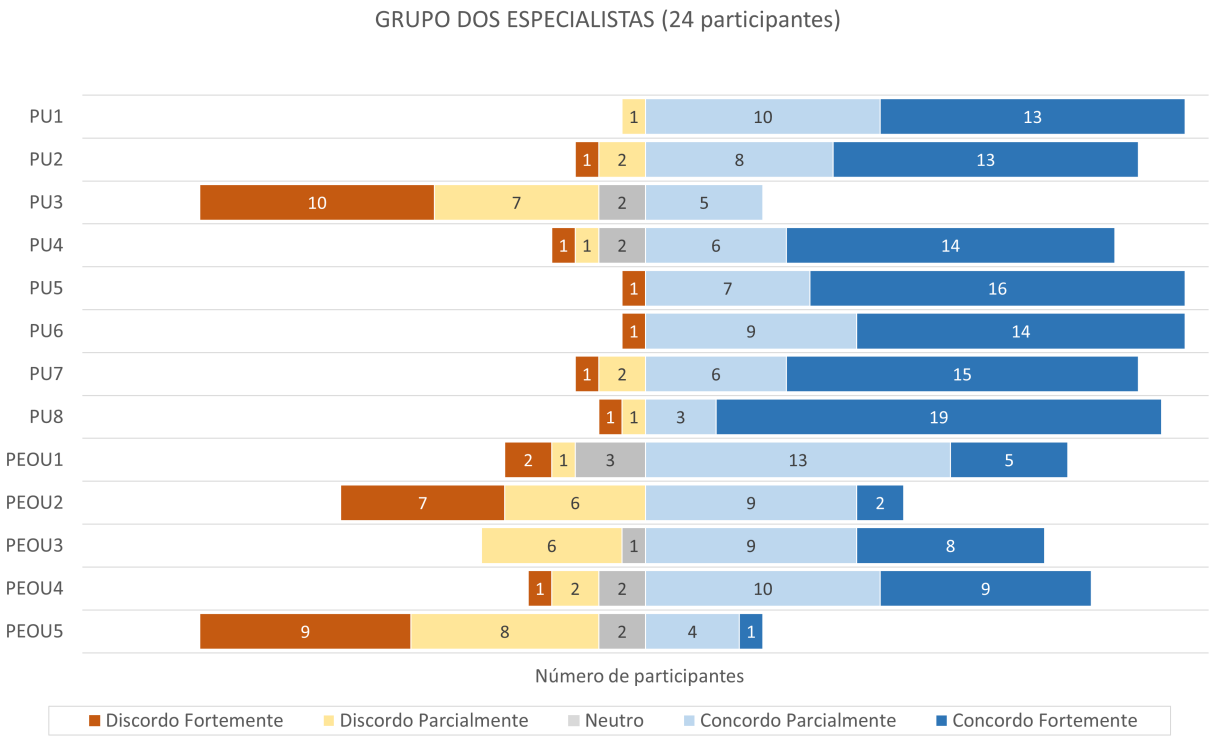

Figura 13: Respostas do grupo dos especialistas paras as questões relacionadas à Utilidade Percebida (PU) e Facilidade de Uso Percebida (PEOU)

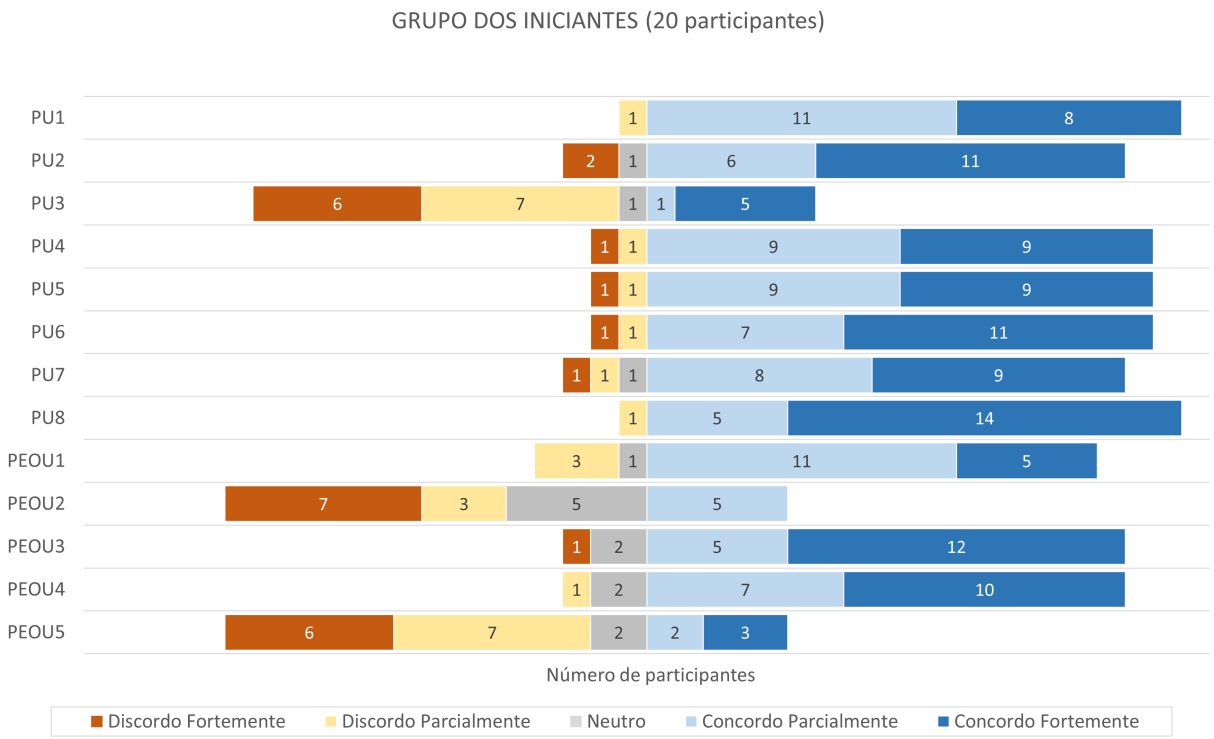

Figura 14: Respostas do grupo dos iniciantes paras as questões relacionadas à Utilidade Percebida (PU) e Facilidade de Uso Percebida (PEOU)

grupo dos especialistas obteve-se M_d=4 e para o grupo dos iniciantes M_d=5. Em todas as outras perguntas, iniciantes e especialistas concordam em suas respostas. Analisando os resultados do TAM, pode-se concluir que o EVA é útil e fácil de usar para terapias de TEA de acordo com os profissionais de saúde.

Os participantes também deram sugestões para melhorar este trabalho, tais como: perguntar a idade da criança antes de iniciar a terapia, incluindo uma fase de teste antes do jogo começar, aumentar o tempo de permanência da lâmpada inteligente, implementar uma versão para *tablet* ou *smartphone* do EVA, implementar o EVA como avatar, incluir atividades com música, melhorar as expressões do robô mostrando sua boca além dos olhos, disponibilizar jogos para reconhecer vogais, consoantes e números, fornecer questões simples de matemática e português e personalizar a sessão de terapia de acordo com a criança.

4.3 Considerações Finais

Este capítulo apresentou uma extensão do robô EVA projetado para terapias do Transtorno do Espectro do Autismo (TEA). Esta proposta aprimorou as capacidades do EVA para interação multimodal usando efeitos sensoriais de luz com uma lâmpada inteligente e reconhecimento de expressão facial com uma webcam para reconhecer as emoções das crianças.

Foi desenvolvido um jogo sério com três etapas que podem auxiliar as terapias de regulação emocional para crianças com TEA. A primeira etapa é um jogo onde a criança tem que identificar as cores da lâmpada inteligente. A segunda etapa é um jogo onde a criança tem que identificar as emoções do robô. E a terceira etapa pede que a criança imite as emoções do robô com suas próprias expressões faciais. O jogo foi avaliado por 44 profissionais de saúde utilizando o modelo TAM. Os resultados foram muito promissores, indicando que os profissionais de saúde consideraram a proposta útil e fácil de usar em sessões de terapia para crianças com TEA.

O próximo capítulo apresenta EvaML, uma linguagem baseada em XML para o robô EVA.

5 Linguagem EvaML

Originalmente, o robô EVA possui uma linguagem de programação visual (VPL), que visa facilitar o desenvolvimento de interações por pessoas sem experiência em programação de computadores. A VPL foi construída usando uma poderosa ferramenta gráfica chamada GoJS[1]. Durante o processo de criação de um script utilizando a VPL, o usuário é responsável por inserir os comandos do robô, que são nós no fluxo de execução. No entanto, os elementos são posicionados automaticamente na área da interface gráfica pelo framework GoJS. Os elementos do grafo que representam o script são reposicionados automaticamente após a remoção ou inserção de um nó ou aresta do grafo. Quando um script se torna grande, esse reposicionamento automático geralmente dificulta bastante o processo de criação do script, pois muitas vezes, os elementos (nós) acabam tendo suas posições alteradas gerando uma confusão visual. À medida que o número de elementos gráficos no script de interação aumenta, a visualização do gráfico se torna muito limitada. Para se ter uma visão geral do gráfico, a interface permite utilizar os recursos de "*zoom in*" e "*zoom out*", porém, com o tamanho reduzido dos elementos, fica muito difícil identificar visualmente cada um deles. É possível selecionar um grupo de nós e arestas e duplicá-los usando o recurso de copiar e colar, o problema, nesse caso, é que o script passa a ter elementos diferentes (que podem ter propriedades diferentes) com o mesmo identificador visual (nome) e isso dificulta muito o processo de edição das propriedades desses elementos duplicados. A observação destas limitações da linguagem visual do robô surgiu durante o desenvolvimento do jogo sério apresentado no capítulo anterior (ROCHA; VALENTIM et al., 2022).

Embora o uso de ferramentas gráficas torne um usuário com alguma experiência produtivo, um usuário com habilidades avançadas no domínio da aplicação pode ter sua eficiência reduzida (NOVÁK, 2010). Pensando nestas questões, este trabalho propõe a criação de uma linguagem específica de domínio baseada em XML para criar sessões de terapia com o robô EVA, com os seguintes objetivos:

[1] https://gojs.net/latest/index.html

1. possibilitar maior controle na entrada e edição de comandos da linguagem e seus respectivos parâmetros;
2. adicionar abstrações de elementos de programação que visam facilitar a construção de scripts;
3. possibilitar o desenvolvimento de scripts independentes da interface de controle do robô, ou seja, utilizando qualquer editor de texto.

A linguagem XML tem muitas vantagens para o desenvolvimento de linguagens específicas de domínio, destacando-se:

- ser mais legível para não-programadores do que as linguagens de propósito geral;
- em uma linguagem específica de domínio baseada em XML, a gramática pode ser descrita usando uma DTD (*Document Type Definition*) ou XML *Schema*;
- é simples analisar a estrutura XML usando o DOM (*Document Object Model*).

Com o objetivo de facilitar o desenvolvimento de sessões interativas por pessoas com conhecimento técnico em programação, mas ainda mantendo a legibilidade dos códigos dos scripts, esta dissertação propõe a EvaML (ROCHA; MELO et al., 2022), uma linguagem baseada em XML para a especificação de sessões interativas usando a plataforma de robótica *open source* EVA. A EvaML possibilita a criação de scripts de interação para o robô EVA usando apenas um simples editor de texto. Todos os comandos que controlam os elementos de interação multimodal do robô estão presentes na EvaML, entre eles, o componente *Light* (que controla a lâmpada inteligente), os comandos de reconhecimento de voz e o comando *UserEmotion* que possibilita o reconhecimento da expressão facial do usuário através de uma *webcam*. A linguagem também possui elementos para criação e manipulação de variáveis, geração de números aleatórios, controles condicionais usando elementos *switch* e *case* e outros. O *parser* EvaML gera automaticamente um script correspondente no formato JSON que pode ser adicionado ao banco de dados de scrpits do robô e então ser executado por ele.

5.1 Elementos de um Documento EvaML

A Figura 15 apresenta um script EvaML com o seu *elemento raiz* <evaml> e seu atributo name. Este elemento raiz contém os três elementos seguintes: <settings>, <script> e <macros>.

Figura 15: Seções de um documento EvaML

Documentos XML são formados como "árvores" de elementos. Uma árvore XML começa em um elemento raiz e se ramifica da raiz para os elementos filhos. A estrutura em árvore de um documento EvaML pode ser visualizada na Figura 16.

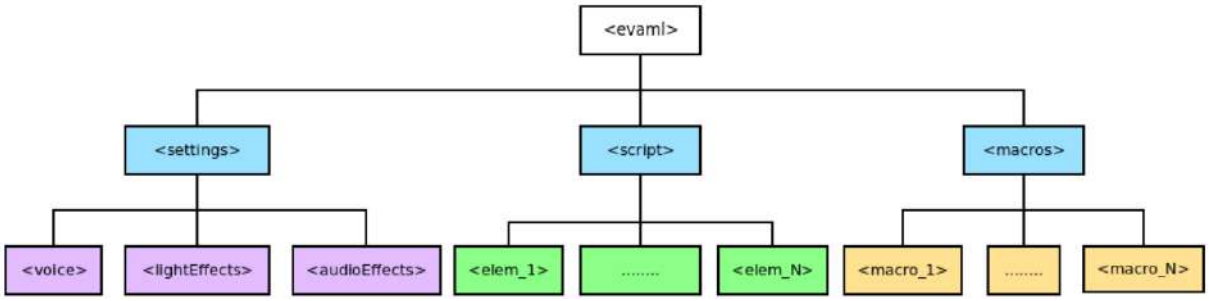

Figura 16: Árvore de elementos de um documento EvaML

A Tabela 3 mostra o elemento raiz do documento EvaML (<evaml>) e os elementos <settings>, <script> e <macros> que representam as seções do documento. São apresentados os conteúdos e atributos de cada elemento. Na coluna *Atributo*, um atributo sublinhado indica que este é obrigatório. Na coluna de *Conteúdo*, os indicadores de ocorrência são usados para indicar a ordem e o número de vezes que um elemento pode ou deve ocorrer. O caractere "**,**" (vírgula) indica que todos os elementos filhos listados devem ser usados na sequência mostrada. O caractere "|" (barra vertical) indica que qualquer elemento pode ocorrer dentro do elemento pai. O sinal de "+" (sinal de adição), por

outro lado, indica que o elemento filho deve aparecer uma ou mais vezes. O caractere "*" (asterisco) indica que o elemento pode ser usado zero ou mais vezes dentro do elemento pai. O caractere "?" (ponto de interrogação) indica que o elemento é opcional, ou seja, o elemento pode não existir ou existe apenas uma ocorrência dele.

Tabela 3: Elementos de um documento EvaML (elemento raiz e elementos principais)

Elemento	**Atributo**	**Conteúdo**
evaml	name	(settings, script, macros?)
settings		(voice | lightEffects? | audioEffects?)
script		(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)
macros		(macro+)

Elemento settings - Define algumas características globais do script. É possível definir como será o timbre da voz e o idioma em que o robô irá se comunicar. Também é possível definir se o código gerado, ao ser executado, irá reproduzir comandos de efeitos de luz, efeitos sonoros ou até mesmo tocar música. Ao configurar esses parâmetros, é possível modificar globalmente o funcionamento do script sem ter que alterar diretamente as definições de seus elementos individuais. A Listagem 5.1 apresenta um trecho de código que exemplifica o elemento <settings>.

```
1 <settings>
2   <voice tone="en-US_AllisonV3Voice" />
3   <lightEffects mode="ON" />
4   <audioEffects mode="ON"/>
5 </settings>
```

Listagem 5.1: Definição de alguns parâmetros do script no elemento **settings**.

Elemento script - Contém a sequência de comandos que o robô deve executar. Pode-se ver alguns deles no trecho de código apresentado na Listagem 5.2. Na linha 2 do script, se encontra o comando <light> que acende a lâmpada inteligente definindo sua cor como azul. Na próxima linha, o comando <talk> faz o robô dizer algo, por exemplo, se apresentar. O comando <wait>, na linha 4, faz com que a execução do script seja pausada por 2000ms (2s). Na próxima linha, o comando <audio> reproduz um arquivo de áudio chamado "*mario-start*". Então, o robô se despede falando "Bye" e desliga a lâmpada inteligente. O atributo block definido como "TRUE" no elemento <audio> (linha 5) faz com que a execução do script siga somente após o término da reprodução do arquivo de

áudio.

```
<script>
  <light state="ON" color="BLUE" />
  <talk>Hi, I am robot EVA</talk>
  <wait duration="2000" />
  <audio source="mario-start" block="TRUE" />
  <talk>Bye</talk>
  <light state="OFF" />
</script>
```

Listagem 5.2: Código de exemplo no elemento **script**.

Elemento macros - É uma das abstrações criadas na linguagem EvaML. Como pode ser visto no próximo trecho de código, mostrado na Listagem 5.3, é possível criar **macros** que podem ser referenciadas dentro do elemento <script>. Uma macro tem o atributo id que serve para identificá-la. Estas macros podem ser usadas dentro da seção <script> usando o comando <useMacro>. O atributo macro do comando <useMacro> faz referência ao elemento <macro> definido dentro de <macros> . Durante o processo de *parsing* do documento EvaML, as macros são expandidas com seu conteúdo na seção <script>. Não há limite para o número de macros criadas, nem para o número de referências a essas macros no script. Como pode ser visto na Tabela 3, o elemento **macros** não é obrigatório.

```
<script>
  <useMacro macro="START" />
</script>
<macros>
  <macro id="START">
    <talk>Hello, I'm robot Eva.</talk>
    <talk>What is your name?</talk>
    <talk>Let us play one more time?</talk>
  </macro>
</macros>
```

Listagem 5.3: Código de exemplo no elemento **macros**.

5.2 Comandos da Linguagem EvaML

Antes de apresentar a descrição de cada comando da linguagem EvaML, a Tabela 4 mostra os elementos da EvaML que representam os comandos da linguagem. Seus atributos e o que cada elemento pode conter também são listados. Para tal representação, será utilizada a mesma notação utilizada na Tabela 3. A descrição de cada símbolo usado pode ser vista na Seção 5.1. Será apresentada a seguir uma breve descrição de cada comando da EvaML, para uma descrição mais detalhada sobre os comandos da linguagem e exemplos de scripts com cada elemento, acesse o manual da EvaML disponível neste link[2].

Tabela 4: Elementos de um documento EvaML (comandos)

Elemento	Atributo	Conteúdo
voice	tone	empty
lightEffects	mode	empty
audioEffects	mode	empty
random	id, min, max,	empty
wait	id, duration	empty
talk	id	text
stop		empty
light	id, state, color	empty
goto	target	empty
motion	id, type	empty
userEmotion	id	empty
evaEmotion	id, emotion	empty
useMacro	macro	empty
listen	id	empty
audio	id, source, block	empty
led	id, animation	empty
counter	id, var, op, value	empty
switch	id, var	(case+, default?)
macro	id	(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | listen* | audio* | led* | counter* | switch*)
case	op, value	(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)
default		(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)

<voice> - Como um elemento de configuração da linguagem, este comando possui apenas um atributo que define, ao mesmo tempo, o timbre de voz (seu gênero) a ser

[2]https://github.com/midiacom/eva-robot/blob/master/EvaML-Reference-Manual/EvaML-Reference-Manual.pdf

usado pelo robô, e o idioma que será utilizado durante o processo de conversão de texto para fala do serviço IBM Watson. Ele possui apenas o atributo tone, e como já foi citado anteriormente, quando um atributo, na coluna de atributos das Tabelas 3 e 4 se encontra sublinhado, seu uso é obrigatório. Como é um comando de configuração, deve estar contido no elemento <settings>. A Tabela 5 exibe uma pequena lista com algumas opções de vozes para o robô.

Tabela 5: Vozes para o serviço de texto para fala do IBM Watson

Código	**Gênero**	**Idioma**
pt-BR_IsabelaV3Voice	feminino	português do Brasil
en-US_AllisonV3Voice	feminino	inglês dos EUA
en-US_EmilyV3Voice	feminino	inglês dos EUA
en-US_HenryV3Voice	masculino	inglês dos EUA
es-LA_SofiaV3Voice	feminino	espanhol latinoamericano
es-ES_EnriqueV3Voice	masculino	espanhol

<random> - Este comando gera um número inteiro aleatório no intervalo fechado [min, max]. O atributo min, representa o limite inferior, e o atributo max, representa o limite superior do número aleatório a ser gerado. O valor gerado pela função aleatória é armazenado em uma região especial da memória do robô, que funciona como um vetor. O caractere **$** acessa o elemento no final desse vetor.

<wait> - Este comando pausa a execução do script pelo intervalo de tempo definido no seu atributo duration. A unidade de tempo usada é o milissegundo.

<talk> - Uma das capacidades de interação multimodal do robô EVA é a fala e através do uso do comando <talk> o robô pode falar um texto especificado. Ao se definir o texto a ser falado é possível utilizar o conteúdo da memória do robô como parte do texto, utilizando-se, no corpo do texto, o caractere **$**, que referencia uma área especial da memória do robô ou a notação #**var** que referencia o conteúdo de uma variável definida pelo usuário. O caractere **$**, por convenção, na linguagem original do EVA, acessa o valor no final do *array* na memória que armazena as respostas retornadas pelos comandos que interagem com o usuário, como os comandos <listen> e <userEmotion>. Como mencionado anteriormente, o *array* na memória do robô também armazena o valor gerado pelo comando <random>. Dentro de um texto, pode-se usar o caractere **$** para incluir a última resposta (retornada por um comando <listen>, por exemplo), **$1** para incluir a primeira resposta, ou **$-1** para incluir a penúltima resposta. A figura 17 mostra esta região especial da memória do EVA que funciona como um *array* e pode ter seus

elementos indexados pelo caractere **$**. O **$1** representa o elemento mais antigo do *array* e **$** representa o elemento mais novo, enquanto o **$-1** representa o predecessor de **$**.

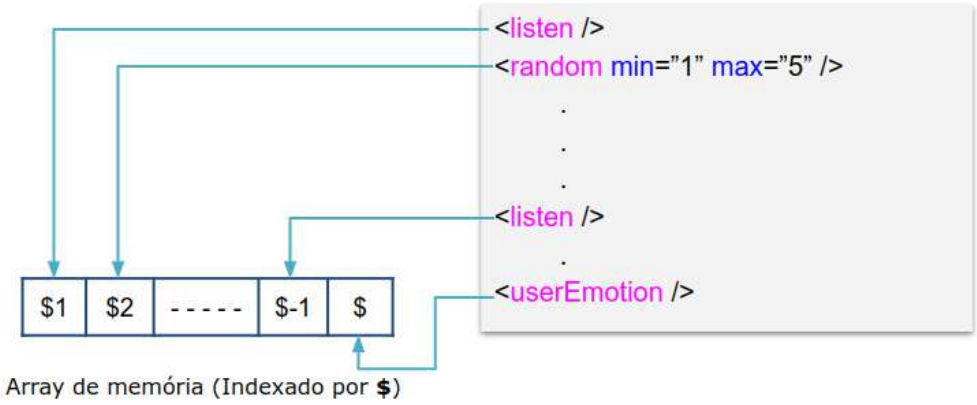

Figura 17: The EVA Robot Memory Array View

<stop> - Este comando é muito simples e como o nome sugere, ele interrompe a execução do script.

<light> - O controle da lâmpada inteligente pode ser feito usando este comando. O seu atributo state pode assumir os valores "ON" e "OFF" e o atributo color, define a cor da lâmpada. Essa cor pode ser indicada usando a representação hexadecimal RGB "#00ff00" (cor verde) ou algumas das cores da lista predefinida: "WHITE", "BLACK", "RED", "PINK", "GREEN", "YELLOW", "BLUE".

<goto> - Este comando altera o fluxo de execução do script para o comando com id referenciado em seu atributo target. O atributo id define o rótulo que será usado como valor no atributo target do comando <goto>.

<motion> - Conforme mencionado na Seção 3.2.5, o robô pode mover sua cabeça e o comando <motion> é responsável por controlar este movimento. O comando possui o atributo type que pode assumir os seguintes valores: "YES", "NO", "CENTER", "LEFT", "RIGHT", "UP", "DOWN", "ANGRY", "2UP", "2DOWN", "2LEFT" e "2RIGHT". Ao se definir type para "YES", faz com que o robô execute um movimento "*sim*" com a cabeça. Usar o valor "NO" faz com que o robô sinalize um "*não*" com sua cabeça. As opções "LEFT", "RIGHT", "UP" e "DOWN" são responsáveis por movimentar a cabeça do robô nas direções correspondentes. O valor "ANGRY" faz com que o robô incline a cabeça para baixo. A opção "CENTER" faz com que a cabeça do robô retorne à sua posição original centralizada. Os valores "2UP", "2DOWN", "2LEFT" e "2RIGHT" executam os movimentos nas direções correspondentes, porém, com o dobro da amplitude do

movimento.

<userEmotion> - O robô é capaz de reconhecer expressões faciais através de uma *webcam*. Ele faz isso usando um módulo externo escrito em Python que roda como um serviço dentro do mesmo dispositivo que roda o software do robô, um Rasperry PI 4. O processo de captura da expressão facial do usuário é o seguinte. O EVA envia uma solicitação de reconhecimento facial para o módulo Python, o módulo recebe a solicitação, ativa a *webcam* e retorna as seguintes expressões[3] como uma string: "NEUTRAL", "ANGRY", "DISGUST", "FEAR", "SURPRISE", "HAPPY" e "SAD". Essa resposta pode ser usada no restante do script sendo acessada através da variável **$**. A Figura 7 ilustra o processo de comunicação do robô com o módulo *Python*.

<evaEmotion> - Este comando controla a apresentação das expressões do olhar do robô na tela de 5.5 polegadas. O seu atributo emotion pode ter os seguintes valores: "HAPPY", "SAD", "ANGRY" e "NEUTRAL". Estas expressões, respectivamente traduzidas para o português, podem ser vistas na Figura 5.

<macro> - Como foi visto na Seção 5.1 uma *macro* define uma sequência de comandos que podem ser referenciados dentro do elemento <script> usando o comando <useMacro>. O elemento <macro> tem apenas o atributo id que é usado para identificá-lo. É importante saber que um elemento <macro> não pode conter a definição de outra *macro* e não pode, por meio de um comando <useMacro>, referenciar outra *macro*.

<useMacro> - Já foi apresentado como as macros podem ser definidas usando o comando <macro>. O comando <useMacro> faz com que o código da macro, referenciado em seu atributo macro, seja expandido pelo código onde o comando <useMacro> está declarado. Este processo de expansão de macro ocorre na primeira etapa do processo de *Parsing* da linguagem EvaML e será explicado na Seção 5.2.1. O comando <useMacro> possui apenas o atributo macro que se refere à macro que será expandida.

<listen> - Conforme mencionado anteriormente, o robô pode reconhecer a voz humana e para isso utiliza o serviço de conversão de fala para texto (STT) da API na nuvem do Google. A captura de áudio é feita usando os 8 microfones da placa Matrix Voice. O áudio capturado é enviado para a nuvem, processado, e então, o texto resultante do processo de STT é retornado para ser utilizado pelo software do robô. A resposta, em formato de string, pode ser referenciada através do caractere **$**.

<audio> - Este comando reproduz um arquivo de áudio contido na pasta "*sonidos*"[4].

[3] O simulador EvaSIM trabalha com cinco expressões que serão apresentadas na Seção 6.1

[4] A pasta "sonidos" está localizada no diretório raiz do aplicativo do robô (no Raspberry Pi) ou no

O seu atributo source deve indicar apenas o nome do arquivo, sem o caminho da pasta e sem a sua extensão. O EVA só reproduz arquivos no formato *wav*. O seu atributo block pode ter os seguintes valores, "TRUE" ou "FALSE". Esses valores definem se a execução do arquivo de áudio deve bloquear a execução do script, ou seja, o comando que vem após o comando <audio> só será executado após o término da reprodução do mesmo.

<led> - Este comando controla a animação[5] com os LEDs no peito do robô EVA. O seu atributo animation pode assumir os seguintes valores: "HAPPY" (verde), "SAD" (azul), "ANGRY" (vermelho), "STOP" (sem cor/desligado), "SPEAK" (azul), "LISTEN" (verde) e "SURPRISE" (amarelo).

<counter> - A criação de uma variável na memória do EVA é feita através deste comando. Com ele é possível criar, inicializar e realizar operações matemáticas sobre essas variáveis. O atributo var define o nome variável, enquanto o atributo op define o tipo de operação, podendo assumir os seguintes valores: "=" (**atribuição**), "+" (**adição**), "*" (**multiplicação**), "/" (**divisão**) e "%" (**módulo**). O atributo value define o valor a ser atribuído à variável, no caso de uma atribuição, ou o valor a ser usado nas operações aritiméticas. Para referenciar o valor de uma variável criada pelo usuário, diferentemente do caractere **$**, é necessário utilizar o caractere "#" antes do nome da variável. Essa notação deve ser seguida quando uma variável definida pelo usuário for utilizada dentro de um texto no elemento <talk> e no atributo value dos comandos <case>. No atributo var de um comando <switch>, o nome da variável é utlizado sem o caractere "#".

<switch> - Este comando define a variável que será comparada com os valores definidos nos comandos <case>. Para isso, seu atributo var deve conter o nome da variável a ser comparada, por exemplo: <switch var="$"> ou <switch var="x">. O atributo var do comando <switch> determina com qual variável as comparações serão feitas nos comandos <case> e pode assumir os valores **$** ou qualquer outro nome de variável que tenha sido declarada anteriormente.

<case> - O comando <case> especifica uma sequência de comandos que serão executados se a condição definida em seus atributos for verdadeira. O comando <case> possui o atributo op, que define o tipo de comparação ou operador lógico que será processado, e o atributo value, que contém o valor a ser comparado com a variável definida no atributo var do comando <switch>. O conteúdo de value pode ser uma constante (um número ou uma string), o caractere **$** ou outra variável usada no script. É importante lembrar que, no atri-

diretório raiz do simulador do EVA.

[5] Algumas animações dos LEDs estão associadas a outros comandos do robô e são ativadas automaticamente com eles, como por exemplo os comandos <talk> e <listen>.

buto value, para se referir ao valor de uma variável declarada anteriormente, deve-se usar o caractere "#" antes do nome da variável. O atributo op pode assumir três valores, que determinam três tipos diferentes de comparação: "**exact**" (exata), "**contain**" (contém) e "**math**" (matemática).

A comparação do tipo "**exact**" é usada para comparar o conteúdo do atributo value do comando <case> com o valor referenciado por **$**. O valor especificado no atributo value será comparado exatamente com o valor referenciado por **$**, caractere por caractere. Esta comparação não diferencia letras maiúsculas de minúsculas.

A comparação do tipo "**contain**" é usada para comparar o conteúdo do atributo value do comando <case> com o valor referenciado por **$**. Mas neste caso, a comparação será verdadeira se o valor contido no atributo value do comando <case> for igual ao valor em **$** ou uma substring do mesmo. Esta comparação não diferencia letras maiúsculas de minúsculas.

As comparações de tipo "**exact**" e "**contain**" só devem ser usadas quando o atributo var do comando <switch> for "**$**", caso contrário, uma mensagem de erro será lançada pelo *parser*. Essas comparações são baseadas em strings e não em números.

Para realizar uma comparação matemática no script, deve-se utilizar os símbolos de igualdade, desigualdade, maior que, menor que, etc. Na EvaML, devido ao conflito gerado pelo uso do caractere "<" no processo de *parsing*, os operadores de comparação do tipo "**math**" (matemático) são os seguintes: "**eq**", "**lt**", "**gt**", "**lte**", "**gte**" e "**ne**", que representam, respectivamente, *igual*, *menor que*, *maior que*, *menor ou igual*, *maior ou igual* e *diferente*. O uso da comparação matemática assume que o conteúdo do atributo value do comando <case> é um número. Essas regras podem ser melhor compreendidas com o auxílio da Tabela 6.

<default> - Conforme mostrado na Tabela 4, o comando <default> compõe o grupo de elementos (comandos) que são filhos do elemento <switch> e seu uso não é obrigatório. Se usado, deve vir como o último filho de um elemento <switch>, após todos os comandos <case>. O bloco de comandos contido em <default> será executado se nenhuma condição avaliada nos comandos <case> for verdadeira.

5.2.1 Processo de Parsing de um Script EvaML

O processo de *parsing* de um documento EvaML ocorre em quatro etapas, que são executadas por quatro módulos, e podem ser vistas na Figura 18. Essas etapas serão explicadas

Tabela 6: Tipos de comparação e operações lógicas do comando <**case**>

Atributo var do <switch>	Tipo de comparação (atributo op do <case>)	Tipo do valor no atributo value do <case>	Descrição
$	**exact**	string	Comparação exata entre duas strings. Não é case sensitive.
	contain	string	Verifica se $ contém a string em value. Não é case sensitive.
$ ou variável	**math** ("eq", "lt", "gt", "lte", "gte", "ne")	Uma constante numérica, $ ou #**var**.	Comparação numérica entre $ ou outra variável, com um valor constante, o $, ou outra variável contida no atributo value de <case>. Somente trabalha com números inteiros.

em detalhes a seguir.

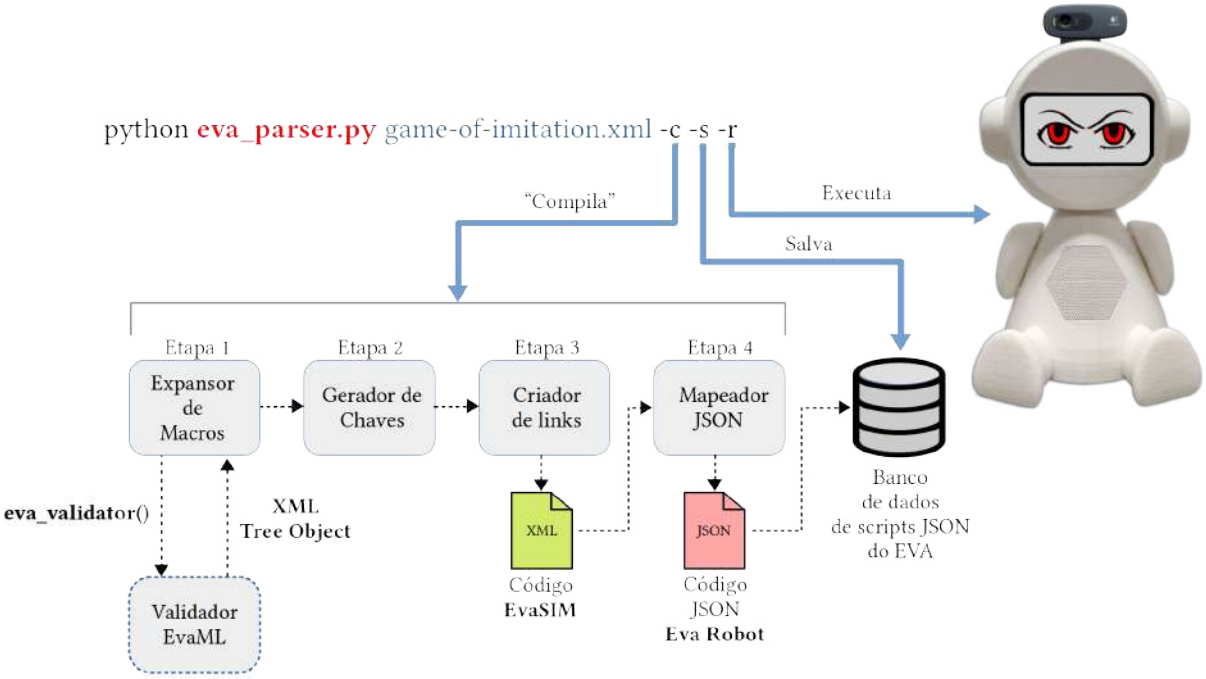

Figura 18: Processo de *parsing* e exemplo de linha de comando

A primeira etapa consiste no processo de expansão das macros referenciadas dentro do elemento <script>. Nesta etapa, os comandos contidos nas macros são expandidos dentro do script. Esta expansão é feita nos locais do script onde as macros são referenciadas, através do comando <useMacro>.

O script que o robô EVA executa é representado como um grafo com nós e arestas (links) codificados em JSON. Os nós são os comandos da linguagem (ou ações que o robô pode executar) e os links determinam o fluxo de execução do script. Esta segunda etapa

do *parser* executa a tarefa de criação das chaves identificadoras dos nós do grafo. Elas vão identificar cada nó, de maneira unívoca. Esses nós são conectados por meio de links que usam essas chaves para referenciá-los.

A terceira etapa é responsável por analisar o fluxo de execução do script. É nessa etapa que o *parser* identifica o fluxo de execução do script, com seus possíveis desvios, saltos e repetições, gerando os links que formam o grafo de execução do script. Como saída desta etapa, tem-se o arquivo XML destinado ao simulador do robô EVA, chamado de EvaSIM, que será apresentado em detalhes no Capítulo 6.

Na quarta etapa, o *parser* mapeia os comandos e links resultantes da etapa anterior usando os templates JSON que representam as estruturas dos comandos do robô. O *parser* identifica o elemento XML com seus atributos, processando cada comando de maneira específica. Como resultado desta última etapa, é gerado um arquivo JSON que pode ser enviado para o banco de dados de scripts do robô e executado.

Parâmetros do *parser* - O *parser* pode ser executado usando três parâmetros que são apresentados na Tabela 7. Usando apenas o parâmetro "**-c**", o analisador gera dois arquivos, um no formato EvaSIM e outro no formato JSON, que é o código a ser executado no robô. Utilizando o parâmetro "**-s**", o arquivo JSON gerado é inserido diretamente no banco de dados do EVA. E finalmente, usando o parâmetro "**-r**", faz com que o robô inicie imediatamente o script. A Figura 18 mostra um exemplo da linha de comando usando o *Parser* com seus parâmetros. Os parâmetros "**-s**" e "**-r**" só tem funcionalidade se forem executados de dentro do diretório da aplicação do robô no Raspberry Pi.

Tabela 7: Parâmetros do *Parser* EvaML

Parâmetro	Definição
-c	Interpreta o script EvaML gerando um arquivo XML no formato do simulador EvaSIM e um arquivo JSON
-s	Salva/Insere o arquivo JSON gerado no banco de dados do robô.
-r	Faz com que o robô EVA rode o script imediatamente.

5.2.2 Processo de Validação de um Script EvaML

A linguagem EvaML foi especificada utilizando-se uma especificação em XML Schema (SCHEMA, 2004) que pode se encontrada no Apêndice A. O propósito de um esquema é definir e descrever uma classe de documentos XML usando essas construções para restringir e documentar o significado, uso e relacionamentos de suas partes constituintes: tipos de dados, elementos e seus conteúdos, atributos e seus valores. Um XML Schema fornece

um meio para definir a estrutura, conteúdo e semântica de documentos XML.

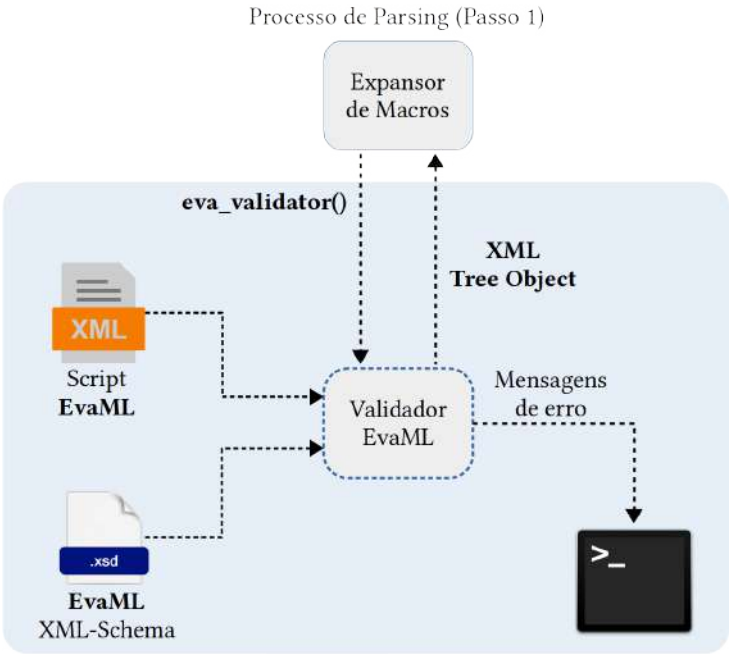

Figura 19: Processo de validação de um documento EvaML usando XML Schema

O processo de validação do script EvaML ocorre em paralelo e é iniciado pelo módulo *Expansor de Macros* que usa o módulo auxiliar *Validador EvaML*. A Figura 19 apresenta os elementos envolvidos neste processo. A etapa 1 do *parser* chama a função de validação (*eva_validator*) no módulo auxiliar passando como parâmetro o nome do script a ser validado. Dentro do módulo auxiliar está indicada a URL do arquivo XML Schema que especifica a linguagem EvaML. O módulo auxiliar é responsável por exibir as mensagens do validador XML no terminal do sistema. Durante o processo de validação podem ocorrer três casos distintos:

1. O script EvaML é bem-formado e também válido, o módulo auxiliar retorna um objeto *XML Tree* para o módulo de expansão de macros dando prosseguimento ao processo de *parsing*.

2. O script não é bem-formado e o validador imprime uma mensagem sinalizando o problema no script indicando a possível linha do código onde o problema foi identificado. O módulo auxiliar retorna um objeto nulo para o módulo de expansão de macros, que interrompe o processo de *parsing*.

3. O script é bem-formado, porém, não é válido, ou seja, não segue a gramática definida no arquivo XML Schema da linguagem. O módulo de validação mostra no terminal

todos os problemas identificados e também retorna um objeto nulo para o módulo de expansão de macros, interrompendo o processo de *parsing* do script.

O processo de validação do script EvaML usando o arquivo XML Schema da linguagem é capaz de indicar várias falhas no script. Ele pode indicar a falta de um elemento <voice> na seção <settings>, pode acusar a falta de algum atributo que tenha seu uso definido como obrigatório, pode indicar a ocorrência de duplicidade em atributos definidos como únicos, como no caso do atributo id, e pode verificar a corretude de muitos outros critérios definidos na especificação XML Schema.

Este capítulo apresentou a linguagem EvaML, proposta por esta dissertação. O próximo capítulo apresenta o simulador EvaSIM, capaz de executar scripts desenvolvidos para o robô EVA utilizando a linguagem de programação visual original do robô ou utilizando a linguagem EvaML.

6 Simulador EvaSIM

Embora o EVA use hardware de baixo custo e software de código aberto, nem sempre é prático ter um robô físico à mão, principalmente durante o projeto iterativo de terapias. Portanto, é difícil testar uma sessão de terapia robótica se você não tiver o robô físico montado com todos os seus componentes de hardware e também é difícil treinar pessoas (técnicos ou pesquisadores/entusiastas de IHR), ensinando-os a programar um aplicativo ou sessão de terapia interativa usando a ferramenta de programação do robô. Assim, estas são as motivações para desenvolver uma ferramenta que permita testar um script desenvolvido na VPL ou em EvaML para o robô EVA.

Com o objetivo de fornecer um ambiente de teste para os scripts EvaML, foi desenvolvido um software que simula o robô EVA, chamado de EvaSIM (ROCHA, M. M. da et al., 2022; ROCHA, M. et al., 2022). Um requisito importante que foi levado em consideração no desenvolvimento do EvaSIM é que ele fosse portável, por esse motivo, a linguagem Python foi escolhida para o seu desenvolvimento.

A interface gráfica do EvaSIM foi desenvolvida utilizando-se *Tkinter*[1], uma das bibliotecas mais populares para criar interfaces gráficas de usuário (GUI - *Graphical User Interface*) em Python. O pacote *Tkinter* é uma fina camada orientada a objetos sobre a biblioteca *Tcl/Tk*.

A Figura 20 mostra a interface gráfica de usuário do EvaSIM. O layout da aplicação foi definido usando três objetos do tipo *Frame* da biblioteca *Tkinter*. O primeiro *Frame*, à esquerda da janela da aplicação, contém um objeto *Canvas* onde são desenhadas as figuras que representam os componentes do robô. O segundo *Frame*, que fica no centro da janela, contém dois elementos importantes da aplicação. Em sua parte superior, há um grupo de botões que são responsáveis por controlar o EvaSIM. Abaixo do grupo de botões de controle, há um objeto *Text* que é usado para criar um emulador de terminal para o EvaSIM. Neste terminal são apresentadas diversas ações e estados do robô simulado. O

[1] https://docs.python.org/3/library/tkinter.html

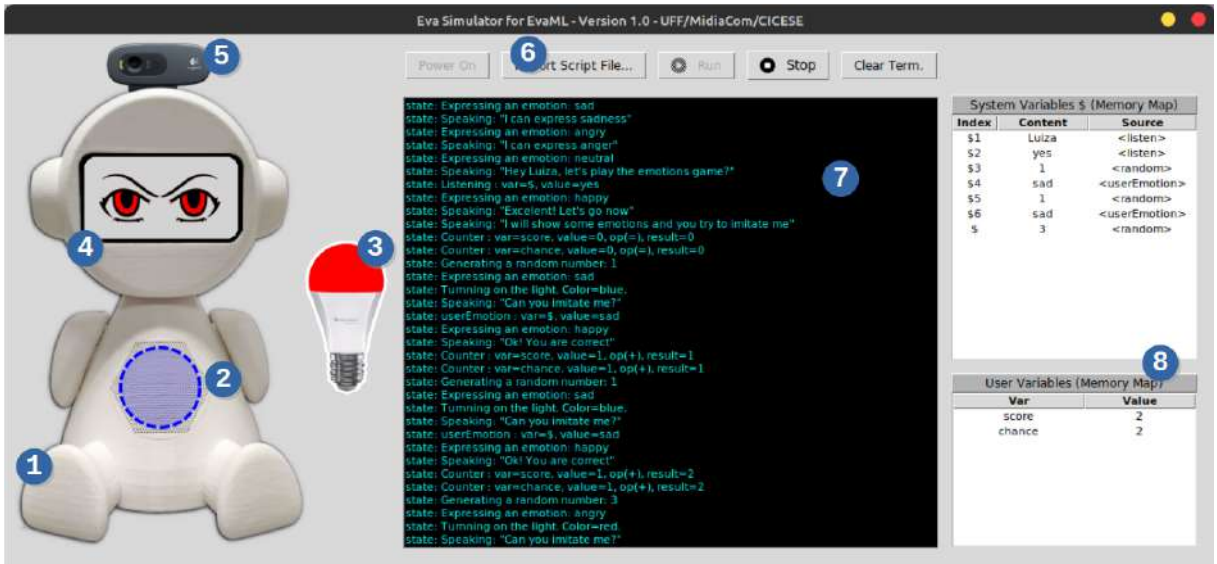

Figura 20: Elementos da interface gráfica de usuário do EvaSIM

terceiro *Frame* à direita contém duas tabelas que exibem os valores das variáveis do sistema e do usuário durante a execução do script. Uma descrição de cada elemento é apresentada a seguir, usando os números indicados dentro dos círculos azuis como referência:

1. Corpo do robô EVA impresso em 3D.

2. Representa os LEDs RGB que fazem parte da placa Matrix Voice, localizada no tórax do robô. Esse elemento, como no robô físico, pode assumir várias cores e essas cores variam de acordo com a ação que o robô está realizando no momento.

3. É a representação gráfica da lâmpada inteligente que pode assumir os estados ligado ou desligado, podendo também apresentar as cores definidas no script do robô.

4. Representa a tela de 5,5 polegadas do robô. Este elemento gráfico pode mostrar as expressões do olhar do EVA, apresentando as quatro expressões disponíveis no robô físico.

5. Apresenta a webcam integrada ao robô.

6. O conjunto de botões que controlam o simulador. Da esquerda para a direita, o primeiro botão é aquele que inicia o simulador do robô fazendo com que ele entre em modo *stand by*, aguardando o carregamento de um script. O segundo botão abre uma janela de diálogo de abertura de arquivo, que permite carregar um script. O terceiro botão executa o script carregado. Este botão só é habilitado após o carregamento de um script. O próximo botão interrompe um script em execução e, por último, o botão que limpa o emulador de terminal.

7. Apresenta a emulação de um terminal onde são apresentadas algumas informações importantes, como as ações que estão sendo realizadas pelo robô, detalhes das operações com variáveis, cores e estados da lâmpada inteligente, o texto que o robô está falando, algumas mensagens de alerta e possíveis mensagens de erro no script.

8. Apresenta as tabelas do mapa de memória do simulador. Essas duas tabelas destinam-se a mostrar dinamicamente os valores das variáveis do sistema e do usuário durante a execução do script. A tabela superior mostra os valores das variáveis do sistema que armazenam as respostas obtidas nos processos de interação com usuário, como captura de voz e reconhecimento de expressões faciais. Esse conjunto de variáveis também contém os valores gerados pelo comando de geração de números aleatórios da VPL. Essas variáveis do sistema são indexadas usando o caractere "$". Como a memória do robô está repleta de valores de diferentes origens, a tabela superior possui, além das colunas de índice e conteúdo, uma coluna extra que mostra a origem do valor dessa variável. A segunda tabela apresenta as variáveis criadas pelo desenvolvedor do script, com seus nomes e seus respectivos valores.

6.1 Simulando o Reconhecimento de Expressões Faciais

Como visto na Figura 20, item (9), o EvaSIM simula o processo de reconhecimento facial. Esta função foi implementada utilizando-se uma janela com expressões faciais representadas por emojis[2]. No simulador, as respostas que podem ser fornecidas através da janela que representa as expressões faciais do usuário são: "NEUTRAL", "HAPPY", "ANGRY", "SAD" e "SURPRISED". A execução do comando <userEmotion>, responsável por capturar a expressão do usuário através da webcam, abre uma janela com um conjunto de expressões faciais. O usuário, através do mouse, pode indicar sua expressão facial. Esta resposta é processada pelo simulador da mesma forma que o robô físico. Um pequeno vídeo mostrando a simulação do *Jogo da Imitação* (ROCHA; VALENTIM et al., 2022) no EvaSIM pode ser encontrado nesta URL: https://youtu.be/OfGelKZIA9c.

6.2 Simulando o Reconhecimento de Voz

O EVA pode se comunicar com o usuário por meio de interação por voz, capturando o áudio das respostas e transformando-o em texto usando a API do Google que fica na nu-

[2]É um pictograma ou ideograma, ou seja, uma imagem que transmite a ideia de uma palavra ou frase completa

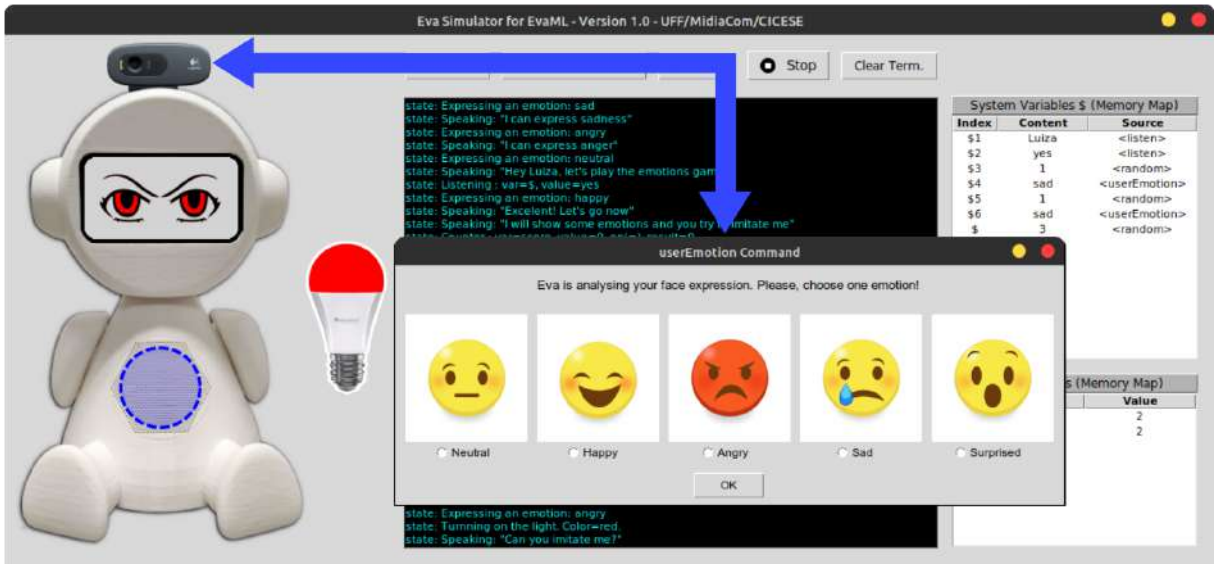

Figura 21: Simulando o reconhecimento de expressões faciais

vem. Para facilitar esse processo, dentro do simulador, esse tipo de interação multimodal foi representado por uma caixa de texto, onde o usuário pode responder usando texto escrito ao invés de fala. A Figura 22 mostra a simulação do processo de captura de voz do robô.

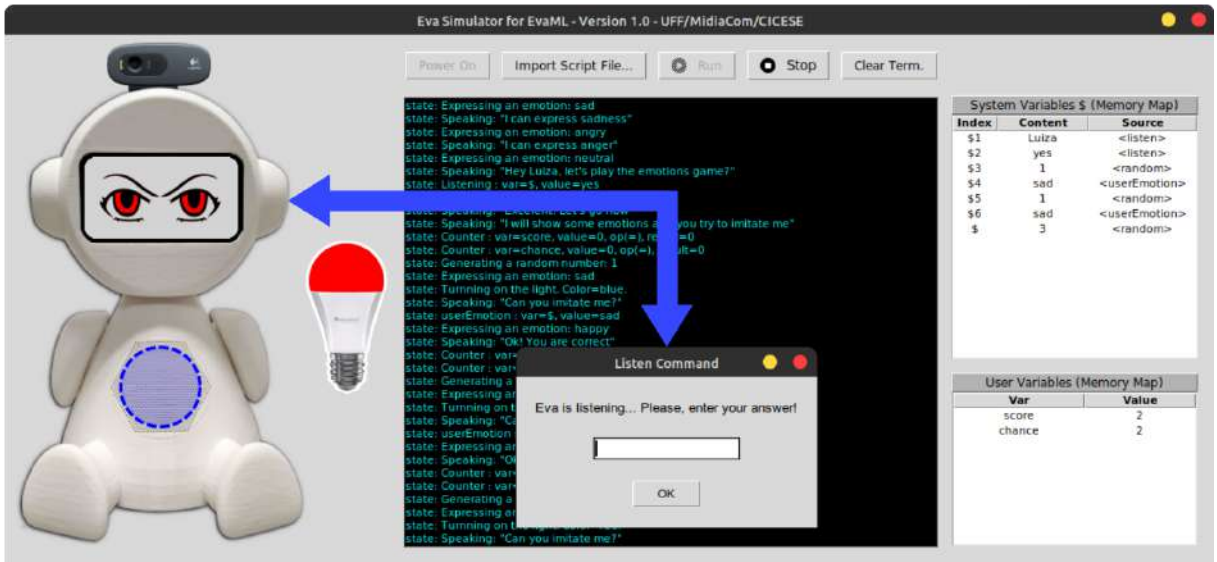

Figura 22: Simulando o reconhecimento de voz

6.3 Simulando a Movimentação da Cabeça do Robô

O EvaSIM foi projetado para ser uma ferramenta leve e que exigisse baixo poder computacional para ser executada. Portanto, nenhuma animação sofisticada foi implementada

nesta versão 2D do simulador. O robô físico, conforme apresentado na Seção 3.2.5, pode mover sua cabeça. Este movimento utilizado em conjunto com outros elementos pode aumentar a expressividade do robô. Ao executar um script para o robô e encontrar um elemento <motion>, o EvaSIM utiliza o terminal para indicar que um movimento está sendo executado, mostrando também o tipo de movimento realizado. A Figura 23 mostra um exemplo da mensagem no terminal indicando a execução do elemento <motion> e o valor do atributo type.

```
state: Matrix Leds. Animation=STOP
state: Moving the head! Movement type: CENTER
state: Pausing. Duration=1000 ms
state: Matrix Leds. Animation=SURPRISE
state: Moving the head! Movement type: 2RIGHT
state: End of script.
```

Figura 23: Indicando o movimento da cabeça do robô

A Figura 24 apresenta com mais detalhes a emulação de um terminal onde são apresentadas algumas informações importantes sobre a execução do script. Nesse exemplo, pode-se ver a seleção de voz, o estado e a cor da lâmpada inteligente sendo definidos, os textos sendo falados, a captura do nome de usuário através do comando <listen> e a manipulação da variável **x**.

```
==============================================================================
                          Eva Simulator for EvaML
                 Version 1.0 - UFF/MidiaCom/CICESE [2021]
==============================================================================
state: Starting the script: script01
state: Selected Voice: en-US_AllisonV3Voice
state: Turnning on the light. Color=white.
state: Speaking: "Hi, I'm the robot Eva and I'm a socially assistive robot"
state: Speaking: "What is your name?"
state: Listening : var=$, value=Marcelo
state: Turnning on the light. Color=green.
state: Speaking: "Hi Marcelo, how are you? Let's start the script"
state: Counter : var=x, value=0, op(=), result=0
state: Counter : var=x, value=1, op(+), result=1
state: Speaking: "The value of x is 1"
state: Speaking: "Hey Marcelo, I will terminate. Bye"
state: Turnning off the light.
state: End of script.
```

Figura 24: Emulador de terminal

6.4 Executando arquivos JSON

A Figura 25 mostra a arquitetura geral do EvaSIM dividida em três módulos principais. O módulo Parser recebe como entrada um código do editor VPL (em formato JSON) fazendo

o processo de parsing do script do robô. Esse processo gera uma nova representação da estrutura do código VPL, agora transformada em um arquivo XML. O arquivo XML é passado para o módulo Execution, que é responsável por percorrer a estrutura em árvore XML, identificar os nós que representam os comandos da linguagem e emitir uma série de sinais de diversos tipos que são enviados ao próximo módulo. O último módulo da arquitetura do simulador é o módulo Graphical User Interface (GUI) que controla os elementos gráficos do simulador.

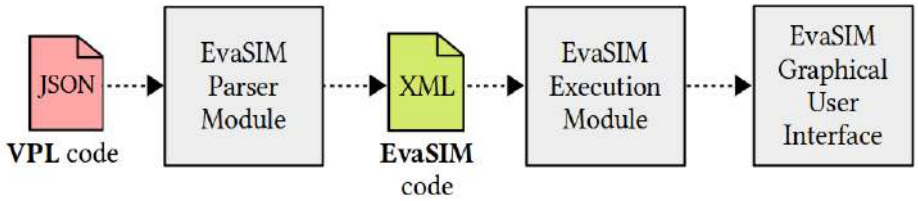

Figura 25: Arquitetura geral do EvaSIM

6.4.1 O Funcionamento do EvaSIM

O EvaSIM usa alguns módulos extras que podem ser instalados facilmente usando o gerenciador de pacotes Python *pip*, que são: o módulo *Ibm-watson* para acessar a API Text-To-Speech do *IBM Watson*, o módulo *Playsound* que permite a reprodução de arquivos de áudio, e o módulo *Tkinter* usado para criar a interface gráfico de usuário do simulador.

Tabela 8: Comandos da VPL simulados pelo EvaSIM

Comando da VPL	**Parâmetros**
Voice	*tone*
Random	*min, max*
Wait	*duration*
Talk	*text*
Light	*state, color*
evaEmotion	*emotion*
Audio	*source, block*
Led	*animation*
Counter	*counter_name, value, operation*
Condition	*condition, comp_type*
Motion	*type*
Listen	*none*
userEmotion	*none*

A Tabela 8 mostra os comandos da VPL que podem ser simulados pelo EvaSIM e

seus parâmetros. A seguir, uma breve descrição de cada um deles. O comando *Voice* seleciona o timbre de voz e o idioma que será usado no script. O comando *Random* gera números aleatórios em um intervalo definido pelo usuário. O comando *Wait* faz com que o script seja pausado por um determinado período de tempo. O comando *Talk* faz com que o texto seja transformado em áudio e então falado pelo simulador. O comando *Light* controla a lâmpada inteligente. O comando *evaEmotion* determina a expressão na face do robô. O comando *Audio* pode reproduzir sons e músicas especificados em seu atributo *source*. O comando *Led* controla a simulação dos LEDs da placa que fica no peito do robô. O comando *Counter* permite criar variáveis e realizar operações matemáticas sobre elas. O comando *Condition* avalia o resultado de uma expressão lógica e pode alterar o fluxo de execução do script. O comando *userEmotion* utiliza uma webcam para identificar a expressão facial do usuário. O comando *Listen* faz com que o robô capture a fala do usuário convertendo-a em texto e, por fim, o comando *Motion* move a cabeça do robô conforme o tipo de movimento indicado no parâmetro *type*. A simulação dos três últimos comandos citados foi apresentada com detalhes nas Seções 6.1, 6.2 e 6.3.

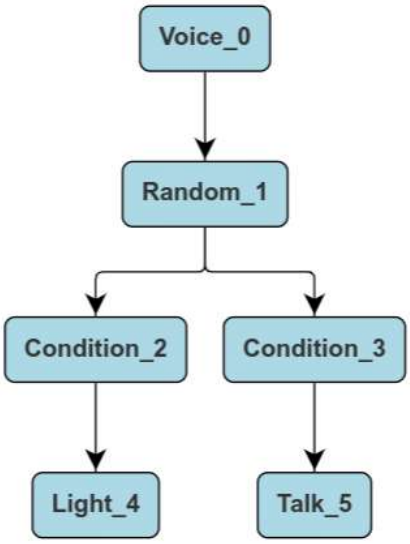

Figura 26: Um exemplo de script VPL

Para entender todo o processo de interpretação de um script feito na VPL e o processo de simulação como um todo, é necessário primeiro entender como é feita a codificação de um script gráfico. Conforme mostrado na Figura 26, a representação de um script na VPL é feita através de um grafo. Os nós do grafo representam as capacidades do robô (elementos de linguagem) e as arestas (que daqui em diante serão chamadas de links) indicam o fluxo de execução do script. A codificação desse script em modo gráfico é feita através de um arquivo no formato JSON. A Figura 27 mostra alguns trechos de código que representam: (a) a estrutura de um script vazio, (b) um exemplo de um nó *Condition*

e (c) um exemplo de um link com seus atributos *from* e *to*.


```
{
  "_id": "c82c9a0b-9926-4e12-9235-f0cee62d7e65",
  "name": "script01",
  "data": {
    "node": [],
    "link": []
  }
}
```


(a)


```
{
  "key": 1646387260007,
  "name": "Condition_2",
  "type": "if",
  "color": "lightblue",
  "isGroup": false,
  "text": "$ == 2",
  "opt": 5,
  "__gohashid": 3261
}
```


(b)


```
{
  "from": 1646387240212,
  "to": 1646387248220,
  "__gohashid": 3145
}]
```


(c)

Figura 27: (a) Estrutura de um script VPL vazio. (b) Um exemplo de um nó do tipo *Condition*. (c) Um exemplo de uma *aresta* (link)

A Figura 28 apresenta detalhadamente o processamento e a simulação do código VPL no EvaSIM. Conforme discutido anteriormente e ilustrado na Figura 25, há duas etapas principais envolvidas na execução de um script VPL antes de enviar os comandos para a interface gráfica do usuário. A primeira etapa, chamada de *JSON Data Mapping*, executa dentro do módulo *parser* do EvaSIM (*EvaSIM Parser Module*) e analisa o arquivo JSON que contém a representação do script VPL. A segunda etapa, chamada de *EvaSIM XML Code Processing*, executa dentro do módulo de execução do EvaSIM (*EvaSIM Execution Module*) e é responsável por rodar o script.

6.4.2 Mapeamento de dados JSON *(JSON Data Mapping)*

Após o carregamento do arquivo JSON, um processo de mapeamento de dados é iniciado. Conforme mostrado na Figura 27(a), um script VPL codificado é composto por três atributos, o atributo *_id* que é um código que identifica exclusivamente o script no banco de dados do robô, o atributo *name* e um atributo chamado *data* que contém dois arrays, um array de objetos node e outro array de objetos link. Como mencionado anteriormente, um nó representa um comando da linguagem do EVA, e como pode ser visto na Figura 27(b), um nó possui um conjunto de pares *chave* e *valor* que representam, ao mesmo tempo, o elemento gráfico da VPL, como os atributos *color* e *name*, e as informações relacionadas ao comando de linguagem que está sendo representado, como o *type*, *key* e *text* atributos. Cada elemento VPL tem seu conjunto específico de atributos. O elemento

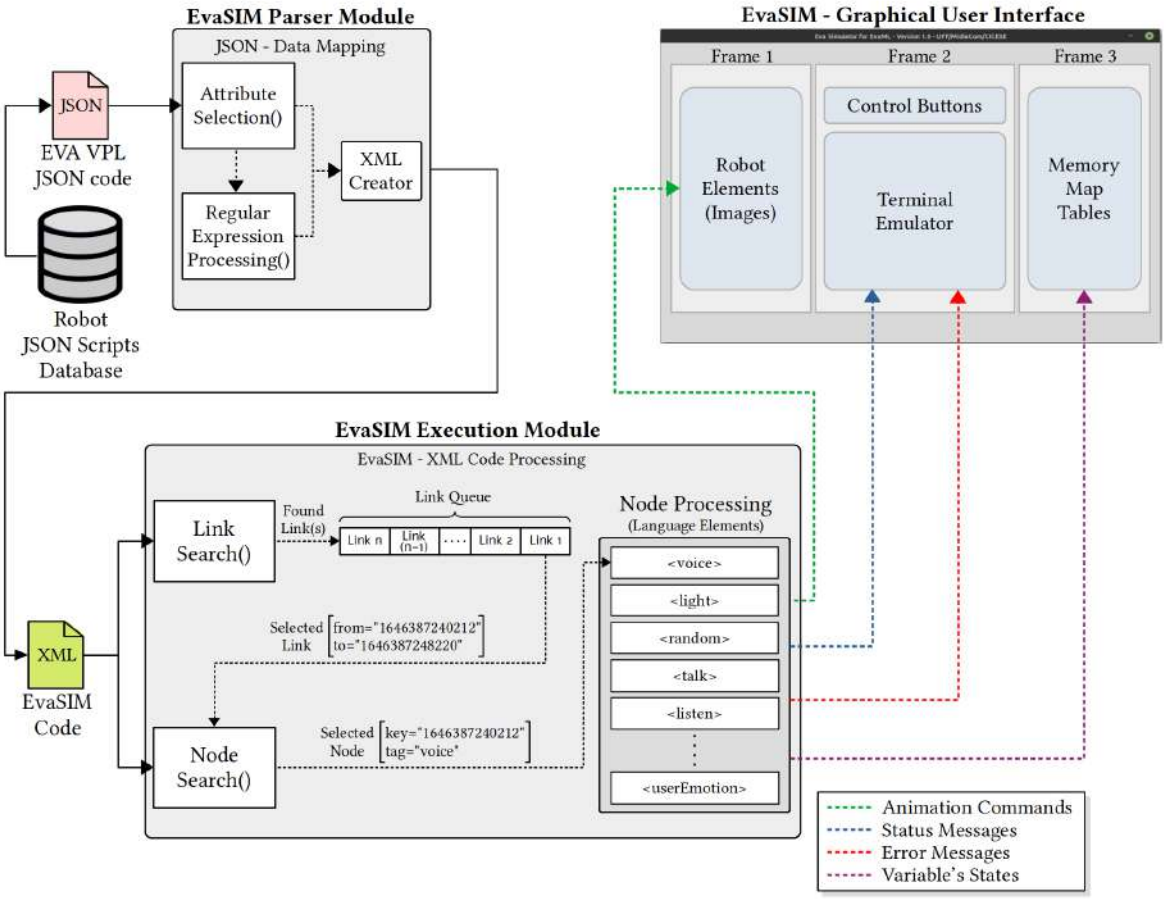

Figura 28: Processamento e Simulação de um script da VPL no EvaSIM

Voice possui um atributo que permite ao usuário definir o timbre da voz e o idioma que a aplicação reconhecerá nos processos texto para fala e fala para texto. O elemento *Light*, por exemplo, possui um atributo que determina sua cor e outro que determina se a lâmpada deve ser ligada ou desligada. O processo de seleção de atributos lê a lista de nós do arquivo JSON, selecionando para cada tipo de nó os atributos relevantes para sua execução no simulador. Cada nó tem um tipo que é indicado pelo atributo *type*. Alguns atributos do código do robô vêm na forma de uma string composta, ou seja, há mais de um atributo dentro da string. Esses atributos compostos são passados para a parte do código que usa expressões regulares para decompor a string em atributos individuais. Como exemplo, o atributo *text* do comando *Condition*, representa a expressão booleana "$ == 2"contendo dois operandos e um operador relacional. A expressão, após ser enviada para a função de decomposição, retorna três atributos, um atributo chamado *var*, que contém o "$", outro atributo chamado *valor*, que contém o número "2", e outro atributo chamado *op* que contém o operador relacional de igualdade.

A saída da etapa *JSON Data Mapping* é um arquivo XML usado como entrada para a etapa seguinte. É nessa etapa que se executa efetivamente o script EVA, conforme ilustrado na Figura 28.

```
1  <?xml version='1.0' encoding='utf8'?>
2  <evaml name="script01">
3     <script>
4        <voice key="1646387240212" tone="en-US_AllisonV3Voice"/>
5        <random key="1646387248220" min="1" max="2"/>
6        <case key="1646387260007" op="eq" value="1" var="$"/>
7        <case key="1646387268473" op="eq" value="2" var="$"/>
8        <light key="1646387289446" state="on" color="#00b0ff"/>
9        <talk key="1646387340575">Hello!</talk>
10    </script>
11    <links>
12       <link from="1646387240212" to="1646387248220"/>
13       <link from="1646387248220" to="1646387260007"/>
14       <link from="1646387248220" to="1646387268473"/>
15       <link from="1646387260007" to="1646387289446"/>
16       <link from="1646387268473" to="1646387340575"/>
17    </links>
18 </evaml>
```

Listagem 6.1: Código XML resultante da etapa de mapeamento de dados JSON

6.4.3 Processamento de Código XML EvaSIM *(EvaSIM XML Code Processing)*

Esta etapa de processamento toma como entrada um arquivo XML resultante da etapa anterior. O código XML formatado para EvaSIM pode ser visto na Listagem 6.1. A seção <script> do código XML contém como elementos filhos os nós que representam os comandos da linguagem do robô. A seção <links> contém os links que determinam o fluxo de execução do script. Como pode ser visto na Listagem 6.1, quando comparado à Figura 27, o mapeamento e conversão do arquivo JSON para XML trouxe mais legibilidade ao código uma vez que as propriedades gráficas dos elementos da VPL foram descartadas e os elementos compostos, no caso da expressão lógica do elemento *Condition*, foram decompostos em três atributos XML: var="$", op="eq"e valor="2". Os operadores relacionais "==", ">", "<", ">=", "<="e "!="correspondem aos valores do atributo *op* "eq", "gt", "lt", "gte", "lte", "ne"respectivamente.

A etapa de processamento do Código XML do EvaSIM pesquisa e seleciona um ou mais links na lista de links e os insere na fila de links (*Link Queue*). Essa fila de links é então processada recursivamente até ficar vazia. O processamento da fila começa com a remoção do primeiro link. O valor da chave contida no atributo *from* do link é passado para a função que realiza uma pesquisa na lista de nós (comandos VPL) na seção <script> do código XML. A função procura um nó por seu atributo *key*. Após o nó ser encontrado, ele é enviado para a função responsável pelo processamento dos nós. O tipo de nó (elemento de linguagem) é obtido através do nome do elemento XML correspondente. O comando é então executado e pode se comunicar com os elementos da interface gráfica do EvaSIM enviando alguns tipos de informações, tais como: comandos de animação, que geram algum tipo de animação gráfica sobre a figura do robô e seus componentes, mensagens de status e mensagens de erro, que são informações enviadas para o emulador de terminal do EvaSIM, e valores de variáveis, que são informações relacionadas às variáveis do módulo de memória do robô e que são enviadas para as tabelas de mapas de memória na janela do EvaSIM. A execução do script termina quando o valor da chave contida no atributo *to* de um link, após o processo de busca do link, não corresponde a nenhum valor da chave do atributo *from* de um link na lista de links. Isso indica que o elemento referenciado pela chave no atributo *to* do link que está sendo processado corresponde a um nó folha

da árvore XML. A chave contida no atributo *to* é passada para a função de busca de nós. Em seguida, o nó é encontrado e processado, encerrando o script.

O código do EvaSIM é *open source* e está disponível através deste link[3].

Este capítulo apresentou o simulador EvaSIM, que permite a execução de scripts desenvolvidos para o robô EVA tanto na VPL (em sua codificação no formato JSON) quanto em EvaML. O próximo capítulo apresenta as avaliações de usuários sobre EvaML e EvaSIM.

[3]https://github.com/midiacom/eva-robot/tree/master/EvaML-EvaSIM-source-code

7 Avaliação

Este capítulo apresenta o processo de avaliação da linguagem EvaML e do simulador do robô EVA, o EvaSIM. A Seção 7.1 discute a metodologia utilizada para elaborar e estruturar a etapa de avaliação. Na Seção 7.2, são detalhados os experimentos realizados. Na Seção 7.3, os resultados obtidos são analisados e discutidos.

Para realização da avaliação da linguagem EvaML e do simulador EvaSIM, foram efetuados experimentos com usuários, nos quais foram usados como base o paradigma *Goal Question Metric (GQM)* (BAILLIE, 2005), o *System Usability Scale (SUS)* (BROOKE, 1996) e o framework *Cognitive Dimensions of Notations* (BLACKWELL; GREEN, 2003). Nas próximas seções, serão apresentados mais detalhes sobre a metodologia seguida, testes e os resultados obtidos com os experimentos efetuados.

7.1 Metodologia

O paradigma GQM é um mecanismo do tipo *top-down* para definição de objetivos mensuráveis (CALDIERA; ROMBACH, 1994). O GQM e sua aplicação nos trabalhos de Montevecchi (2022) e Mattos (2021) foram utilizados como métodos para orientar e estruturar os experimentos realizados e os resultados obtidos em suas pesquisas. Como apresentado em Caldiera e Rombach (1994), a aplicação da abordagem GQM resulta em um modelo hierárquico de três níveis: conceitual (*goals*), operacional (*questions*) e quantitativo (*metrics*). Para definir esse modelo, primeiro são estabelecidos os *goals* (objetivos). Cada *goal* definido é decomposto em *questions* (questões) que buscam respondê-los, tonando-os quantificáveis. Por sua vez, as *questions* são refinadas em *metrics* (métricas) objetivas e/ou subjetivas.

A Tabela 9 apresenta os objetivos definidos para avaliação da linguagem EvaML e do simulador EvaSIM do ponto de vista do usuário. Para cada objetivo na coluna "Objetivo", a coluna "Descrição" contém os objetos de análise, isto é, a linguagem EvaML e o simulador EvaSIM, um propósito (avaliação), uma questão ou aspecto a ser mensurado

(e.g., para o G1 a usabilidade da linguagem EvaML) e o ponto de vista pelo qual a medição será considerada, (e.g, para o G1 é o ponto de vista dos usuários). As próximas seções discutem as questões e métricas estabelecidas para cada objetivo definido. Para o G1, o framework CDN (*Cognitive Dimensions of Notations*) (BLACKWELL; GREEN, 2003) foi utilizado em conjunto com um grupo de questões adicionais específicas. Para o G2, foram utilizados o questionário *System Usability Scale* (SUS) (BROOKE, 1996) e um conjunto de questões adicionais específicas.

Tabela 9: Objetivos determinados para os experimentos realizados com a linguagem EvaML e o simulador EvaSIM.

Objetivo	Descrição
G1	Analisar a linguagem para desenvolvimento de sessões interativas para o robô EVA, a *EvaML*, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários.
G2	Analisar o software simulador do robô EVA, o *EvaSIM*, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários.

7.1.1 G1 - Questões e Métricas

Para alcançar o objetivo G1, isto é, *Analisar a linguagem para desenvolvimento de sessões interativas para o robô EVA, a EvaML, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários*, foram usadas as questões específicas descritas na Tabela 10 e um questionário, que pode ser visto na Tabela 11, baseado no framework *Cognitive Dimensions of Notations* (CDN).

O framework das "Dimensões Cognitivas" não considera questões de representação em si, ele é baseado apenas em propriedades estruturais. As questões tanto de eficácia (por exemplo, tamanho ideal de botões) quanto de estética estão fora de seu alcance. Essa restrição tem duas grandes vantagens. A primeira é que problemas idênticos (ou melhor, isomórficos) podem ser identificados mesmo quando ocorrem sob formas muito diferentes. Como um exemplo inicial Green (1989) observou que a estrutura de arquivos de um sistema operacional de computador tem muito em comum com a estrutura de um sequenciador de padrões (*Pattern Sequencer*) para uma bateria eletrônica e, portanto, os problemas e soluções em um caso provavelmente se aplicariam ao outro. "Dimensões cognitivas" são características de linguagens de computador consideradas puramente como estruturas de informação ou notações. Portanto, elas se aplicam a muitos tipos de linguagem – interativa ou de programação, de alto ou baixo nível, procedimental ou declarativa, de propósito especial ou de propósito geral (GREEN, 1989). Uma dimensão cognitiva de

uma notação é uma característica da maneira como a informação é estruturada e representada. Cada dimensão descreve um aspecto de uma estrutura de informação. A Tabela 11 contém nove questões relacionadas às seguintes dimensões cognitivas (BLACKWELL; GREEN, 2003):

- **Viscosidade**: Esta dimensão está relacionada à resistência à mudança, isto é, o grau de esforço para se realizar pequenas modificações e manter o estado consistente do artefato produzido;
- **Propensão a erros**: Esta dimensão está relacionada ao fato da notação induzir o usuário a equívocos e o sistema oferecer pouca proteção;
- **Dependências ocultas**: Esta dimensão está relacionada ao fato de importantes ligações entre as entidades não estarem visíveis, isto é, situações onde um elemento A depende de B e B depende de C, geralmente, são explicitadas, porém, a ligação estendida A depende de C, não é explicitada, nem as ligações inversas são explicitadas, como C causa dependência em B e B causa dependência em A;
- **Comprometimento prematuro**: Esta dimensão está relacionada às restrições na ordem de se fazer alguma coisa usando a notação, isto é, o sistema notacional força o usuário a tomar decisões antes das informações apropriadas estarem disponíveis;
- **Verbosidade**: Algumas notações podem ser extensas;
- **Proximidade do mapeamento**: Quão intimamente relacionada está a notação com o resultado que está descrevendo;
- **Consistência**: Esta dimensão está relacionada ao fato de semânticas semelhantes serem expressas de maneiras sintaticamente similares.;
- **Expressividade dos papéis**: A utilidade de qualquer entidade é facilmente percebida;
- **Visibilidade**: Esta dimensão está relacionada à habilidade de facilmente visualizar os componentes, isto é, qual o esforço para se visualizar uma porção desejada do produto sendo construído com a notação;

Na Tabela 10, as questões Q1 e Q8 usam a escala Likert (LIKERT, 1932), onde é possível escolher o nível de concordância ou discordância em relação a uma afirmação. Neste caso, o valor 1 indica a discordância total e 5 indica a concordância total. Para

as questões Q2 a Q4 as respostas são apenas *sim* ou *não* e para as questões Q5 a Q7 as respostas são dadas em minutos. A questão Q8 solicita uma resposta na escala Likert para cada elemento da linguagem EvaML. Para as respostas de cada dimensão referentes ao questionário baseado no framework CDN foi utilizada a escala Likert de 1 a 5. O resultado para cada dimensão será calculado a partir da mediana das respostas dadas. Além das nove questões, foi solicitado a cada participante que desse uma nota final, de um a cinco, para a linguagem EvaML.

Tabela 10: Questões do objetivo G1.

Questão	Descrição
Q1	Você teve dificuldade de entender a estrutura de um script na linguagem EvaML?
Q2	Você concluiu com sucesso a etapa 1 do teste?
Q3	Você concluiu com sucesso a etapa 2 do teste?
Q4	Você concluiu com sucesso a etapa 3 do teste?
Q5	Quanto tempo você levou para concluir a etapa 1 do teste?
Q6	Quanto tempo você levou para concluir a etapa 2 do teste?
Q7	Quanto tempo você levou para concluir a etapa 3 do teste?
Q8	Indique o nível de dificuldade no uso de cada elemento da linguagem EvaML.

A Tabela 12 apresenta as métricas criadas para responder às questões Q1 a Q8 do objetivo G1. A métrica Clareza (CL) é usada para responder à questão Q1, a métrica Eficácia (EF) é usada para responder às questões Q2 a Q4, a métrica Tempo (T) é usada para responder às questões Q5 a Q7, e a métrica Facilidade de Uso Percebida (PEOU) é usada para responder à questão Q8.

7.1.2 G2 - Questões e Métricas

Para alcançar o objetivo G2, isto é, *Analisar o software simulador do robô EVA, o EvaSIM, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários*, foram usadas as questões específicas descritas na Tabela 13 e as questões do SUS (BROOKE, 1996). Na Tabela 13, as questões Q9 a Q12 usam a escala Likert (LIKERT, 1932), onde é possível escolher o nível de concordância ou discordância em relação a uma afirmação. Neste caso, o valor 1 indica a discordância total e 5 indica a concordância total.

A Tabela 14 apresenta as métricas criadas para responder às questões Q9 a Q12 do objetivo G2. A métrica Utilidade Percebida (PU) é usada para responder à questão Q9,

Tabela 11: Questões relacionadas às Dimensões Cognitivas (CDN) do objetivo G1

Dimensão Cognitiva	Fator	Questão (CDN)
Viscosidade	Negativo	A modificação de um elemento do seu código exigiu a modificação de outros elementos?
Propensão a erros	Negativo	Você acha que a sintaxe da linguagem EvaML induz você a cometer erros quando usa seus elementos?
Dependências ocultas	Negativo	Você considera que a linguagem EvaML possui muitas dependências ocultas entre seus elementos?
Comprometimento prematuro	Negativo	Você considera que a linguagem EvaML restringe muito a ordem de criação dos elementos da sua aplicação?
Verbosidade	Negativo	Você considera a linguagem EvaML verbosa?
Proximidade do mapeamento	Positivo	O quão fácil foi implementar as etapas 1, 2 e 3 do teste com os elementos da linguagem EvaML?
Consistência	Positivo	Eu acho que os elementos da linguagem EvaML com semânticas similares são expressos por sintaxes similares.
Expressividade dos papéis	Positivo	Eu entendo a funcionalidade dos elementos da linguagem EvaML.
Visibilidade	Positivo	Usando EvaML, eu consegui identificar claramente os componentes da minha sessão interativa para o robô EVA.

a métrica Facilidade de Uso Percebida (PEOU) é usada para responder às questões Q10 e Q11, e a métrica Clareza (CL) é usada para responder à questão Q12.

Para atingir o objetivo G2, também foi aplicado o questionário SUS (JORDAN et al., 1996). A Escala de Usabilidade do Sistema (SUS) fornece uma ferramenta confiável para medir a usabilidade. Consiste em um questionário de dez itens que utiliza a escala Likert de cinco pontos (LIKERT, 1932). Os usuários precisam selecionar uma das cinco opções para cada questão em que é possível escolher o nível de concordância ou discordância em relação a uma afirmação. Nesse caso, o valor 1 é dado para indicar discordância total e 5 para concordância total. A Tabela 15 mostra o questionário SUS para o EvaSIM. O SUS produz um único número (a pontuação SUS) para representar uma medida composta da usabilidade geral (JORDAN et al., 1996). Além disso, a pontuação para cada questão do SUS não é significativa por si só e, portanto, não será apresentada uma tabela de métricas para cada questão da Tabela 15. A pontuação final SUS em conjunto com as quatro

Tabela 12: Métricas para questões do G1.

Métrica	Descrição	Questões
CL	*Clareza* (SCHREPP; HINDERKS; THOMASCHEWSKI, 2017), isto é, a facilidade de compreensão do produto avaliado. Será medida usando a resposta do usuário em escala Likert de 1 a 5.	Q1
EF	*Eficácia* (BROOKE, 1996), a capacidade dos usuários de concluir as tarefas usando o sistema e a qualidade da saída (do resultado). Será medida, respectivamente, pelo número de respostas positivas às questões Q2, Q3 e Q4 e pela qualidade dos scripts das tarefas 1, 2 e 3.	Q2, Q3 e Q4
T	Tempo, medido em minutos.	Q5, Q6 e Q7
PEOU	*Facilidade de Uso Percebida* (do inglês, *Perceived Ease of Use*) (MARANGUNIĆ; GRANIĆ, 2015), isto é, o grau em que uma pessoa acredita que usar um determinado sistema seria livre de esforço. Será medida usando a resposta do usuário em escala Likert de 1 a 5.	Q8

Tabela 13: Questões do objetivo G2.

Questão	Descrição
Q9	Acho que o EvaSIM me ajudaria a programar scripts para o robô EVA.
Q10	Uma mensagem apresentada pelo EvaSIM foi difícil de entender.
Q11	Foi fácil instalar os componentes de software do EvaSIM.
Q12	Os elementos de interface do EvaSIM representaram bem as funcionalidades do robô.

questões especificas definidas nesta seção são responsáveis por responder ao objetivo G2 com respeito à usabilidade do simulador EvaSIM.

7.1.3 Modelo GQM

A Figura 29 sumariza e ilustra a estrutura do modelo GQM construído nesta dissertação. A seguir, serão apresentados os experimentos realizados para a coleta dos dados.

7.2 Experimentos com Usuários

Nesta dissertação foram realizados experimentos práticos com usuários tanto no experimento com a linguagem EvaML quanto no experimento com o simulador EvaSIM. Esta seção detalha as duas atividades propostas. A Seção 7.2.1 apresenta o experimento com

Tabela 14: Métricas para as questões do G2.

Métrica	Descrição	Questões
PU	*Utilidade Percebida* (do inglês, *Perceived Usefulness*) (MARANGUNIĆ; GRANIĆ, 2015), isto é, o grau em que uma pessoa acredita que o uso de um determinado sistema melhoraria seu desempenho no trabalho. Será medida usando a resposta do usuário em escala Likert de 1 a 5.	Q9
PEOU	*Facilidade de Uso Percebida* (do inglês, *Perceived Ease of Use*) (MARANGUNIĆ; GRANIĆ, 2015), isto é, o grau em que uma pessoa acredita que usar um determinado sistema seria livre de esforço. Será medida usando a resposta do usuário em escala Likert de 1 a 5.	Q10 e Q11
CL	*Clareza* (SCHREPP; HINDERKS; THOMASCHEWSKI, 2017), isto é, a facilidade de compreensão do produto avaliado. Será medida usando a resposta do usuário em escala Likert de 1 a 5.	Q12

a linguagem EvaML e a Seção 7.2.2 apresenta o experimento com o simulador EvaSIM.

7.2.1 Experimento com a Linguagem EvaML

O objetivo G1 busca *analisar a linguagem para desenvolvimento de sessões interativas para o robô EVA, a EvaML, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários.* O conjunto de questões elaboradas a partir desse objetivo configura o questionário usado no experimento com usuários. Para avaliar a usabilidade da linguagem EvaML no desenvolvimento de sessões interativas para o robô EVA, foi realizada uma atividade dividida em três etapas. O objetivo da atividade era a criação de uma sessão interativa para o robô EVA usando os elementos definidos na linguagem EvaML de forma progressiva. Ao finalizar as três etapas, a sessão interativa deveria ser similar a disponível neste link[1]. A descrição detalhada da atividade está disponível no Apêndice C.

O experimento contou com 13 participantes com idades variando de 21 a 35 anos, entre eles, 12 alunos do sexto período do curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense (UFF - Niterói) e uma Mestra em Ciência da Computação. Após o desenvolvimento da atividade, os participantes responderam ao questionário com as questões referentes ao objetivo G1. Esse questionário foi criado

[1]https://www.youtube.com/watch?v=uDkwUEX8IeA

Tabela 15: Questionário SUS para o objetivo G2

Questão	Descrição
Q13	Eu acho que gostaria de usar o EvaSIM com frequência.
Q14	Achei o EvaSIM desnecessariamente complexo.
Q15	Achei o EvaSIM fácil de usar.
Q16	Acho que precisaria do apoio de um técnico para poder usar o EvaSIM.
Q17	Achei que as várias funções do EvaSIM estavam bem integradas.
Q18	Achei que havia muita inconsistência no EvaSIM.
Q19	Eu imagino que a maioria das pessoas aprenderia a usar o EvaSIM muito rapidamente.
Q20	Achei o EvaSIM muito complicado de usar.
Q21	Eu me senti muito confiante usando o EvaSIM.
Q22	Eu precisaria aprender muitas coisas antes de começar a usar o EvaSIM.

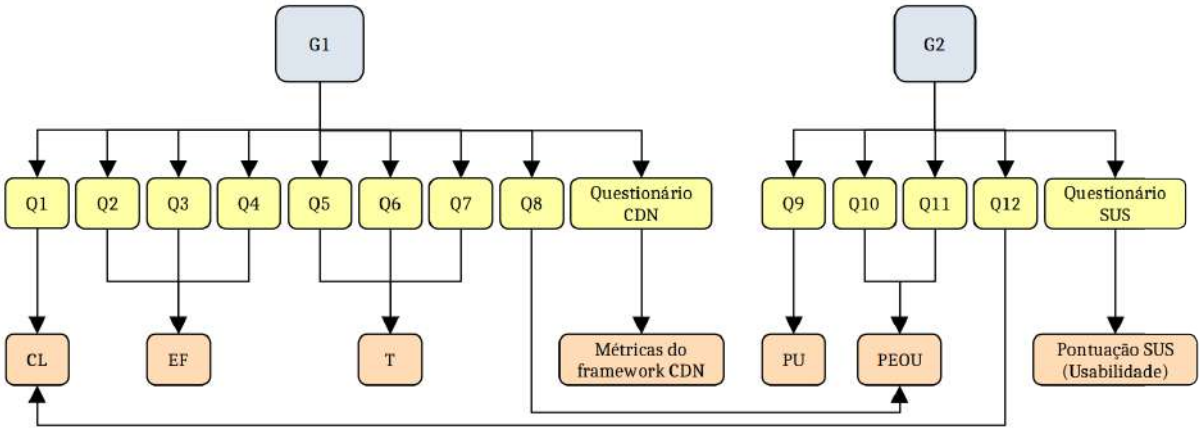

Figura 29: Estrutura hierárquica do modelo GQM desenvolvido nesta dissertação

usando o aplicativo *Google Forms*[2] e disponibilizado *on-line*. O Anexo D contém todo o conteúdo do questionário que foi preenchido pelos participantes. A participação de um dos alunos de graduação foi excluída do experimento pois o mesmo respondeu de maneira incoerente a algumas questões, como por exemplo, na questão que perguntava o nível de experiência do usuário com o robô EVA, o aluno declarou ter muita experiência com o robô. Essa afirmação comprometia diretamente a análise dos dados. Sendo assim, um total de 12 pessoas participaram do experimento com a linguagem EvaML.

Antes que a atividade fosse apresentada aos alunos de graduação, eles foram convidados a assistir uma apresentação sobre o robô EVA, sobre a linguagem EvaML e sobre o simulador EvaSIM. Após a apresentação, a atividade foi liberada aos alunos através de um link[3] com um tutorial. Além das informações referentes à execução da atividade, o tutorial sugeriu aos participantes a instalação de duas ferramentas que poderiam auxiliá-los

[2]Sobre o Google Forms: https://www.google.com/forms/about
[3]https://midiacom.github.io/eva-robot/port-version.html

no desenvolvimento da atividade, a IDE Visual Studio Code[4] e o plugin da Red Hat que dá suporte à linguagem XML. Também foram indicados todos os passos necessários para a instalação das bibliotecas de softwares necessárias para a execução do *parser* EvaML e um link[5] para o download das versões do simulador EvaSIM, tanto para Windows quanto para Linux. Junto com o EvaSIM, foi disponibilizado aos participantes um arquivo de template com uma estrutura mínima de um documento EvaML contendo a declaração e o caminho do arquivo XML Schema da linguagem. Também foi disponibilizado um Manual de Referência da linguagem EvaML[6] e os participantes foram estimulados a fazer uma leitura prévia desse manual.

Como dito anteriormente, a atividade foi dividida em três etapas. Na primeira etapa, os participantes deveriam definir os parâmetros globais da sessão interativa, selecionado o timbre de voz a ser utilizado pelo robô. O objetivo era que o participante pudesse utilizar o elemento que controla os efeitos sensoriais de luz, o elemento que é capaz de executar arquivos de áudio, o comando que controla as expressões do olhar do robô e o comando de fala do EVA. O foco principal da primeira etapa da atividade era a produção de um script EvaML que fizesse com o que o robô se apresentasse ao usuário. Foi pedido aos participantes que cronometrassem o tempo utilizado para finalização da etapa 1.

O objetivo da etapa 2 era fazer com que o robô interagisse com o usuário via voz, obtendo seu nome, perguntando se ele, o usuário, gostaria de jogar um jogo. Nessa tarefa, foram explorados: o elemento de interação via voz do robô, o conceito de variáveis da linguagem e a utilização dos elementos de estrutura condicional da EvaML, como os comandos <switch>, <case> e <default>. Os participantes foram instruídos a marcar o tempo que levaram para executar a etapa 2 da atividade.

Na etapa 3, os participantes deveriam implementar o *Jogo da Imitação*. Nessa tarefa, além dos elementos já apresentados nas duas tarefas anteriores, os participantes deveriam gerar aleatoriedade, utilizando o elemento para geração de números inteiros aleatórios da linguagem, também foi cobrada dos participantes a utilização do elemento responsável por reconhecer a expressão facial do usuário, através da webcam, foram cobrados os conceitos de variáveis específicos da arquitetura do EVA como a utilização do caractere $, a criação e manipulação de variáveis utilizando o comando <counter> e, por último, foi sugerido aos participantes a utilização das *macros*, a fim de facilitar o desenvolvimento e deixar o código mais limpo. A tarefa 3 exigiu um conhecimento mais detalhado dos

[4]https://code.visualstudio.com/
[5]https://github.com/midiacom/eva-robot/tree/master/EvaSIM%20Testing%20Version
[6]https://github.com/midiacom/eva-robot/blob/master/EvaML-Reference-Manual/EvaML-Reference-Manual.pdf

tipos de comparação da EvaML, selecionados através do atributo op do comando <case> e também a utilização de variáveis responsáveis por armazenar o *score* e o número de rodadas jogo.

Durante a realização das três etapas da atividade com a linguagem EvaML, o simulador EvaSIM foi utilizado como ferramenta auxiliar no processo de depuração e testes dos scripts criados pelos participantes.

Após a finalização da atividade com a linguagem EvaML, os participantes enviaram seus scripts e preencheram um questionário contendo, além das questões apresentadas na Tabela 10, outras nove questões que estão relacionadas às seguintes dimensões cognitivas: *viscosidade*; *propensão a erros*; *dependências ocultas*; *comprometimento prematuro*; *verbosidade*; *proximidade do mapeamento*; *consistência*; *expressividade dos papéis*; *visibilidade* (BLACKWELL; GREEN, 2003).

7.2.2 Experimento com o Simulador EvaSIM

O *goal* G2 busca *analisar o software simulador do robô EVA, o EvaSIM, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários*. O conjunto de questões elaboradas a partir desse objetivo configura o questionário usado no experimento. Para avaliar a usabilidade do simulador EvaSIM do ponto de vista do usuário, foi realizado um experimento que contou com a participação dos mesmos 12 usuários que participaram do experimento com a EvaML. Também participou do experimento um outro grupo de 12 usuários que fazem parte do grupo de pesquisa dos criadores do robô EVA, liderado pelo pesquisador Jesus Favela do CICESE no México, composto também por pesquisadores de universidades americanas e espanholas. Além dos usuários já citados, o experimento contou com a participação de mais dois alunos do mestrado em Ciência da Computação da Universidade Federal Fluminense (UFF) em Niterói. Sendo assim, 26 usuários da área da computação com idades variando de 21 a 43 anos participaram do experimento com o simulador EvaSIM. Vale ressaltar a relevância da participação do grupo de pesquisa dos criadores do robô pois são usuários que têm algum tipo de experiência com a versão física do EVA. Após a realização da atividade, os participantes responderam a um questionário com as *questions* referentes ao *goal* G2. Esse questionário foi criado usando o aplicativo *Google Forms*[7] e disponibilizado *on-line*. O Anexo D contém todo o conteúdo do questionário que foi preenchido pelos 26 participantes.

[7]Sobre o Google Forms: https://www.google.com/forms/about

No segundo experimento com o EvaSIM, o objetivo era que o usuário pudesse ter uma experiência com a aplicação desde o processo de instalação. Para isso, o primeiro passo da atividade pedia que o usuário baixasse o EvaSIM na versão específica do seu sistema operacional (Windows ou Linux) e realizasse o processo de instalação. Para a realização desse primeiro passo da atividade proposta, todos os arquivos e instruções necessárias foram disponibilizadas neste link[8].

Após o download e instalação do EvaSIM, o usuário deveria executar o simulador e ativá-lo através do botão "Power On". O usuário então deveria ser capaz de ouvir a saudação, via voz, feita pelo simulador. O próximo passo para a execução da atividade pedia que o participante, através deste link [9], assistisse a um vídeo de uma sessão interativa com o robô EVA. Após a apresentação do vídeo, foi disponibilizado aos participantes, através deste link[10], o arquivo do script para robô EVA que gerou a sessão interativa no vídeo assistido. Após o download do arquivo, o usuário deveria importá-lo no simulador, utilizando o botão "Import Script File". Em seguida, o usuário deveria executar o script clicando no botão "Run" e interagir com o simulador. O script de interação utilizava todos os recursos de interação do robô físico, como: fala, efeitos sensoriais de luz, animação com os leds no tórax do robô, tocar arquivos de áudio, expressão do olhar, captura de voz e reconhecimento de expressões faciais. O script também utilizava os recursos da memória do robô através do uso de variáveis. Durante o processo de interação com o EvaSIM (executando o script), o usuário poderia executar ou encerrar a execução do script, além de observar as várias mensagens exibidas no terminal do simulador, ele também poderia limpar o terminal utilizando o botão "Clear Term.", o usuário também deveria observar as tabelas de memória que apresentavam o estado da memória (simulada) do robô. Como descrito na Seção 6.2, o processo de captura de voz do robô físico, que usa os recursos de fala-para-texto da API do Google, é simulado no EvaSIM através de uma janela para a entrada via texto. O processo de reconhecimento facial do EVA, como descrito na Seção 6.1, é simulado no EvaSIM através de uma janela onde são exibidos cinco emojis representando as emoções. Através da seleção de uma dessas imagens, o usuário pode indicar sua emoção.

Após a finalização da atividade com o simulador EvaSIM, os participantes preencheram um questionário com as questões apresentadas na Tabela 13 além das questões do questionário SUS, apresentadas na Tabela 15.

[8]https://github.com/midiacom/eva-robot/tree/master/EvaSIM%20Testing%20Version
[9]https://www.youtube.com/watch?v=uDkwUEX8IeA
[10]https://drive.google.com/file/d/1rEwCr80NpYrl1VOy5A4czRVHOiOZVcKx/view?usp=sharing

7.3 Resultados

Esta seção analisa e discute os dados obtidos dos questionários e experimentos realizados para responder aos objetivos definidos nesta dissertação.

Para responder às questões do objetivo G1, foram usados o valor médio das respostas à questão Q1, o número de respostas positivas às questões Q2, Q3 e Q4, o tempo médio em minutos para as questões Q5, Q6 e Q7 e a metodologia já estabelecida para as questões do framework *Cognitive Dimensions of Notations* (CDN). Para o objetivo G2, foram usados os valores médios das respostas às questões Q9 a Q12 e a pontuação SUS para as demais questões do questionário SUS.

7.3.1 G1 - Análise dos Resultados

Para alcançar o *goal* G1, isto é, *Analisar a linguagem para desenvolvimento de sessões interativas para o robô EVA, a EvaML, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários*, foram elaboradas questões e métricas especificamente para esta dissertação que, em conjunto com nove questões baseadas nas dimensões cognitivas do framework CDN, indicam se o objetivo G1 foi alcançado.

Após o término do experimento com a linguagem EvaML, cada participante respondeu a um formulário. Uma das questões do formulário pedia para o participante indicar o seu nível de experiência com a linguagem XML. Para a resposta, foi usada uma escala de 1 a 5, onde o 1 indicava que o participante nunca tinha usado XML e o 5 indicava que o participante era experiente no uso de XML. Os 12 participantes do experimento foram separados em dois grupos. O grupo dos experientes em XML, que foi formado pelos 6 participantes que selecionaram as opções 3, 4 ou 5 como resposta, e o grupo dos iniciantes em XML, que foi formado pelos outros 6 participantes que selecionaram a opção 1 ou 2 como resposta. Os resultados desta seção serão discutidos com base na comparação entre os dois grupos, o grupo dos experientes em XML e o grupo dos iniciantes em XML. A Tabela 16 apresenta os valores da mediana (M_d), da moda (M_o), o nível de discordância[11] (ND), o nível de neutralidade[12] (NN) e o nível de concordância[13] (NC) para as respostas à questão Q1, considerando os dois grupos de participantes, os experientes em XML e os

[11]Nível de discordância (ND) é a porcentagem do número de participantes que discordam parcialmente ou discordam fortemente.

[12]Nível de neutralidade (NN) é a porcentagem do número de participantes que nem discordam e nem concordam, ou seja, se colocam como neutros.

[13]Nível de concordância (NC) é a porcentagem do número de participantes que concordam parcialmente ou concordam fortemente.

iniciantes em XML. Já a Tabela 17 apresenta a mediana (M_d), o intervalo interquartil (IIQ), a média ($\overline{X}$) e o desvio padrão (σ) para as respostas referentes às questões Q5 a Q7 também para os dois grupos de participantes. Estas respostas correspondem aos tempos (em minutos) utilizados por cada participante durante a realização de cada etapa da atividade proposta. As respostas dos dois grupos para as questões Q1 e Q5 a Q7 foram analisadas com o teste estatístico não paramétrico de Mann-Whitney e suas respectivas tabelas, apresentam em suas últimas colunas, os valores de U e p (unicaudais) referentes ao teste.

Tabela 16: Questão Q1 (Clareza) para o objetivo G1 - Resultados

Questão	**Métrica**	**Experientes em XML (6)**					**Iniciantes em XML (6)**					**Mann-Whitney**	
		M_d	M_o	**ND**	**NN**	**NC**	M_d	M_o	**ND**	**NN**	**NC**	U	p-exato
Q1	CL	1,0	1	100%	0%	0%	1,0	1	100,00%	0,00%	0,00%	15	0,350

Tabela 17: Questões Q5 a Q7 para o objetivo G1 - Resultados

Questão	**Métrica**	**Experientes em XML (6)**				**Iniciantes em XML (6)**				**Mann-Whitney**	
		M_d	**IIQ**	$\overline{X}$	σ	M_d	**IIQ**	$\overline{X}$	σ	U	p-exato
Q5 (Etapa 1)	T	15,00	4,25	13,83	5,18	20,50	13,75	19,33	7,89	8,50	0,066
Q6 (Etapa 2)	**T**	**13,50**	**6,25**	**12,67**	**4,11**	**22,00**	**10,75**	**21,83**	**8,31**	**5,00**	**0,021**
Q7 (Etapa 3)	T	47,50	23,00	44,50	14,35	62,00	42,50	59,00	21,91	11,50	0,155

Na Seção 7.1.1, foram definidas quatro métricas para responder às questões Q1, Q2 a Q4, Q5 a Q7 e Q8, respectivamente, as métricas Clareza (CL), Eficácia (EF), Tempo (T) e Facilidade de Uso Percebida (PEOU).

Para a análise da métrica Clareza (CL), que foi definida como *"a facilidade de compreensão do produto avaliado"*, foi calculada a mediana das respostas dadas em escala Likert de cinco pontos para a questão Q1. Para essa questão, a resposta com valor 1 indicava que o participante discordava fortemente, isto é, que ele(a) "não encontrou dificuldade" no entendimento da estrutura de um script EvaML. Portanto, um valor de mediana para as respostas à questão Q1 menor que 2 indica que a estrutura de um script EvaML foi facilmente compreendida. Como pode ser visto na Tabela 16, o nível de discordância (ND) para a questão Q1, para os participantes dos dois grupos, foi de 100%. É possível ver também que as modas, para os dois grupos, foram iguais a 1, significando que um maior número de integrantes nos dois grupos discorda fortemente. O valor das medianas, para os dois grupos, foi igual a 1, sendo assim, os integrantes dos dois grupos discordam fortemente mostrando que foi fácil entender a estrutura de um script EvaML. Foi feita uma análise estatística usando o teste não paramétrico de Mann-Whitney com o objetivo de comparar as respostas dos grupos e verificar se há diferença significativa entre elas.

Como pode ser visto na Tabela 16, nas suas duas últimas colunas, U é igual a 15 e p é igual a 0,350. Como $p > 0{,}05$, não há diferença significativa nas respostas dos dois grupos. Com base nos resultados obtidos na análise da métrica Clareza (CL), podemos concluir que a linguagem EvaML foi facilmente compreendida do ponto de vista do usuário.

Em relação às questões Q2, Q3 e Q4, a métrica Eficácia (EF), definida como *"a capacidade dos usuários de concluir as tarefas usando o sistema e a qualidade da saída (do resultado das tarefas)"*, é medida usando-se o somatório do número de respostas positivas apoiadas pela corretude dos scripts dos usuários para as etapas 1, 2 e 3 da atividade com a linguagem EvaML. Através das respostas no formulário, todos os 12 participantes afirmaram ter concluído com sucesso as três etapas da atividade proposta, com isso, foi obtido 100% de respostas positivas. Os scripts criados pelos participantes foram avaliados e todos os integrantes dos dois grupos fizeram o uso correto dos elementos da linguagem, implementando o "Jogo da Imitação" de maneira semelhante à versão apresentada no vídeo de exemplo. Com base nesses resultados, obtidos na análise da métrica Eficácia (EF), que é baseada no número de respostas positivas apoiadas pela corretude dos scripts dos usuários, pode-se concluir que a linguagem EvaML é eficaz do ponto de vista do usuário.

As questões Q5, Q6 e Q7 são relacionadas ao tempo gasto pelos participantes durante a realização de cada uma das etapas. A métrica Tempo (T) é *"a medida do tempo, em minutos"*, gasto para a execução de cada etapa da atividade. A Tabela 17 apresenta a mediana (M_d), o intervalo interquartil (IIQ), a média ($\overline{X}$) e o desvio padrão (σ) para cada uma dessas questões. O gráfico que resume a distribuição quantitativa desses dados pode ser visto na Figura 30. Ele apresenta as informações para os dois grupos, os experientes em XML e os iniciantes em XML. Para a questão Q5, etapa 1 da atividade, o valor da mediana para o grupo dos experientes em XML foi de 15,00 minutos e o valor do intervalo interquartílico (IIQ) foi de 4,25 minutos. Para o grupo dos iniciantes em XML, a mediana foi de 20,50 minutos e o valor do IIQ foi 13,75 minutos. Para essa etapa, o valor de U foi 8,5 e o valor de p foi 0,066. Como $p > 0{,}05$, não há diferença significativa entre as opiniões dos dois grupos para Q5. Como pode ser visto no gráfico, para a etapa 1 da atividade, o grupo dos experientes em XML possui um *outlier*[14], que foi um participante que concluiu a etapa em apenas 3 minutos. Para a questão Q6, etapa 2 da atividade proposta, o grupo dos experientes em XML apresentou uma mediana com valor 13,50 minutos e IIQ=6,25 minutos. Para o grupo dos iniciantes em XML o valor da mediana foi de 22,00 minutos e o valor do IIQ foi de 10,75 minutos, com U=5 e p=0,021. Como p<0,05, para a questão

[14]Os outliers são dados que se diferenciam drasticamente de todos os outros.

Q6, há diferença significativa entre as opiniões dos dois grupos. Para a questão Q7 (etapa 3) a mediana para o grupo dos experientes em XML vale 47,50 minutos e o valor do IIQ é de 23,00 minutos. Para o grupo dos iniciantes em XML a mediana vale 62,00 minutos e o IIQ vale 42,50 minutos, com U igual a 11,5 e p igual a 0,155. Como p>0,05, não há diferença significativa entre as opiniões dos dois grupos para a questão 7. Os resultados das medianas da métrica T para as três etapas indicam que o grupo dos experientes com XML conseguiu terminar cada uma as três etapas em menos tempo que os integrantes do grupo dos iniciantes em XML. Porém, os resultados dos testes estatísticoss aplicados sobre as respostas dos dois grupos para cada uma das etapas da atividade, indicaram que há diferença significativa entre as respostas do dois grupos apenas para a questão 6, que é referente a etapa 2 da atividade. Com base nos resultados obtidos na análise estatística da métrica Tempo (T), não houve diferença no tempo gasto entre os dois grupos nas etapas 1 e 3. Contudo, na etapa 2, o grupo dos experientes em XML terminou a tarefa em um tempo menor que os iniciantes em XML, indicando que a experiência em XML pode influenciar no tempo gasto no processo de criação de scripts com a linguagem EvaML.

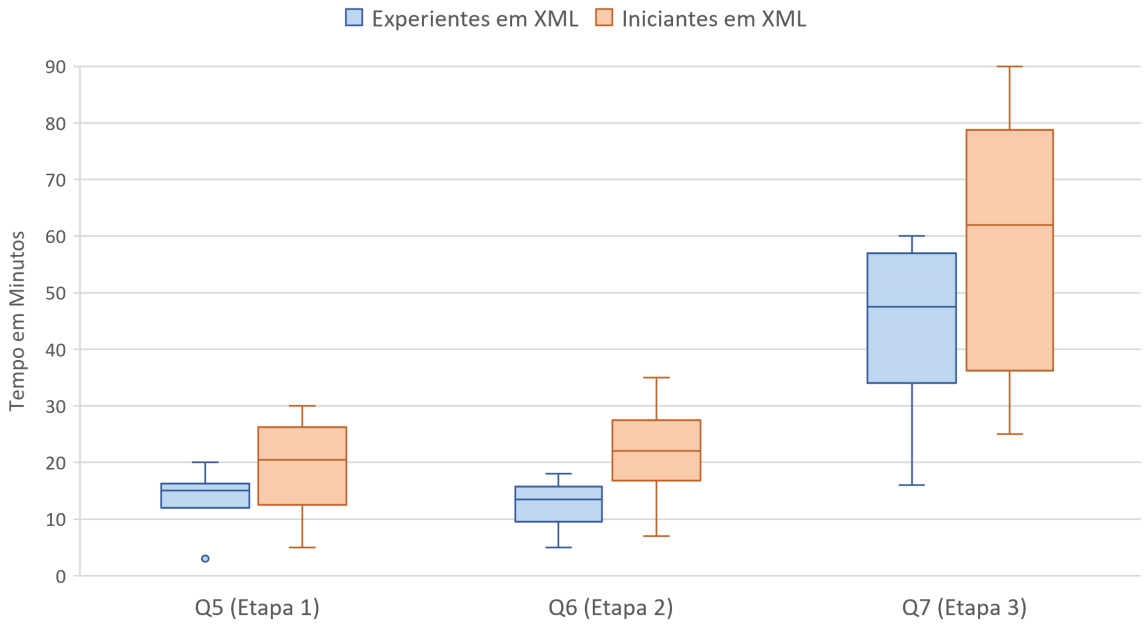

Figura 30: Tempo de duração das etapas 1, 2 e 3 da atividade com a EvaML

A questão Q8 está relacionada ao nível de dificuldade encontrado por cada participante na utilização dos elementos da linguagem EvaML. Essa questão serviu para avaliar as opiniões dos participantes sobre os dezenove elementos (comandos) da EvaML. Para a análise da métrica Facilidade de Uso Percebida (PEOU), que é definida como *"o grau em que uma pessoa acredita que usar um determinado sistema seria livre de esforço"* (MARANGUNIĆ; GRANIĆ, 2015), foram calculadas as medianas e as modas das respostas

dadas em escala Likert de cinco pontos (1 a 5) para Q8. Para essa questão, o valor 1, indicava que foi "muito fácil" utilizar o elemento EvaML, e o valor 5, indicava o oposto, que foi "muito difícil". Para os elementos não usados durante a atividade, os participantes ainda poderiam optar por responder selecionando a opção "não usei o elemento". Os elementos não utilizados não foram computados no cálculos das medianas e das modas. Medianas com valores menores que 3 indicam que o comando foi de fácil utilização. A Tabela 18 apresenta a lista dos 19 elementos da linguagem EvaML avaliados na Q8, as medianas e as modas das respostas referentes à métrica (PEOU) para os dois grupos de participantes e os valores de U e p-exato unicaudais, referentes ao teste estatístico não paramétrico utilizado na comparação das respostas dos dois grupos.

Tabela 18: Questão Q8 para o objetivo G1 - Resultados

Elemento EvaML	**Experientes em XML (6)**		**Iniciantes em XML (6)**		**Mann-Whitney**	
	M_d	M_o	M_d	M_o	U	p-exato
voice	1	1	1	1	14,5	0,465
random	1	1	1	1	15	0,350
wait	1	1	1	1	12	0,197
talk	1	1	1	1	15	0,469
stop	1	1	1	1	18	0,469
light	1	1	1	1	15	0,350
goto	1	1	1,5	1	11,5	0,268
userEmotion	1	1	1	1	14,5	0,294
evaEmotion	1	1	1	1	15	0,350
useMacro	2	1 e 3	1	1	7,5	0,089
listen	1	1	1	1	15	0,350
audio	1	1	1	1	15	0,350
led	**1**	**1**	**1**	**- -**	**- -**	**- -**
motion	**1**	**- -**	**1**	**- -**	**- -**	**- -**
counter	1,5	1	1,5	1	18	0,469
switch	2,5	1 e 4	2	2	13,5	0,242
macro	2	2	1	1	5	0,086
case	2,5	1 e 3	2	2	13,5	0,242
default	1	1	1	1	10	0,214

A primeira observação a ser feita sobre os resultados da métrica (PEOU) relativa a Q8 é em relação aos elementos *led* e *motion*, destacados em negrito na Tabela 18. A utilização desses elementos não foi cobrada em nenhuma das etapas para a conclusão da atividade proposta e, ao analisar os scripts dos participantes, nenhum deles utilizou os elementos *led* e *motion*. Como esses elementos não foram citados no texto da atividade, o que se esperava é que nenhuma opinião fosse expressa a respeito deles, o que não aconteceu

de fato. Nós acreditamos que a opinião dos participantes acerca dos dois elementos foi baseada na leitura prévia do manual de referência da linguagem EvaML, disponibilizado durante a atividade, ou foi resultado de uma má interpretação da questão do formulário. Dos 12 participantes do experimento, 2 participantes do grupo dos experientes em XML e 1 do grupo dos iniciantes opinaram sobre o elemento *led*. Para o elemento *motion* apenas 1 integrante de cada grupo deu sua opinião. Esse número reduzido de participantes nas respostas para os elementos *led* e *motion* impossibilitou a identificação da moda e a execução do teste estatístico sobre os dois grupos, por isso, as colunas referentes à moda, ao valor de U e o valor de p-exato, na Tabela 18, ficaram sem informação. As Figuras 31 e 32 apresentam dois gráficos relacionados às opiniões dos grupos dos especialistas em XML e dos iniciantes em XML, respectivamente. Eles apresentam os comandos da linguagem EvaML com suas respectivas avaliações. Cada barra do gráfico tem sua cor representando uma das classificações possíveis (indicadas na legenda do gráfico) e no interior de cada região colorida o número de participantes que optaram pela classificação indicada. Antes de continuar com a análise da métrica (PEOU) para Q8, é importante ressaltar que os usuários que participaram do experimento foram encorajados a deixar sugestões e impressões sobre a linguagem.

Ao se observar a Tabela 18 para o grupo dos experientes em XML, pode-se ver que 15 dos 19 comandos da linguagens têm suas medianas (M_d) com valores menores que 2 e suas modas (M_o) iguais a 1, indicando que eles foram considerados "muito fáceis" de utilizar pelos experientes com XML. Os 4 comandos restantes têm suas medianas dentro do intervalo $[2, 3)$ indicando que foram considerados "fáceis" de usar. Dentro do grupo dos experientes em XML, 1 integrante não utilizou o comando *useMacro*, 2 integrantes consideraram o comando como de dificuldade "moderada" enquanto 1 participante achou o comando "fácil" de usar e outros 2 participantes o consideraram "muito fácil" de usar. Dois integrantes do grupo dos experientes em XML classificaram o comando *switch* como "difícil" de usar, contudo, um número maior de integrantes do mesmo grupo, 3 ao todo, o classificou como "fácil" ou "muito fácil" de usar. O comando *case* foi classificado como "difícil" de usar por 1 participante do grupo dos experientes em XML, porém, foi avaliado como "fácil" ou "muito fácil" de usar por um número maior de participantes do mesmo grupo, 3 ao todo. Analisando-se o comportamento das medianas na Tabela 18 para o grupo dos iniciantes em XML, é possível ver que 17 dos 19 comandos da linguagem apresentam medianas com valores menores que 2 e modas iguais a 1, indicando que os iniciantes em XML classificaram esses elementos como "muito fáceis" de usar. Apenas dois comandos, o comando *switch* e o comando *case*, foram considerados "fáceis" de usar

pelos usuários inciantes em XML. Ao se observar o gráfico na Figura 32, pode-se ver o destaque em vermelho para a classificação do comando *goto*, que foi avaliado por apenas um participante do grupo dos iniciantes como "muito difícil" de usar. Um participante do grupo dos experientes em XML e dois do grupo dos iniciantes em XML relataram em seus comentários algum tipo de dificuldade no uso do *goto* dentro da estrutura *switch*. Como pode ser visto na Tabela 18 os valores de p são todos maiores que 0,05, indicando que não há diferença significativa entre as respostas dos dois grupos. Com base nos resultados obtidos na análise da métrica Facilidade de Uso Percebida (PEOU), concluímos que a utilização dos elementos da linguagem EvaML seria livre de esforço, do ponto de vista do usuário.

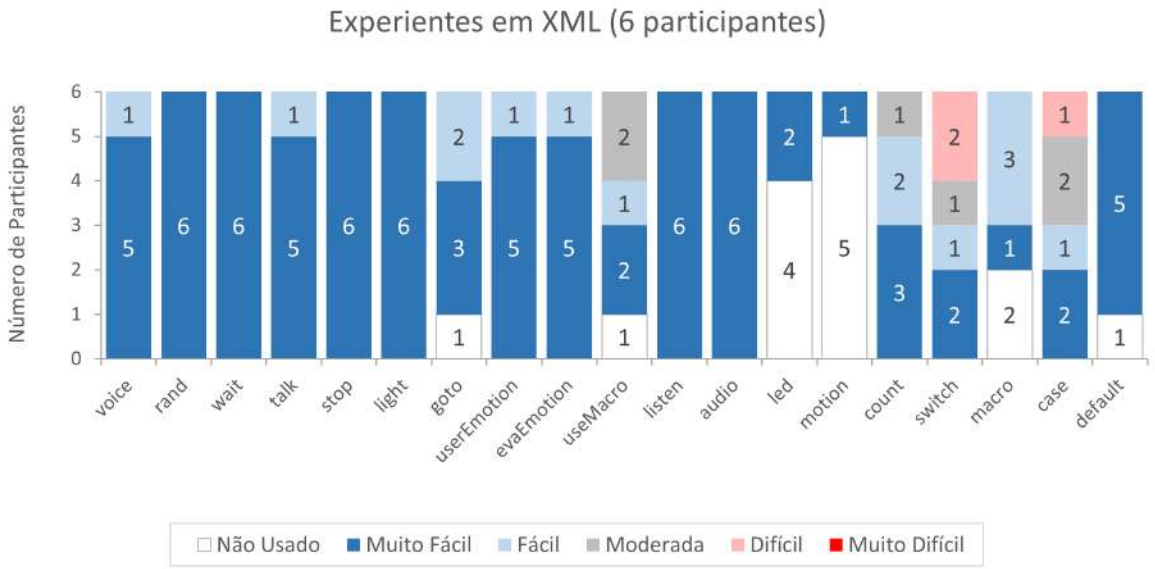

Figura 31: Nível de dificuldade dos comandos da EvaML do ponto de vista dos experientes em XML

Para atingir o objetivo G1, além das questões Q1 a Q8, os participantes responderam a um questionário com nove questões relacionadas a nove dimensões cognitivas baseadas no framework CDN (BLACKWELL; GREEN, 2003). Essas nove questões podem ser vistas na Tabela 11. Para as respostas de cada dimensão foi utilizada a escala Likert de 1 a 5. O resultado de cada dimensão será calculado a partir da mediana das respostas dadas. Os resultados com medianas no intervalo [1, 2) indicam que o participante discorda fortemente com a questão. Já os resultados com medianas no intervalo [2, 3) indicam que o participante discorda parcialmente. Os resultados com medianas no intervalo [3, 4) indicam que o participante não discorda e nem concorda com a questão. Os resultados com medianas no intervalo [4, 5) indicam que o participante concorda parcialmente. E, por último, os resultados com medianas iguais a 5 indicam que o participante concorda

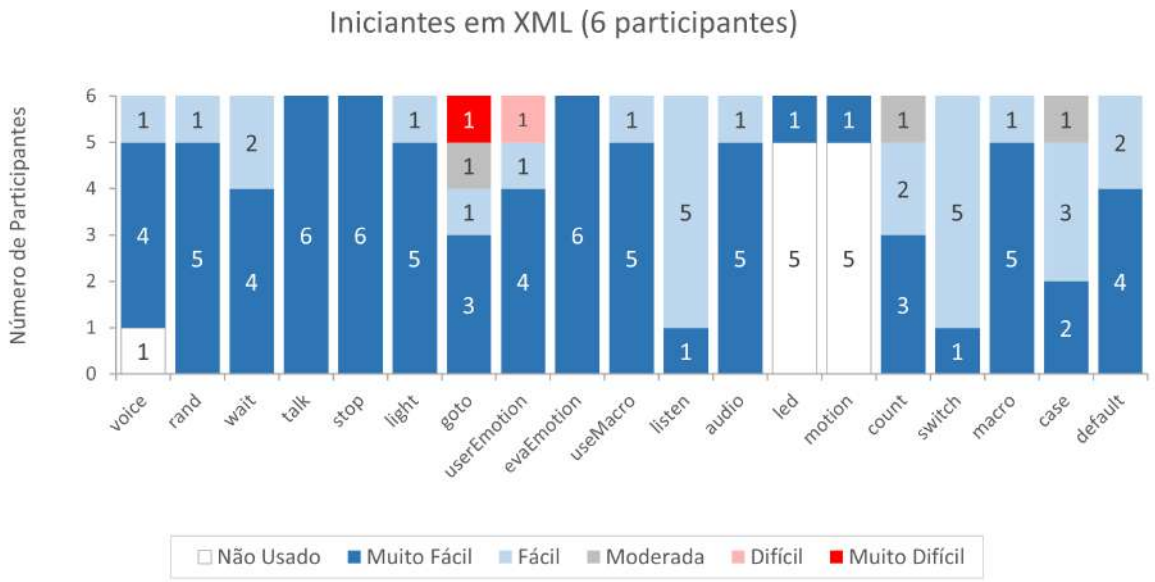

Figura 32: Nível de dificuldade dos comandos da EvaML do ponto de vista dos iniciantes em XML

fortemente com a questão. Além das nove questões, foi solicitado a cada participante que desse uma nota final, de um a cinco, para a linguagem EvaML.

É importante destacar que, como mostrado na Tabela 11, algumas dimensões cognitivas estão relacionadas a fatores positivos da linguagem, como: *proximidade do mapeamento*, *consistência*, *expressividade dos papéis* e *visibilidade*. Enquanto outras dimensões cognitivas estão ligadas a fatores negativos da linguagem, como: *viscosidade*, *propensão a erros*, *dependências ocultas*, *comprometimento prematuro* e *verbosidade*. Uma boa avaliação de usabilidade de uma linguagem é aquela em que as dimensões cognitivas, que estão relacionadas a fatores positivos da linguagem, são mais bem avaliadas, e que as dimensões cognitivas, que estão ligadas a fatores negativos da linguagem, recebam valores mais baixos no experimento. A Tabela 19 apresenta os valores da mediana (M_d), da moda (M_o), o nível de discordância[15] (ND), o nível de neutralidade[16] (NN) e o nível de concordância[17] (NC) para as respostas de cada uma das nove dimensões cognitivas e para os dois grupos, o dos experientes em XML e dos iniciantes em XML. A tabela também mostra os valores referentes ao teste estatístico não paramétrico de Mann-Whitney aplicado sobre as respostas dos dois grupos.

[15]Nível de discordância (ND) é a porcentagem do número de participantes que discordam parcialmente ou discordam fortemente.

[16]Nível de neutralidade (NN) é a porcentagem do número de participantes que nem discordam e nem concordam, ou seja, se colocam como neutros.

[17]Nível de concordância (NC) é a porcentagem do número de participantes que concordam parcialmente ou concordam fortemente.

Tabela 19: Questões relacionadas às Dimensões Cognitivas para o objetivo G1 - Resultados

Dimensão Cognitiva	Experientes em XML (6)					Iniciantes em XML (6)					Mann-Whitney	
	M_d	M_o	ND	NN	NC	M_d	M_o	ND	NN	NC	U	p-exato
Viscosidade	1,5	1	66,67%	16,67%	16,67%	1,5	1 e 2	100,00%	0,00%	0,00%	15,0	0,350
Propensão a Erros	1	1	100,00%	0,00%	0,00%	1,5	1	83,33%	16,67%	0,00%	14,0	0,294
Dependências Ocultas	3	3	33,33%	66,67%	0,00%	2,5	2 e 3	50,00%	33,33%	16,67%	16,0	0,409
Comprometimento Prematuro	3,5	4	33,33%	16,67%	50,00%	2	1, 2 e 3	66,67%	33,33%	0,00%	9,0	0,090
Verbosidade	2	2	83,33%	16,67%	0,00%	2	2	66,67%	16,67%	16,67%	10,5	0,120
Proximidade do Map. (1)	5	5	0,00%	0,00%	100,00%	5	5	0,00%	0,00%	100,00%	15,0	0,350
Proximidade do Map. (2)	5	5	0,00%	0,00%	100,00%	5	5	0,00%	0,00%	100,00%	15,0	0,350
Proximidade do Map. (3)	3	2, 3 e 4	33,33%	33,33%	33,33%	3	3	0,00%	66,67%	33,33%	14,0	0,294
Consistência	4	4	0,00%	33,33%	66,67%	4,5	5	0,00%	16,67%	83,33%	11,5	0,155
Expressividade dos Papéis	4,5	4 e 5	0,00%	0,00%	100,00%	4,5	4 e 5	0,00%	0,00%	100,00%	18,0	0,469
Visibilidade	5	5	0,00%	0,00%	100,00%	5	5	0,00%	0,00%	100,00%	15,0	0,350

As Figuras 33 e 34 apresentam as respostas escolhidas para cada uma das questões do questionário sobre as dimensões cognitivas para os dois grupos, o grupo dos especialistas em XML e o grupo dos iniciantes em XML, respectivamente. Cada região colorida das barras contém a indicação da quantidade de participantes que escolheram a opção indicada. Com base nos dados da Tabela 19 e nos gráficos das Figuras 33 e 34 será apresentada um breve discussão sobre os resultados das respostas às questões para os dois grupos. Primeiro serão analisadas as respostas relacionadas às dimensões cognitivas que estão ligadas a fatores negativos da linguagem, como a *viscosidade*, *propensão a erros*, *dependências ocultas*, *comprometimento prematuro* e *verbosidade*.

Para a dimensão *viscosidade*, os experientes em XML apresentaram um nível de discordância de aproximadamente 66,67% indicando que a maioria do grupo discorda que a modificação de um elemento do seu código exigiu a modificação de outros elementos. O valor da moda ($M_o = 1$) para o grupo mostra que, dos que discordaram, a maioria discorda fortemente. Com base no valor da mediana ($M_d = 1{,}5$) conclui-se que o grupo dos experientes em XML discorda fortemente sobre esse questão. O grupo dos iniciantes em XML apresentou um nível de discordância (ND) de 100%, ou seja, todos os integrantes do grupo discordam sobre essa questão. A moda para o grupo foi 1 e 2, mostrando que 50%

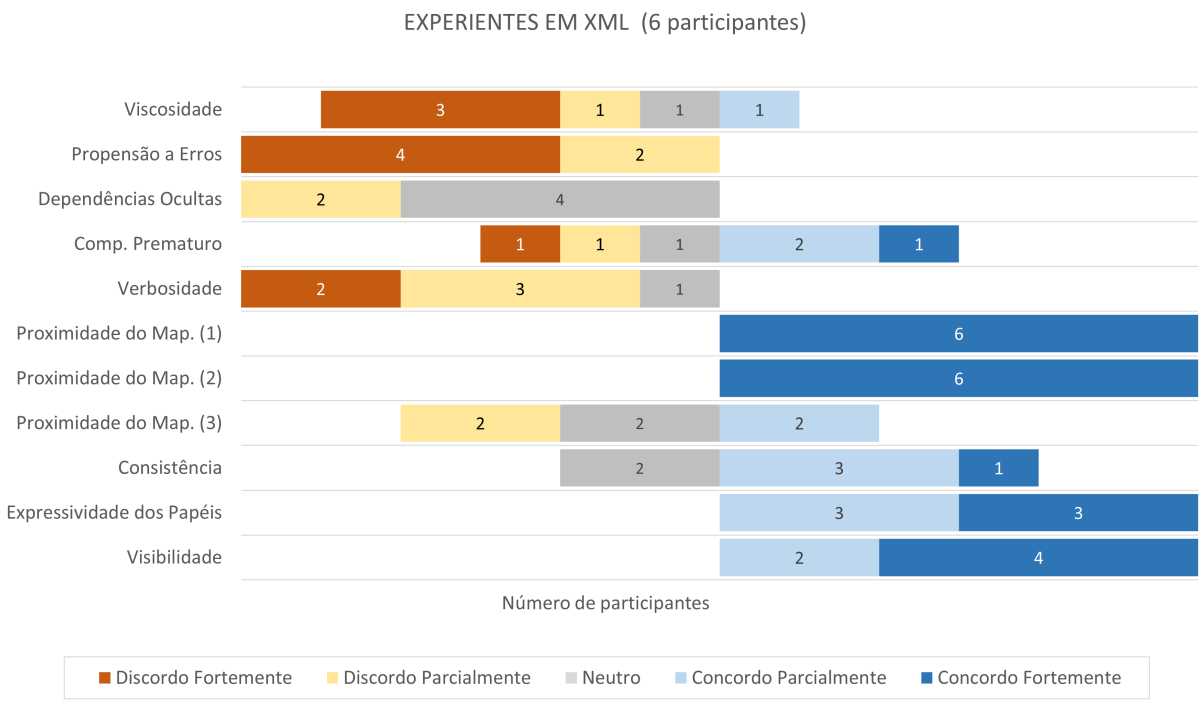

Figura 33: Respostas do grupo dos experientes em XML para as questões relacionadas às dimensões cognitivas (CDN)

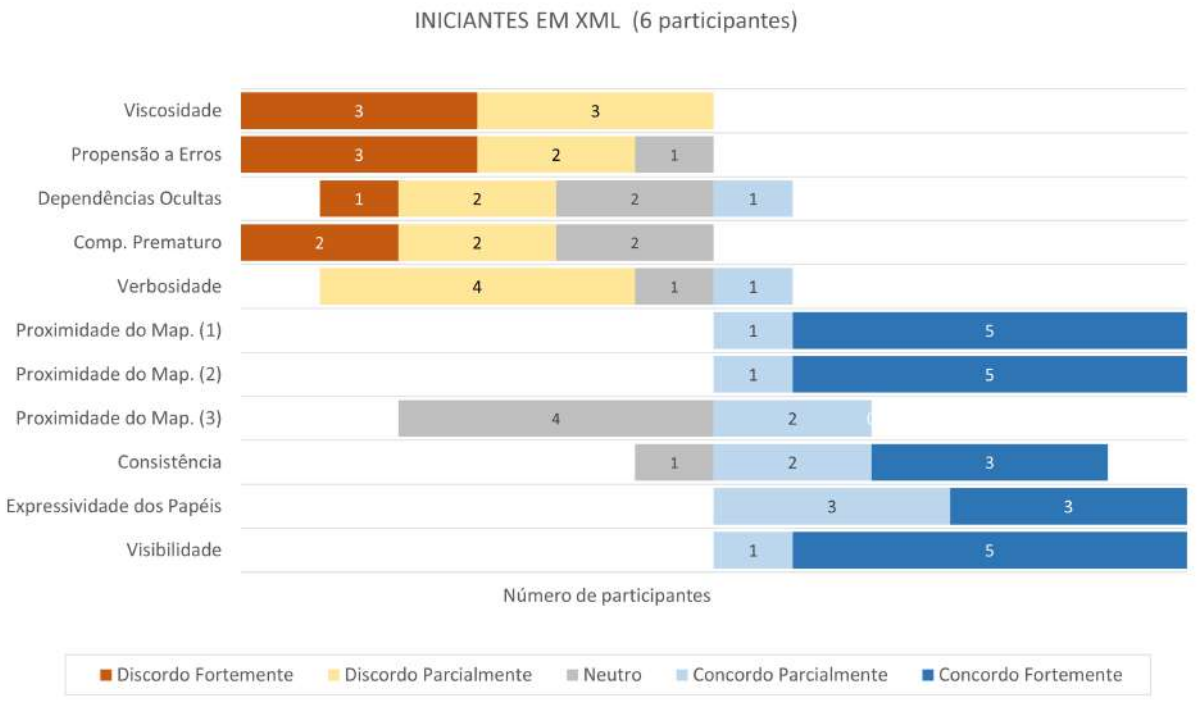

Figura 34: Respostas do grupo dos iniciantes para as questões relacionadas às dimensões cognitivas (CDN)

dos integrantes discorda fortemente e os outros 50% discorda parcialmente. Com base no valor da mediana ($M_d = 1{,}5$) conclui-se que os iniciantes discordam fortemente que a modificação de um elemento do seu código exigiu a modificação de outros elementos.

O nível de discordância do grupo dos experientes em XML, nas suas avaliações referentes à dimensão *propensão a erros*, foi de 100%, indicando que todos os integrantes do grupo discordam que a linguagem EvaML induza o usuário a cometer erros. O valor da moda ($M_o = 1$) para o grupo mostra que a maioria discorda fortemente. Com base no valor da mediana ($M_d = 1$) conclui-se que os experientes em XML discordam fortemente com a questão. Os iniciantes em XML apresentaram um nível de discordância de 83,33% e moda igual a 1. Com base no valor da mediana ($M_d = 1{,}5$) conclui-se que os iniciantes também discordam fortemente que a linguagem EvaML induza o usuário a cometer erros.

Algumas dimensões cognitivas possuem uma definição bastante abstrata e muitas vezes de difícil compreensão. A dimensão *dependências ocultas* é uma delas. Dos experientes em XML, 1/3 discorda parcialmente que a EvaML possua muitas dependências ocultas e 2/3 se posicionaram neutros sobre essa questão. Segundo Raaijmakers et al. (2000), um problema com o ponto médio da escala Likert é que ele pode significar, além da neutralidade, que o usuário "não tem resposta", assim como quando está "indeciso", "não sabe" ou "não tem opinião". Alguns participantes demonstraram não ter compreendido corretamente o conceito sobre *dependências ocultas* e um nível de neutralidade (NN) de aproximadamente 66,67%, para o grupo dos experientes, pode indicar uma dificuldade dos participantes em avaliar corretamente o significado da questão. Com base na mediana ($M_d = 3$) pode-se concluir que os experientes em XML não discordam e nem concordam que a EvaML possua muitas dependências ocultas. Os iniciantes em XML apresentaram um nível de discordância de 50%, contra 33,33% de nível de neutralidade e 16,67% de nível de concordância. O valor da mediana para o grupo dos iniciantes em XML foi de 2,5, podendo-se concluir que os inciantes em XML discordam parcialmente que a EvaML possua muitas dependências ocultas entre seus elementos.

Para a dimensão cognitiva *comprometimento prematuro*, com base no valor da mediana ($M_d = 3{,}5$) para o grupo dos especialistas em XML, pode-se concluir que o grupo se posicionou com neutralidade sobre a questão da linguagem restringir muito a ordem de criação dos elementos das suas aplicações. Com o valor da mediana igual a 2 e com 66,67% de nível de discordância, conclui-se que os iniciantes em XML discordam parcialmente com a questão.

Uma linguagem verbosa é aquela que precisa de mais palavras ou palavras mais lon-

gas do que o necessário para expressar adequadamente a intenção do código. Sobre a dimensão *verbosidade*, com base no valor das suas medianas ($M_d = 2$), pode-se concluir que os dois grupos discordam parcialmente que a EvaML seria um linguagem verbosa. Os experientes em XML apresentaram um nível de discordância maior sobre essa dimensão, 83,33% enquanto os iniciantes apresentaram um nível de discordância de aproximadamente 66,67%. Considerando que a EvaML é uma linguagem baseada em XML e que XML é uma linguagem naturalmente verbosa, esse resultado é bastante promissor.

A partir deste ponto, será apresentada uma breve discussão sobre os resultados das respostas relacionadas às dimensões ligadas a fatores positivos da linguagem, como a *consistência*, *expressividade dos papéis*, *visibilidade* e a *proximidade do mapeamento*.

Para a dimensão *consistência* os experientes em XML apresentaram um nível de concordância de 66,67% e os iniciantes apresentaram um nível de concordância de 83,33%. Com valores de medianas ($4 \leq M_d < 5$) pode-se concluir que os dois grupos concordam que os elementos da linguagem EvaML com semânticas similares são expressos por sintaxes similares.

A dimensão *expressividade dos papéis* está relacionada à capacidade do usuário de entender as funcionalidades dos elementos da linguagem. Para essa questão, os resultados para dois grupos foram exatamente os mesmos. Os dois apresentaram um nível de concordância de 100%, com medianas iguais a 4,5. Os dois grupos apresentaram modas bimodais ($M_o = 4$ e 5) mostrando que 50% dos usuários dos dois grupos concordou parcialmente e os outros 50% concordou fortemente com a questão. Com base no valor das medianas ($M_d = 4{,}5$), pode-se concluir que os dois grupos concordaram parcialmente que foram capazes de entender as funcionalidades dos elementos da linguagem EvaML.

A *visibilidade* está relacionada à habilidade de facilmente visualizar os componentes da linguagem. Tanto os experientes em XML quanto os iniciantes, apresentaram um nível de concordância de 100% com valores de medianas iguais a 5. Observando-se os gráficos das Figuras 33 e 34 pode-se ver que nem todos os experientes em XML e nem todos os iniciantes concordaram fortemente. Através dos valores das modas dos dois grupos, que foram iguais a 5, pode-se constatar que a maioria dos experientes em XML e dos iniciantes concordou fortemente com a questão. Com base nos valores idênticos das medianas do dois grupos ($M_d = 5$), conclui-se que tanto os experientes em XML quanto os iniciantes puderam facilmente visualizar os componentes da linguagem EvaML.

A dimensão cognitiva *proximidade do mapeamento* está relacionada à proximidade da notação com os elementos do domínio, ou seja, quão intimamente relacionada está a

notação utilizada com o resultado que está descrevendo e através dela busca-se mensurar o quão fácil foi implementar cada etapa da atividade proposta com os elementos da linguagem. Como o experimento com a linguagem EvaML foi uma atividade dividida em três etapas, com níveis crescentes de dificuldade, essa questão foi repartida em três, uma para cada etapa da atividade proposta. Para as etapas 1 e 2 do teste, os dois grupos apresentaram um nível de concordância de 100% com valores iguais para as suas medianas ($M_d = 5$). A partir desses valores pode-se concluir que os dois grupos concordam fortemente que foi muito fácil completar a primeira e a segunda etapa do teste com os elementos da linguagem. A terceira etapa do teste exigia dos participantes uma lógica mais elaborada e um maior entendimento sobre a criação e manipulação de variáveis, exigindo também um maior domínio sobre as estruturas condicionais e de repetição da EvaML. Com valores para as suas medianas iguais a 3, pode-se concluir que os dois grupos concluíram a etapa 3 do teste com um grau de dificuldade médio. Essa dificuldade aparece nos comentários de alguns participantes que indicaram que o manual da linguagem poderia ser mais detalhado sobre a utilização dos comandos *switch* e *goto* juntos. Outros participantes sinalizaram uma certa dificuldade na forma de se trabalhar com variáveis enquanto outros sentiram falta de uma estrutura de loop, como um *for*, por exemplo.

As dimensões ligadas a fatores positivos da linguagem, obtiveram os maiores valores para as suas medianas. As mais bem avaliadas pelos dois grupos, são: *consistência*, *expressividade dos papéis*, *visibilidade* e *proximidade do mapeamento*. A avaliação da dimensão *proximidade do mapeamento* foi dividida em três partes e teve suas etapas 1 e 2 muito bem avaliadas, sendo apenas a sua etapa 3 avaliada com neutralidade pelos dois grupos. As duas dimensões cognitivas que obtiveram os melhores valores para as suas medianas foram a *visibilidade* e a *expressividade dos papéis*.

Ao finalizarem a atividade, os participantes deveriam dar uma nota de 1 a 5 para a linguagem EvaML. A Tabela 20 apresenta os valores da mediana (M_d), do intervalo interquartil (IIQ), da média ($\overline{X}$) e do desvio padrão (σ) para as notas de cada grupo (experientes e iniciantes em XML) e também para os participantes como um todo. Considerando-se as medianas e as médias, tanto para os grupos quanto para o todo, todas as notas finais dadas pelos participantes para a linguagem EvaML foram maiores ou iguais a 4.

Como se pode ver nas Tabelas 19 e 20, os valores de p referentes aos testes estatísticos aplicados sobre as respostas dos dois grupos, são todos maiores do que 0,05, indicando que não há diferença significativa nas respostas dos dois grupos. A partir desse resultado é possível concluir que tanto os experientes em XML quanto os iniciantes, tiveram uma

experiência semelhante com a linguagem EvaML.

Tabela 20: Linguagem EvaML (notas finais) - Resultados

Experientes em XML (6)				**Iniciantes em XML (6)**				**Todos (12)**				**Mann-Whitney**	
M_d	**IIQ**	$\overline{X}$	σ	M_d	**IIQ**	$\overline{X}$	σ	M_d	**IIQ**	$\overline{X}$	σ	U	p**-exato**
4,5	1	4,50	0,50	4	1	4,33	0,47	4	1	4,42	0,49	15	0,350

Os usuários que participaram do experimento foram encorajados a deixar sugestões e impressões sobre a linguagem. Um participante sugeriu a criação de uma estrutura de loop. Um outro participante sugeriu a mudança do nome do elemento *counter* para *number*. Foi sugerida a criação de uma estrutura *if/then/else* e a adição de mais exemplos no manual de referência, principalmente para o elemento *switch*.

Algumas impressões deixadas pelos participantes do experimento foram bem positivas, como: *"Achei sensacional! Achei a linguagem muito intuitiva e fácil de usar"*, *"Uma linguagem facilmente utilizável"*, *"No geral, a linguagem é bem intuitiva e de fácil utilização"* e *"Linguagem muito simples e intuitiva de se usar"*. Destacando que esses comentários vieram de participantes dos dois grupos.

Com base nos resultados das análises das métricas Clareza (CL), Eficácia (EF), Facilidade de Uso Percebida (PEOU) e Tempo (T), associadas às questões Q1, Q2 a Q4, Q5 a Q7 e Q8, junto com a avaliação de usabilidade da linguagem utilizando o framework *Cognitive Dimensions of Notations* (CDN), foi alcançado o objetivo G1 avaliando a usabilidade geral da linguagem EvaML.

7.3.2 G2 - Análise dos Resultados

Para alcançar o objetivo G2, isto é, *Analisar o software simulador do robô EVA, o EvaSIM, com o propósito de avaliação da sua usabilidade do ponto de vista dos usuários*, foram elaboradas questões e métricas especificamente para esta dissertação que, em conjunto com dez questões do questionário SUS, indicam se o objetivo G2 foi alcançado.

A fim de avaliar a usabilidade do simulador EvaSIM do ponto de vista do usuário, foram desenvolvidos dois experimentos. O primeiro experimento contou com a participação dos mesmos usuários do experimento com a linguagem EvaML (12 participantes). O segundo experimento contou também com a participação de um outro grupo de 12 usuários que fazem parte do grupo de pesquisa dos criadores do robô EVA. Além dos já citados, o segundo experimento contou com a participação de mais 2 alunos do mestrado em Ciência da Computação. Como os usuários responderam ao mesmo questionário

com as questões referentes ao objetivo G2 em ambos os experimentos, os resultados serão discutidos em conjunto. Sendo assim, 26 usuários da área da computação com idades variando de 21 a 43 anos participaram do experimento com o simulador EvaSIM. Vale ressaltar a relevância da participação do grupo de pesquisa dos criadores do robô pois são usuários que têm algum tipo de experiência com a versão física do EVA. No formulário havia uma questão que perguntava aos participantes o seu nível de experiência com a versão física do robô EVA. Para as respostas, foi usada a escala Liket de cinco pontos, onde o valor 1 indicava que o usuário nunca teve contato com o robô e o valor 5 indicava que o usuário era muito experiente com o EVA. Para a avaliação dos resultados desta seção, os 26 participantes foram divididos em dois grupos, o grupo dos experientes com o EVA e o grupo dos iniciantes com o EVA. Os participantes que responderam 1 ou 2 a pergunta sobre o nível de experiência com o robô foram colocados no grupo dos iniciantes, enquanto os que responderam de 3 a 5, foram colocados no grupo dos experientes. Após a separação dos grupos, o grupo dos iniciantes com o EVA ficou com 14 participantes e o grupo dos experientes com o EVA ficou com 12.

Na Seção 7.1.2, foram definidas três métricas para responder às questões Q9, (Q10 e Q11) e Q12, respectivamente, as métricas Utilidade Percebida (PU), Facilidade de Uso Percebida (PEOU) e Clareza (CL).

A Tabela 21 apresenta os valores das medianas (M_d), das modas (M_o), os níveis de discordância[18] (ND), os níveis de neutralidade[19] (NN) e os níveis de concordância[20] (NC) para as métricas das questões Q9, Q10, Q11 e Q12, e para os dois grupos de participantes. Os resultados com medianas no intervalo [1, 2) indicam que o participante discorda fortemente com a questão. Já os resultados com medianas no intervalo [2, 3) indicam que o participante discorda parcialmente. Os resultados com medianas no intervalo [3, 4) indicam que o participante não discorda e nem concorda com a questão. Os resultados com medianas no intervalo [4, 5) indicam que o participante concorda parcialmente. E, por último, os resultados com medianas iguais a 5 indicam que o participante concorda fortemente com a questão. As Figuras 35 e 36 apresentam as respostas escolhidas para cada uma das questões Q9 a Q12 para os dois grupos, o grupo dos especialistas em XML e o grupo dos iniciantes em XML, respectivamente. Cada região colorida das barras contém a indicação da quantidade de participantes que escolheram a opção indicada.

[18]Nível de discordância (ND) é a porcentagem do número de participantes que discordam parcialmente ou discordam fortemente.

[19]Nível de neutralidade (NN) é a porcentagem do número de participantes que nem discordam e nem concordam, ou seja, se colocam como neutros.

[20]Nível de concordância (NC) é a porcentagem do número de participantes que concordam parcialmente ou concordam fortemente.

Para a análise da métrica Utilidade Percebida (PU), que é definida como sendo *"o grau em que uma pessoa acredita que o uso de um determinado sistema melhoraria seu desempenho no trabalho"* (MARANGUNIĆ; GRANIĆ, 2015), foi calculada a mediana das respostas dadas em escala Likert de cinco pontos para a questão Q9. Neste caso, o valor 1 indica a discordância total e 5 indica a concordância total. O valor da mediana para as respostas do grupo dos experientes com o EVA, para a questão Q9, foi 5 e o nível de concordância foi de 91,67%. O grupo dos iniciantes, para a mesma questão, apresentou uma mediana com valor igual a 5 e nível de concordância de 100%. A partir dos valores obtidos para as medianas dos dois grupos ($M_d = 5$) conclui-se que os participantes dos dois grupos concordam fortemente que o EvaSIM os ajudaria a programar scripts para o robô EVA. Foi feita uma análise estatística usando o teste não paramétrico de Mann-Whitney com o objetivo de comparar as respostas dos grupos e verificar se há diferença significativa entre eles. A Tabela 21, nas suas duas últimas colunas, apresenta os valores de U e p-exato unicaudais. O valor de U para a Q9 é 74 e o valor de p é 0,315. Como $p > 0{,}05$, não há diferença significativa entre as respostas dos dois grupos. Com base nos resultados obtidos na análise da métrica Utilidade Percebida (PU), pode-se concluir que o simulador EvaSIM é útil do ponto de vista do usuário.

Tabela 21: Questões Q9 a Q12 para o objetivo G2 - Resultados

Questão	**Métrica**	**Experientes com o EVA (12)**					**Iniciantes com o EVA (14)**					**Mann-Whitney**	
		M_d	M_o	**ND**	**NN**	**NC**	M_d	M_o	**ND**	**NN**	**NC**	U	p-exato
Q9	PU	5	5	0,00%	8,33%	91,67%	5	5	0,00%	0,00%	100,00%	74	0,315
Q10	PEOU	1	1	91,67%	0,00%	8,33%	1	1	100,00%	0,00%	0,00%	83,50	0,490
Q11	PEOU	5	5	8,33%	0,00%	91,67%	5	5	0,00%	0,00%	100,00%	70,50	0,247
Q12	CL	5	5	0,00%	8,33%	91,67%	5	5	7,14%	0,00%	92,86%	81	0,450

Para a análise da métrica Facilidade de Uso Percebida (PEOU), definida como *"o grau em que uma pessoa acredita que usar um determinado sistema seria livre de esforço"* (MARANGUNIĆ; GRANIĆ, 2015), foi calculada a mediana das respostas dadas em escala Likert de cinco pontos para as questões Q10 e Q11. Neste caso, o valor 1 indica a discordância total e 5 indica a concordância total. Para a questão Q10, o valor da mediana calculada com base nas respostas do grupo dos experientes com o EVA foi 1, e o nível de discordância foi de 91,67%. Para o grupo dos iniciantes com o EVA, o valor da mediana obtida a partir das respostas à Q10 foi 1 e o nível de discordância foi de 100%. Com base nos valores obtidos das medianas para ambos os grupos ($M_d = 1$) pode-se concluir que os participantes experientes e inciantes com o EVA discordam fortemente que alguma mensagem emitida pelo EvaSIM tenha sido de difícil de compreensão. Os valores de U e p para a questão Q10 são, respectivamente, 83,5 e 0,490. O valor de $p > 0{,}05$ indica que

não há diferença significativa entre as respostas dos dois grupos. Com base nos resultados obtidos na análise da métrica Facilidade de Uso Percebida (PEOU), pode-se concluir que a utilização do simulador EvaSIM é livre de esforço do ponto de vista do usuário.

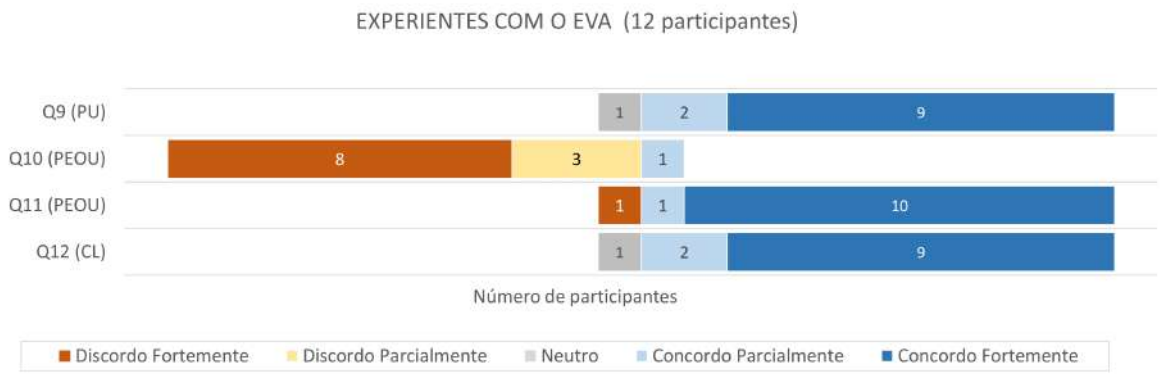

Figura 35: Respostas dos experientes com o EVA para as questões Q9 a Q12

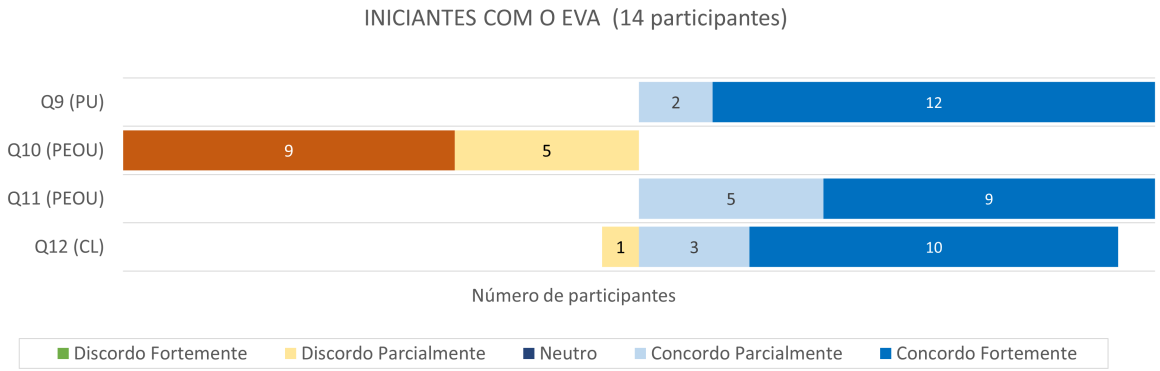

Figura 36: Respostas dos iniciantes com o EVA para as questões Q9 a Q12

Para a questão Q11, também foi utilizada a métrica (PEOU) e sua análise será feita com base no calculo da mediana das respostas obtidas em escala Likert de cinco pontos, com o valor 1 indicando a discordância total e 5 indicando a concordância total. Os valores das medianas para o grupo dos experientes com o robô EVA e para o grupo dos iniciantes foram iguais a 5. Os dois grupos também apresentaram um nível de discordância acima de 91,00%. A partir dos valores obtidos para as medianas dos dois grupos, conclui-se que tanto os experientes com o EVA quanto os iniciantes, concordam fortemente que foi fácil instalar os componentes de software do EvaSIM. Para a questão Q11, U vale 70,5 e p vale 0,247. Como $p > 0{,}05$, não há diferença significativa entre as respostas dos dois grupos. Com base nos resultados obtidos na análise da métrica Facilidade de Uso Percebida (PEOU), concluímos que a utilização do simulador EvaSIM seria livre de esforço do ponto de vista do usuário. Com base nos resultados obtidos na análise da métrica Facilidade de Uso Percebida (PEOU), conclui-se que a utilização do simulador

EvaSIM seria livre de esforço do ponto de vista do usuário.

Para a análise da métrica Clareza (CL), que foi definida como *"a facilidade de compreensão do produto avaliado"* (SCHREPP; HINDERKS; THOMASCHEWSKI, 2017), foi calculada a a mediana das respostas dadas em escala Likert de cinco pontos para a questão Q12. Neste caso, o valor 1 indica a discordância total e 5 indica a concordância total. Para os dois grupos, o dos experientes e dos iniciantes com o EVA, o valor das medianas foi o mesmo, = 5 e ambos apresentaram um nível de concordância acima de 91,00%. Sendo assim, conclui-se que os integrantes dos dois grupos concordam fortemente que os elementos de interface do simulador EvaSIM representaram bem as funcionalidades do robô. Para essa questão, Q12, U vale 81 e p vale 0,450. Como $p > 0{,}05$, não há diferença significativa entre as respostas dos dois grupos. Com base nos resultados obtidos na análise da métrica Clareza (CL), podemos concluir que o simulador EvaSIM foi facilmente compreendido do ponto de vista do usuário.

Além das questões Q9 a Q12, a fim de alcançar o objetivo G2, o participantes tiveram que responder às dez questões do questionário SUS, como discutido na Seção 7.1.2. As questões do SUS podem ser vistas na Tabela 15.

A Tabela 22 apresenta os valores da mediana (M_d), da moda (M_o), o nível de discordância[21] (ND), o nível de neutralidade[22] (NN) e o nível de concordância[23] (NC) para as respostas do questionário SUS. A tabela mostra também os valores de U e p resultantes do teste não paramétrico de Mann-Whitney aplicado sobre as respostas do questionário. O objetivo do teste é verificar se há diferença significativa nas respostas dos dois grupos, o dos experientes com o EVA e o dos iniciantes. Os resultados com medianas no intervalo [1, 2) indicam que o participante discorda fortemente com a questão. Já os resultados com medianas no intervalo [2, 3) indicam que o participante discorda parcialmente. Os resultados com medianas no intervalo [3, 4) indicam que o participante não discorda e nem concorda com a questão. Os resultados com medianas no intervalo [4, 5) indicam que o participante concorda parcialmente. E, por último, os resultados com medianas iguais a 5 indicam que o participante concorda fortemente com a questão. As Figuras 37 e 38 apresentam as respostas escolhidas para cada uma das questões do questionário SUS para os dois grupos, o grupo dos experientes com o EVA e o grupo dos iniciantes com o EVA,

[21] Nível de discordância (ND) é a porcentagem do número de participantes que discordam parcialmente ou discordam fortemente.

[22] Nível de neutralidade (NN) é a porcentagem do número de participantes que nem discordam e nem concordam, ou seja, se colocam como neutros.

[23] Nível de concordância (NC) é a porcentagem do número de participantes que concordam parcialmente ou concordam fortemente.

respectivamente. Cada região colorida das barras contém a indicação da quantidade de participantes que escolheram a opção indicada.

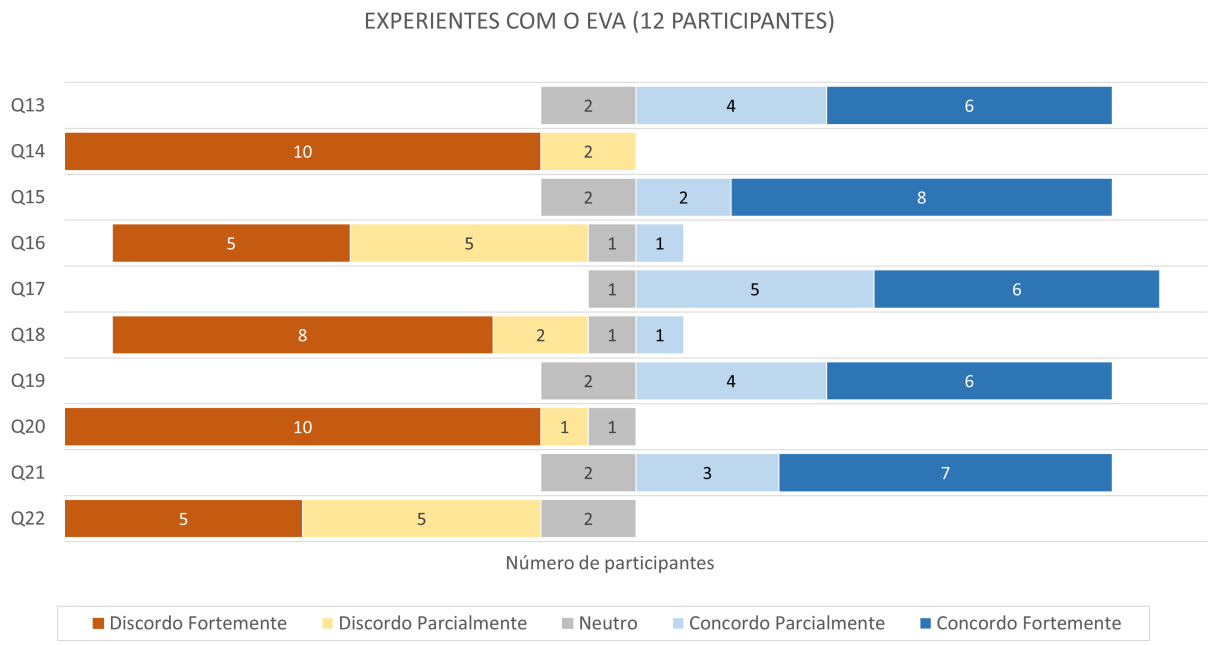

Figura 37: Respostas dos experientes com o EVA para o questionário SUS

Para as questões Q14 até Q21 (questionário SUS), como pode ser verificado na Tabela 22, os valores de p são maiores que 0,05, indicando que não há diferença significativa nas respostas dos dois grupos para essas questões. Já para as questões Q13 e Q22, do mesmo questionário, os valores de p foram menores que 0,05 e isso indica que há diferença significativa nas respostas dos dois grupos para essas questões. Para a questão Q13, que diz *"eu acho que gostaria de usar o EvaSIM com frequência"*, o valor da mediana para o grupo dos experientes com o EVA foi 4,5 e o nível de concordância foi de 83,33%. Já para o grupo dos iniciantes com o EVA, o valor da mediana foi 2,5 e para eles, o nível de discordância foi de 50%. Sendo assim, para a questão Q13, enquanto os experientes com o EVA concordam parcialmente que gostariam de usar o EvaSIM com frequência, o grupo dos iniciantes parece discordar parcialmente com a afirmação.

Para a questão 22, que diz *"eu precisaria aprender muitas coisas antes de começar a usar o EvaSIM"*, o valor da mediana para o grupo dos experientes com o EVA foi 2 e o nível de discordância para essa questão foi de 83,33%. O valor da mediana para o grupo dos iniciantes com o EVA foi 1 e o grupo apresentou um nível de discordância de 100%. Sendo assim, os resultados indicam que os experientes com o EVA discordam parcialmente que teriam que aprender muitas coisas antes de utilizar o EvaSIM, enquanto os iniciantes com o EVA discordam fortemente dessa afirmação.

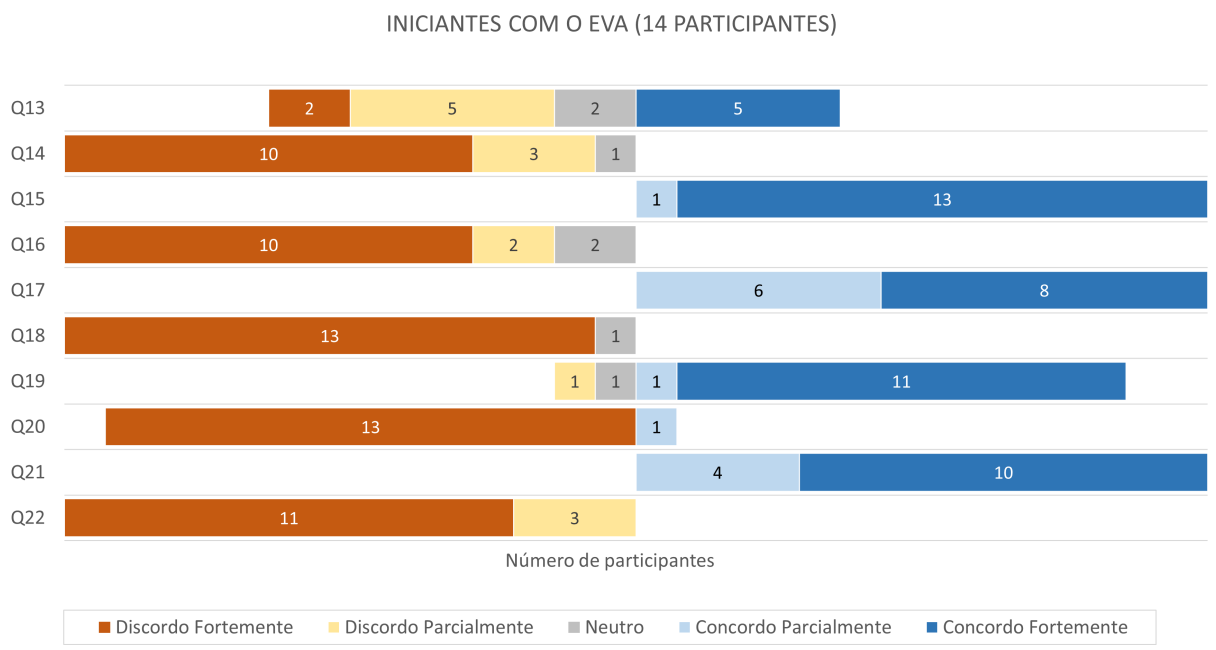

Figura 38: Respostas dos iniciantes com o EVA para o questionário SUS

A metodologia do *System Usability Scale* (SUS) define uma pontuação SUS para cada resposta do seu questionário para representar a medida de usabilidade geral (BROOKE, 1996). Portanto, o valor da pontuação SUS foi calculado para cada usuário e foram calculadas as medianas (M_d), os intervalos interquartis (IIQ), as médias ($\overline{X}$) e os desvios padrão (σ) da pontuação SUS para cada grupo de participantes e também para os participantes como um todo. Esses valores podem ser vistos na Tabela 23. O valor da mediana (M_d) para o grupo dos experientes com o EVA foi 91,25 com um intervalo interquartil (IIQ) igual a 11,87. Para o grupo dos iniciantes com o EVA, os valores foram $M_d = 90$ e $IIQ = 12{,}5$. Os valores de U e p são, respectivamente, 75,5 e 0,334. Como $p > 0{,}05$ não há diferença significativa nas pontuações SUS dos dois grupos. Essas informações são também representadas na forma de um gráfico na Figura 39.

A mediana e o intervalo interquartil obtidos a partir da pontuação SUS de todos os participantes foi $M_d = 90$ e $IIQ = 12{,}05$, respectivamente. A pontuação obtida (90) está acima da média (68) (SAURO; LEWIS, 2016). A usabilidade geral do EvaSIM pode ser classificada como *aceitável* (BANGOR; KORTUM; MILLER, 2008) em uma faixa de 0 a 100, em que abaixo de 50 é *inaceitável*, entre 50 e 70 é *marginalmente aceitável*, e acima de 70 é *aceitável*.

Além da escala de aceitabilidade relativa à pontuação SUS, é possível classificar a usabilidade do EvaSIM utilizando-se uma escala de adjetivos (BANGOR; KORTUM;

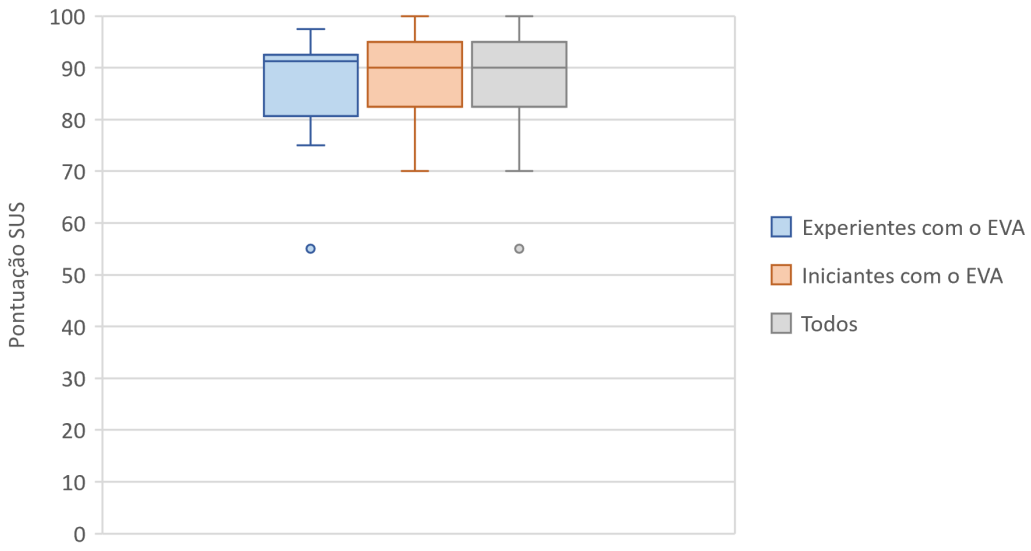

Figura 39: Pontuação SUS - Resultados para os dois grupos e para o todo

Tabela 22: Questões do SUS (Q13 a Q22) para o objetivo G2 - Resultados

Questão	**Experientes com o EVA (12)**					**Iniciantes com o EVA (14)**					**Mann-Whitney**	
SUS	M_d	M_o	**ND**	**N**	**NC**	M_d	M_o	**ND**	**N**	**NC**	U	p**-exato**
Q13	**4,5**	**5**	**0,00%**	**16,67%**	**83,33%**	**2,5**	**2 e 5**	**50,00%**	**14,29%**	**35,71%**	**47**	**0,030**
Q14	1	1	100,00%	0,00%	0,00%	1	1	92,86%	7,14%	0,00%	73	0,298
Q15	5	5	0,00%	16,67%	83,33%	5	5	0,00%	0,00%	100,00%	61	0,126
Q16	2	1 e 2	83,33%	8,33%	8,33%	1	1	85,71%	14,29%	0,00%	61	0,126
Q17	4,5	5	0,00%	8,33%	91,67%	5	5	0,00%	0,00%	100,00%	75	0,334
Q18	1	1	83,33%	8,33%	8,33%	1	1	92,86%	7,14%	0,00%	62,5	0,137
Q19	4,5	5	0,00%	16,67%	83,33%	5	5	7,14%	7,14%	85,71%	64	0,161
Q20	1	1	91,67%	8,33%	0,00%	1	1	92,86%	0,00%	7,14%	77	0,371
Q21	5	5	0,00%	16,67%	83,33%	5	5	0,00%	0,00%	100,00%	69	0,231
Q22	**2**	**1 e 2**	**83,33%**	**16,67%**	**0,00%**	**1**	**1**	**100,00%**	**0,00%**	**0,00%**	**50**	**0,042**

MILLER, 2008), como pode ser visto na Figura 40. Essa escala é dividida em seis categorias associadas à pontuação SUS. Baseado na escala adjetiva, podemos classificar o EvaSIM com o adjetivo "melhor imaginável".

Os participantes do experimento, dos dois grupos, foram incentivados a deixar um comentário sobre o simulador EvaSIM. Um usuário sugeriu uma interface gráfica mais moderna. Um outra sugestão foi a incorporação de algum tipo de movimentação do robô, como a movimentação da cabeça. Um usuário comentou que as informações exibidas no terminal podem ser confusas para um usuário iniciante. Uma sugestão bem interessante, dada por um outro usuário, foi dar mais destaque aos efeitos sensoriais de luz.

Algumas impressões deixadas pelos participantes dos dois grupos foram muito posi-

Tabela 23: Pontuação SUS - Resultados

Experientes com o EVA (12)				Iniciantes com o EVA (14)				Todos (26)				Mann-Whitney	
M_d	IIQ	$\overline{X}$	σ	M_d	IIQ	$\overline{X}$	σ	M_d	IIQ	$\overline{X}$	σ	U	p-exato
91,25	11,87	86,04	11,48	90,00	12,5	88,75	8,49	90,00	12,5	87,50	10,07	75,5	0,334

tivas, como: *"It was very easy to use and test"*, *"The EvaSIM simulator recreates very well the real experience of using the EVA robot"*, *"I've just love it! From a teacher's point of view that cute robot is very useful!"*, *"Gostei muito da interface! A instalação é simples e a forma de usar também, a janela que mostra a fila de variáveis foi muito útil também"*, *"O simulador EvaSIM é simples de usar, fácil de instalar e não apresentou nenhum problema na execução"*.

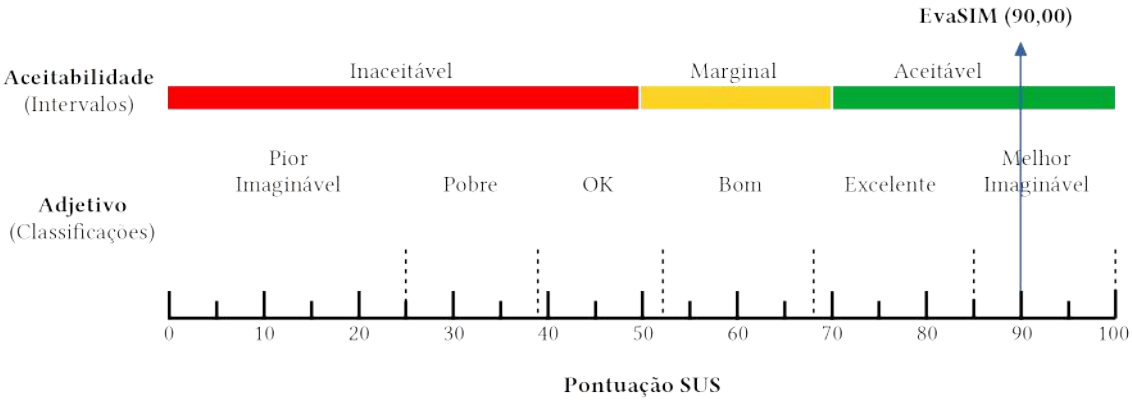

Figura 40: Escalas Adjetiva e de Aceitabilidade associadas à pontuação SUS

Após a análise dos resultados das métricas Utilidade Percebida (PU), Facilidade de Uso Percebida (PEOU), Clareza (CL), associadas às questões Q9 a Q12 e o resultado da classificação do EvaSIM através da pontuação SUS, pode-se concluir que o objetivo G2 foi alcançado.

7.4 Considerações Finais

Para a avaliação da solução proposta nesta dissertação foi planejada uma avaliação utilizando-se o paradigma GQM. Foram estabelecidos dois objetivos (ou *goals*) e para cada um deles, foram definidos conjuntos de questões para respondê-los. Para cada grupo de questões, foram definidas métricas para respondê-las e desta forma, identificar se os objetivos estipulados foram alcançados. Portanto, a linguagem EvaML e o simulador EvaSIM foram analisados em relação à sua usabilidade geral do ponto de vista dos usuários.

Para a obtenção dos dados que seriam utilizados nos cálculos da métricas definidas

para as avaliações da linguagem EvaML e do simulador EvaSIM, foram realizados dois conjuntos de experimentos com usuários.

O primeiro experimento, referente à linguagem EvaML, obteve as respostas de 12 participantes, com idades variando de 21 a 35 anos, entre eles, 11 alunos do sexto período do curso de Bacharelado em Ciência da Computação e uma Mestre em Ciência da Computação. Os participantes foram divididos em dois grupos, o grupo dos experientes em XML (com 6 usuários) e o dos iniciantes em XML (com 6 usuários). Mesmo demandando uma boa quantidade de tempo e dedicação dos participantes, desde a instalação dos componentes de softwares, da leitura do manual de referência da linguagem e da leitura do tutorial sobre a atividade, todos os participantes foram capazes de finalizar, com sucesso, as atividades propostas no experimento.

Através de uma análise estatística, verificou-se que a experiência em XML pode ser um fator relevante no tempo gasto no processo de construção de scripts usando-se a EvaML.

O segundo conjunto de experimentos, referente ao simulador EvaSIM, obteve as respostas de 26 participantes, com idades variando de 21 a 43 anos. Um dos experimentos contou com a participação dos mesmos usuários do experimento com a linguagem EvaML (12 participantes), e o outro experimento contou com a participação de um outro grupo de 12 usuários que fazem parte do grupo de pesquisa dos criadores do robô EVA. Além dos já citados, o segundo experimento contou com a participação de mais 2 alunos do mestrado em Ciência da Computação. Vale ressaltar a relevância da participação do grupo de pesquisa dos criadores do robô pois são usuários que têm algum tipo de experiência com a versão física do EVA. Os participantes foram divididos em dois grupos, o grupo dos experientes com o EVA (com 12 usuários) e o grupo dos iniciantes com o EVA (com 14 usuários).

Todos os objetivos foram alcançados ao final da etapa de avaliação, observando-se uma boa avaliação das dimensões cognitivas que estão ligadas a fatores positivos da linguagem EvaML, considerando-se a utilização do framework CDN. A EvaML foi avaliada positivamente em questões relacionadas à clareza, eficácia, utilidade percebida e percepção da facilidade de uso, do ponto de vista dos usuários. O simulador EvaSIM teve sua usabilidade geral avaliada através da pontuação SUS, obtendo 90 de pontuação e sendo classificado como *aceitável*, na escala de aceitabilidade. Com base na escala adjetiva, o EvaSIM foi classificado como *melhor imaginável*. O simulador EvaSIM foi avaliado positivamente em questões relacionadas à eficácia, utilidade percebida e percepção da facilidade de uso, do ponto de vista dos usuários.

8 Conclusão

Esta dissertação teve como objetivos principais: *"Propor uma linguagem baseada em XML para o desenvolvimento de sessões interativas para o robô EVA, que facilite o desenvolvimento dessas sessões"* e também *"Propor um simulador para o robô EVA capaz de executar suas sessões interativas e auxiliar no desenvolvimento de programas para o robô"*.

Esta dissertação teve como objetivos secundários: *"Estender as capacidades de interação multimodal do robô EVA, com reconhecimento de expressões faciais dos usuários e integração de efeitos sensoriais de luz"* e também *"Propor um jogo sério para terapias de crianças com TEA"*.

Para alcançar o primeiro objetivo desta dissertação, propor uma linguagem baseada em XML para o desenvolvimento de sessões interativas para o robô EVA, que facilite o desenvolvimento dessas sessões, a linguagem EvaML foi proposta. Para isso, foi implementado um parser para linguagem, capaz de transformar o código escrito em EvaML em um arquivo JSON compatível com os arquivos JSON que representam os scripts criados com a linguagem visual do robô. Também foi definido um arquivo XML Schema para a especificação da linguagem EvaML. A fim de avaliar a proposta da linguagem, foi realizada uma avaliação estruturada segundo o paradigma GQM. A linguagem EvaML foi analisada em relação à sua clareza, eficácia, facilidade de uso percebida e teve sua usabilidade avaliada segundo nove dimensões cognitivas do framework CDN, do ponto de vista dos usuários. Todos os objetivos foram alcançados ao final da etapa de avaliação, observando-se bons resultados em relação à sua clareza, eficácia e facilidade percebida de uso. Também foram observados bons resultados na avaliação da usabilidade da linguagem EvaML segundo os critérios definidos pelo framework CDN. Além disso, os usuários avaliaram positivamente a linguagem com afirmações relacionadas à facilidade de uso.

O segundo objetivo, propor um simulador para o robô EVA capaz de executar suas sessões interativas e auxiliar no desenvolvimento de programas para o robô, foi alcançado através da implementação do simulador EvaSIM. A fim de avaliar a proposta do simula-

dor, foi realizada uma avaliação estruturada segundo o paradigma GQM. O EvaSIM foi analisado em relação à sua utilidade percebida, facilidade de uso percebida, clareza e teve sua usabilidade avaliada através das questões do SUS, do ponto de vista dos usuários. Todos os objetivos foram alcançados ao final da etapa de avaliação, observando-se bons resultados em relação à sua utilidade percebida, facilidade de uso percebida e clareza. Utilizando-se a escala de aceitabilidade relativa à pontuação SUS, o EvaSIM teve sua usabilidade geral, classificada como *aceitável* e, utilizando-se uma escala de adjetivos, o EvaSIM foi classificado como "melhor imaginável".

Para alcançar o terceiro objetivo, de estender as capacidades de interação multimodal do robô EVA, dois novos componentes foram adicionados, como elementos de primeira classe, à linguagem de programação visual (VPL) do robô. O primeiro elemento foi o componente que deu ao robô a capacidade de controlar efeitos sensoriais de luz e esses efeitos podem tornar as sessões de terapia mais atraentes e imersivas, principalmente para crianças. O segundo componente, deu ao robô a capacidade de reconhecer as expressões faciais do usuário.

O quarto objetivo, de propor um jogo sério para terapias de crianças com TEA, foi alcançado através do desenvolvimento de um jogo sério em três estágios, utilizando-se a VPL estendida com os dois novos componentes apresentados nesta dissertação. A fim de avaliar a proposta do jogo sério, que utilizou os dois novos componentes apresentados neste trabalho, foram gravados três vídeos, um para cada estágio do jogo, onde uma criança neurotípica de 6 anos jogou o jogo sério com o robô EVA. Os vídeos foram avaliados por 44 profissionais de saúde que responderam a um questionário com questões adaptadas do modelo de aceitação de tecnologia (TAM). Os resultados foram promissores indicando que os profissionais de saúde consideraram a proposta do jogo sério, utilizando os dois componentes propostos nesta dissertação, úteis e fáceis de utilizar em sessões de terapias para crianças com TEA.

Os seguintes trabalhos foram publicados durante o desenvolvimento desta dissertação:

- ROCHA, M., VALENTIM, P., BARRETO, F., MITJANS, A., CRUZ-SANDOVAL, D., FAVELA, J., & MUCHALUAT-SAADE, D. (2021). Towards Enhancing the Multimodal Interaction of a Social Robot to Assist Children with Autism in Emotion Regulation. In International Conference on Pervasive Computing Technologies for Healthcare (pp. 398-415) - PH'2021. Springer, Cham.

- ROCHA, M. M., CRUZ-SANDOVAL, D., FAVELA, J., & MUCHALUAT-SAADE,

D. C. (2022). An Open-Source Socially Assistive Robot for Multisensory Healthcare Therapies. In Proceedings of the 2nd Workshop on Multisensory Experiences - SensoryX'22, together with ACM International Conference on Interactive Media Experiences - ACM IMX 2022. SBC.

- ROCHA, Marcelo M. da; MELO, Sara L. de; MUCHALUAT-SAADE, D. C. (2022) Robôs Socialmente Assistivos: Desenvolvendo Sessões de Terapia Multissensorial com o Robô EVA. Capítulo 4 do Livro de Minicursos do SBCAS 2022. Sociedade Brasileira de Computação, 2022.

- ROCHA, M. M., CRUZ-SANDOVAL, D., FAVELA, J., & MUCHALUAT-SAADE, D. C. (2022) EvaSIM: a Software Simulator for the EVA Open-source Robotics Platform. In: 2022 IEEE RO-MAN: 31st IEEE International Conference on Robot Human Interactive Communication. [S. l.: s. n.], 2022.

8.1 Limitações

A proposta de extensão do robô e de desenvolvimento do Jogo Sério para crianças com Transtorno do Espectro Autista (TEA) foi avaliada durante o cenário pandêmico. Este cenário tornou difícil a aquisição de alguns componentes do robô, incluindo a obtenção do seu corpo impresso em 3D. Sendo assim, a avaliação foi feita utilizando-se uma TV no lugar do robô, e o processo de captura do áudio da voz da criança foi feito usando-se o microfone da própria webcam. Com isso, pode-se considerar o fato de não ter sido usado o robô físico para a gravação dos vídeos, mostrando a criança jogando o jogo sério desenvolvido, como uma limitação da avaliação realizada com os terapeutas em TEA.

Um dos motivos que levaram à escolha da linguagem XML como base para a linguagem EvaML foi o fato de XML ser mais legível para não-programadores do que as linguagens de propósito geral. Apesar desta característica de XML, pessoas com pouco conhecimento técnico em computação teriam dificuldade em desenvolver uma sessão interativa para o EVA usando a linguagem EvaML. Por isso, a avaliação realizada considerou somente usuários com conhecimento em computação, o que pode ser considerada uma limitação deste trabalho.

A representação bidimensional e estática da imagem do robô EVA na interface do simulador EvaSIM impossibilitou que a movimentação da cabeça do robô fosse implementada. Além disso, para o simulador, optou-se pela simplificação do processo de captura de áudio e vídeo. No robô físico, o processo de captura de áudio se dá através da placa

Matrix Voice, enquanto que, no EvaSIM, é utilizada uma caixa de texto para a obtenção da resposta do usuário via teclado. A captura da expressão facial do usuário, feita pelo robô físico, utiliza uma webcam, enquanto que, no simulador, esse processo é substituído por uma janela contendo imagens de *emojis* representando as emoções do usuário. Essas características da implementação do EvaSIM podem ser consideradas limitações do simulador proposto.

8.2 Trabalhos Futuros

Como trabalho futuro, pretende-se mostrar vídeos ou imagens na tela do robô. Isso poderia ser útil para mostrar as emoções de pessoas reais como parte de um jogo. Também é objetivo ampliar a capacidade de comunicação não-verbal do robô, adicionando o elemento "boca" às expressões faciais exibidas no display do EVA. Durante o processo de criação de scripts do jogo, sentiu-se falta um componente na VPL que pudesse gerar números aleatórios. É objetivo também estender a linguagem visual do robô, adicionando esse tipo de componente. Também é nosso objetivo conectar o robô a outros tipos de sensores, como sensores de batimento cardíacos, dando ao EVA a capacidade de obter os sinais vitais do usuário. Outra ideia é integrar o robô em outras aplicações *mulsemídia*. Nesse caso, o robô funcionaria como um avatar e poderia interagir com o jogador, durante um jogo, por meio de voz, dando dicas e motivando o participante, promovendo mais engajamento. Pretende-se também utilizar o robô e suas capacidades estendidas em aplicações na área da educação. Apesar de ser uma plataforma *open-source* a atual arquitetura de software do EVA não permite que novos componentes sejam adicionados ao robô de maneira organizada e sistemática. Por isso, um outro trabalho futuro é propor uma nova arquitetura de controle para o EVA, uma arquitetura que seja facilmente extensível, permitindo que novas funções, controles e dispositivos sejam adicionados ao robô.

A linguagem EvaML é uma posposta de uma linguagem baseada em XML para especificação de sessões interativas para o robô EVA. A partir do processo de *parsing* de um script EvaML, é gerado um arquivo JSON compatível com o EVA. O software interpretador de scripts do robô impôs algumas limitações à linguagem EvaML, principalmente no que diz respeito à criação e manipulação de variáveis. Com a proposta de uma nova arquitetura para o robô, pretende-se também, como trabalho futuro, ampliar os recursos da linguagem EvaML, por exemplo, adicionando à linguagem a capacidade de definir funções parametrizadas. Com a criação de uma nova arquitetura para o robô, é objetivo também que os novos componentes adicionados ao robô sejam integrados à EvaML, como

um biblioteca de software externa à linguagem. Também pretende-se criar uma aplicação de mais alto nível, algo como um *"wizard"*, que possa facilitar o desenvolvimento de scripts por pessoas sem conhecimentos em programação de computadores.

Como trabalho futuro para o simulador EvaSIM, pretende-se criar uma versão em 3D do robô, o que permitirá expandir os comandos interpretados pelo EvaSIM com a inclusão de elementos que controlam a movimento da cabeça do robô físico. Também planeja-se modificar a interface do simulador para que a figura do robô possa ser exibida separadamente, por exemplo, em um tablet. Desta forma, o simulador pode ser usado, em vez do robô real, com potenciais usuários para avaliar o design de um script de interação.

REFERÊNCIAS

AYALA, Angel; CRUZ, Francisco; CAMPOS, Diego; RUBIO, Rodrigo; FERNANDES, Bruno; DAZELEY, Richard. A comparison of humanoid robot simulators: A quantitative approach. In: IEEE. 2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob). [S. l.: s. n.], 2020. p. 1–6.

BAILLIE, J-C. Urbi: Towards a universal robotic low-level programming language. In: IEEE. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. [S. l.: s. n.], 2005. p. 820–825.

BANGOR, Aaron; KORTUM, Philip T; MILLER, James T. An empirical evaluation of the system usability scale. **Intl. Journal of Human–Computer Interaction**, Taylor & Francis, v. 24, n. 6, p. 574–594, 2008.

BARAJAS, Alejandra Ornelas; AL OSMAN, Hussein; SHIRMOHAMMADI, Shervin. A Serious Game for children with Autism Spectrum Disorder as a tool for play therapy. In: IEEE. 2017 IEEE 5th International Conference on Serious Games and Applications for Health (SeGAH). [S. l.: s. n.], 2017. p. 1–7.

BEVILL, Rachael; AZZI, Paul; SPADAFORA, Matthew; PARK, Chung Hyuk; KIM, Hyung Jung; LEE, JongWon; RAIHAN, Kazi; JEON, Myounghoon; HOWARD, Ayanna M. Multisensory robotic therapy to promote natural emotional interaction for children with ASD. In: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI). [S. l.: s. n.], 2016. p. 571–571. DOI: 10.1109/HRI.2016.7451861.

BISCHOFF, Rainer; GRAEFE, Volker. Dependable multimodal communication and interaction with robotic assistants. In: IEEE. PROCEEDINGS. 11th IEEE International Workshop on Robot and Human Interactive Communication. [S. l.: s. n.], 2002. p. 300–305.

BLACKWELL, Alan; GREEN, Thomas. Notational systems–the cognitive dimensions of notations framework. **HCI models, theories, and frameworks: toward an interdisciplinary science. Morgan Kaufmann**, v. 234, 2003.

BROOKE, John. SUS-A quick and dirty usability scale. **Usability evaluation in industry**, London, United kingdom, v. 189, n. 194, p. 4–7, 1996.

CALDIERA, Victor R Basili1 Gianluigi; ROMBACH, H Dieter. The Goal Question Metric Approach. **Encyclopedia of Software Engineering**, p. 528–532, 1994.

CARPIN, Stefano; LEWIS, Mike; WANG, Jijun; BALAKIRSKY, Stephen; SCRAPPER, Chris. USARSim: a robot simulator for research and education. In: IEEE. PROCEEDINGS 2007 IEEE International Conference on Robotics and Automation. [S. l.: s. n.], 2007. p. 1400–1405.

CHEN, Yaofei; DIOS, Rose; MILI, Ali; WU, Lan; WANG, Kefei. An empirical study of programming language trends. **IEEE software**, IEEE, v. 22, n. 3, p. 72–79, 2005.

CIBRIAN, Franceli L; PEÑA, Oscar; ORTEGA, Deysi; TENTORI, Monica. BendableSound: An elastic multisensory surface using touch-based interactions to assist children with severe autism during music therapy. **International Journal of Human-Computer Studies**, Elsevier, v. 107, p. 22–37, 2017.

CRUZ SANDOVAL, D. **Robot conversacional como apoyo a intervenciones no farmacológicas para adultos mayores con demencia Conversational robot to support non-pharmacological interventions for people with dementia**. 2020. Tesis de Doctorado en Ciencias – Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. 125pp. Disponível em: <http://cicese.repositorioinstitucional.mx/jspui/handle/1007/3283>.

CRUZ-SANDOVAL, Dagoberto; FAVELA, Jesus. A Conversational Robot to Conduct Therapeutic Interventions for Dementia. **IEEE Pervasive Computing**, v. 18, n. 2, p. 10–19, 2019. ISSN 15582590. DOI: 10.1109/MPRV.2019.2907020.

_____. A conversational robot to conduct therapeutic interventions for dementia. **IEEE Pervasive Computing**, IEEE, v. 18, n. 2, p. 10–19, 2019.

CRUZ-SANDOVAL, Dagoberto; MORALES-TELLEZ, Arturo; SANDOVAL, Eduardo Benitez; FAVELA, Jesus. A social robot as therapy facilitator in interventions to deal with dementia-related behavioral symptoms. In: IEEE. 2020 15th ACM/IEEE International Conference on Human-Robot Interaction (HRI). [S. l.: s. n.], 2020. p. 161–169.

DAVIS, Fred D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, JSTOR, p. 319–340, 1989.

FACHANTIDIS, Nikolaos; SYRIOPOULOU-DELLI, Christine K.; ZYGOPOULOU, Maria. The Effectiveness of Socially Assistive Robotics in Children With Autism Spectrum Disorder. **International Journal of Developmental Disabilities**, Taylor & Francis, v. 66, n. 2, p. 113–121, 2020. ISSN 20473877. DOI: 10.1080/20473869.2018.1495391. Disponível em: <https://doi.org/10.1080/20473869.2018.1495391>.

FEIL-SEIFER, David; MATARIC', Maja. Robot-assisted therapy for children with autism spectrum disorders. In: PROCEEDINGS of the 7th international conference on Interaction design and children. [S. l.: s. n.], 2008. p. 49–52.

GENA, Cristina; MATTUTINO, Claudio; MALTESE, Walter; PIAZZA, Giulio; RIZZELLO, Enrico. Nao_PRM: an interactive and affective simulator of the Nao robot. In: IEEE. 2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN). [S. l.: s. n.], 2021. p. 727–734.

GOUAILLIER, David; HUGEL, Vincent; BLAZEVIC, Pierre; KILNER, Chris; MONCEAUX, Jérôme; LAFOURCADE, Pascal; MARNIER, Brice; SERRE, Julien; MAISONNIER, Bruno. Mechatronic design of NAO humanoid. In: IEEE. 2009 IEEE international conference on robotics and automation. [S. l.: s. n.], 2009. p. 769–774.

GREEN, Thomas RG. Cognitive dimensions of notations. **People and computers V**, p. 443–460, 1989.

HERWIDODO, Eka Prasetyo; ZAINI, Ahmad et al. INI framework: Indonesian language interpreter software for controlling Nao robot movement. In: IEEE. 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA). [S. l.: s. n.], 2015. p. 63–68.

HIROKAWA, Masakazu; FUNAHASHI, Atsushi; PAN, Yadong; ITOH, Yasushi; SUZUKI, Kenji. Design of a robotic agent that measures smile and facing behavior of children with Autism Spectrum Disorder. In: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). [S. l.: s. n.], 2016. p. 843–848. DOI: 10.1109/ROMAN.2016.7745217.

HU, Paul J; CHAU, Patrick YK; SHENG, Olivia R Liu; TAM, Kar Yan. Examining the technology acceptance model using physician acceptance of telemedicine technology. **Journal of management information systems**, Taylor & Francis, v. 16, n. 2, p. 91–112, 1999.

ISHIMURA, Toshiyuki; KATO, Takeshi; ODA, Kentaro; OHASHI, Takeshi. An open robot simulator environment. In: SPRINGER. ROBOT Soccer World Cup. [S. l.: s. n.], 2003. p. 621–627.

JAVED, Hifza; PARK, Chung Hyuk. Interactions With an Empathetic Agent: Regulating Emotions and Improving Engagement in Autism. **IEEE Robotics Automation Magazine**, v. 26, n. 2, p. 40–48, 2019. DOI: 10.1109/MRA.2019.2904638.

JORDAN, Patrick W; THOMAS, Bruce; MCCLELLAND, Ian Lyall; WEERDMEESTER, Bernard. **Usability evaluation in industry**. [S. l.]: CRC Press, 1996.

JOSUÉ, Marina; MONTEVECCHI, Eyre; ABREU, Raphael; BARRETO, Fabio; SANTOS, Joel; MUCHALUAT-SAADE, Débora Christina. Ambientes multissensoriais aplicados à saúde: desenvolvimento de aplicações e tendências futuras. In: LIVRO de Minicursos do SBCAS 2020. [S. l.]: SBC, 2020. cap. 2.

KIENTZ, Julie A; GOODWIN, Matthew S; HAYES, Gillian R; ABOWD, Gregory D. Interactive technologies for autism. **Synthesis lectures on assistive, rehabilitative, and health-preserving technologies**, Morgan & Claypool Publishers, v. 2, n. 2, p. 1–177, 2013.

KUMAR, Kishy; REEL, Parminder Singh. Analysis of contemporary robotics simulators. In: IEEE. 2011 International Conference on Emerging Trends in Electrical and Computer Technology. [S. l.: s. n.], 2011. p. 661–665.

KURNIAWAN, Oka; LEE, Norman Tiong Seng; DATTA, Subhajit; SOCKALINGAM, Nachamma; LEONG, Pey Kin. Effectiveness of Physical Robot Versus Robot Simulator in Teaching Introductory Programming. In: IEEE. 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). [S. l.: s. n.], 2018. p. 486–493.

LEE, J.; TAKEHASHI, H.; NAGAI, C.; OBINATA, G. Design of a therapeutic robot for interacting with autistic children through interpersonal touch. In: 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication. [S. l.: s. n.], 2012. p. 712–717. DOI: 10.1109/ROMAN.2012.6343835.

LEE, Jaeryoung; TAKEHASHI, Hiroki; NAGAI, Chikara; OBINATA, Goro; STEFANOV, Dimitar. Which robot features can stimulate better responses from children with autism in robot-assisted therapy? **International Journal of Advanced Robotic Systems**, v. 9, 2012. ISSN 17298806. DOI: 10.5772/51128.

LIKERT, Rensis. A technique for the measurement of attitudes. **Archives of psychology**, 1932.

LIMA, Pedro Vitor SG de; BEZERRA, Maria Helena RA; SOUSA TAVARES, Ana Carolinna de; JÚNIOR, José Roberto Fonseca; TEIXEIRA, João Marcelo Xavier Natário; CAJUEIRO, João Paulo Cerquinho; MELO, Guilherme Nunes; HENRIQUES, Diogo B. Improving early robotics education using a line-following robot simulator. In: IEEE. 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE). [S. l.: s. n.], 2018. p. 547–553.

MARANGUNIĆ, Nikola; GRANIĆ, Andrina. Technology acceptance model: a literature review from 1986 to 2013. **Universal access in the information society**, Springer, v. 14, n. 1, p. 81–95, 2015.

MATTOS, Douglas Paulo de. **An Approach for Authoring Mulsemedia Applications Based on Events**. 2021. Tese (Doutorado) – Computing Institute, Fluminense Federal University, Niterói, RJ, Brazil.

MCKNIGHT, Patrick E; NAJAB, Julius. Mann-Whitney U Test. **The Corsini encyclopedia of psychology**, Wiley Online Library, p. 1–1, 2010.

MERNIK, Marjan; HEERING, Jan; SLOANE, Anthony M. When and how to develop domain-specific languages. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 37, n. 4, p. 316–344, 2005.

MITJANS, Adrian Acosta. **Affective computation in human-robot interaction**. 2020. Master thesis – Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. Disponível em: <http://cicese.repositorioinstitucional.mx/jspui/handle/1007/3283>.

MONTEVECCHI, Eyre Brasil Barbosa. **Provendo Interações Oculares em Aplicações Ginga-NCL**. Abr. 2022. Diss. (Mestrado) – Instituto de Computação, UFF, Niterói, RJ, Brasil.

NESTOROV, Nikola; STONE, Emer; LEHANE, Patrick; EIBRAND, Richard. Aspects of socially assistive robots design for dementia care. In: IEEE. 2014 IEEE 27th International Symposium on Computer-Based Medical Systems. [S. l.: s. n.], 2014. p. 396–400.

NILLES, Alexandra Q; BECKMAN, Mattox; GLADISH, Chase; LAVIERS, Amy. Improv: Live coding for robot motion design. In: PROCEEDINGS of the 5th International Conference on Movement and Computing. [S. l.: s. n.], 2018. p. 1–6.

NOVÁK, Marek. Easy implementation of domain specific language using xml. In: PROCEEDINGS of the 10th Scientific Conference of Young Researchers (SCYR 2010), Košice, Slovakia. [S. l.: s. n.], 2010. v. 19.

OEI, Adam C.; PATTERSON, Michael D. Enhancing Cognition with Video Games: A Multiple Game Training Study. **PLoS ONE**, v. 8, n. 3, 2013. ISSN 19326203. DOI: 10.1371/journal.pone.0058546.

PARES, Narcis; MASRI, Paul; VAN WOLFEREN, Gerard; CREED, Chris. Achieving dialogue with children with severe autism in an adaptive multisensory interaction: the"MEDIATE"project. **IEEE Transactions on Visualization and Computer Graphics**, IEEE, v. 11, n. 6, p. 734–743, 2005.

PAULA, Gustavo de; VALENTIM, Pedro; SEIXAS, Flávio; SANTANA, Rosimere; MUCHALUAT-SAADE, Débora. Sensory Effects in Cognitive Exercises for Elderly Users: Stroop Game. In: 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS). [S. l.: s. n.], 2020. p. 132–137. DOI: 10.1109/CBMS49503.2020.00032.

PERICO, Danilo H; HOMEM, Thiago PD; ALMEIDA, Aislan C; SILVA, Isaac J; VILÃO, Claudio O; FERREIRA, Vinicius N; BIANCHI, Reinaldo AC. A robot simulator based on the cross architecture for the development of cognitive robotics. In: IEEE. 2016 XIII Latin American robotics symposium and IV Brazilian robotics symposium (LARS/SBR). [S. l.: s. n.], 2016. p. 317–322.

PINCIROLI, Carlo; BELTRAME, Giovanni. Buzz: a programming language for robot swarms. **IEEE Software**, IEEE, v. 33, n. 4, p. 97–100, 2016.

POT, Emmanuel; MONCEAUX, Jérôme; GELIN, Rodolphe; MAISONNIER, Bruno. Choregraphe: a graphical tool for humanoid robot programming. In: IEEE. RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication. [S. l.: s. n.], 2009. p. 46–51.

RAAIJMAKERS, Quinten AW; VAN HOOF, JTC; HART, H t; VERBOGT, TFMA; VOLLEBERGH, Wilma AM et al. Adolescents' midpoint responses on Likert-type scale items: neutral or missing values? **International Journal of Public Opinion Research**, Oxford University Press, v. 12, p. 208–216, 2000.

ROBINS, Ben; FERRARI, Ester; DAUTENHAHN, Kerstin; KRONREIF, Gernot; PRAZAK-ARAM, Barbara; GELDERBLOM, Gert Jan; TANJA, Bernd; CAPRINO, Francesca; LAUDANNA, Elena; MARTI, Patrizia. Human-centred design methods: Developing scenarios for robot assisted play informed by user panels and field

trials. **International Journal of Human Computer Studies**, Elsevier, v. 68, n. 12, p. 873–898, 2010. ISSN 10715819. DOI: 10.1016/j.ijhcs.2010.08.001. Disponível em: <http://dx.doi.org/10.1016/j.ijhcs.2010.08.001>.

ROCHA, M.; CRUZ-SANDOVAL, D.; FAVELA, J.; MUCHALUAT-SAADE, D. EvaSIM: a Software Simulator for the EVA Open-source Robotics Platform. In: 2022 IEEE RO-MAN: 31st IEEE International Conference on Robot Human Interactive Communication. [S. l.: s. n.], 2022.

ROCHA, Marcelo; MELO, Sara; FAVELA, Jesús; MUCHALUAT-SAADE, Débora Christina. Robôs Socialmente Assistivos: Desenvolvendo Sessões de Terapia Multissensorial com o Robô EVA. In: LIVRO de Minicursos do SBCAS 2022. [S. l.]: SBC, 2022. cap. 4.

ROCHA, Marcelo; VALENTIM, Pedro; BARRETO, Fábio; MITJANS, Adrian; CRUZ-SANDOVAL, Dagoberto; FAVELA, Jesus; MUCHALUAT-SAADE, Débora. Towards Enhancing the Multimodal Interaction of a Social Robot to Assist Children with Autism in Emotion Regulation. In: SPRINGER. INTERNATIONAL Conference on Pervasive Computing Technologies for Healthcare. [S. l.: s. n.], 2022. p. 398–415.

ROCHA, Marcelo Marques da; CRUZ-SANDOVAL, Dagoberto; FAVELA, Jesus; MUCHALUAT-SAADE, Débora C. An Open-Source Socially Assistive Robot for Multisensory Healthcare Therapies. In: SBC. PROCEEDINGS of the 2nd Workshop on Multisensory Experiences-SensoryX'22. [S. l.: s. n.], 2022.

SALLEH, M Haziq Khairul; MISKAM, Mohd Azfar; YUSSOF, Hanafiah; OMAR, Abdul Rahman. HRI assessment of asknao intervention framework via typically developed child. **Procedia Computer Science**, Elsevier, v. 105, p. 333–339, 2017.

SAMONTE, Mary Jane C; GUELOS, Charlzen Mae C; MADARANG, Dana Kei L; MERCADO, Miguel Angelo P. Tap-to-Talk: Filipino Mobile Based Learning Augmentative and Alternative Through Picture Exchange Communication Intervention for Children with Autism. In: PROCEEDINGS of the 2020 The 6th International Conference on Frontiers of Educational Technologies. [S. l.: s. n.], 2020. p. 25–29.

SANDOVAL BRINGAS, J. Andrés; CARREÑO LEÓN, Mónica A.; COTA, Italia Estrada; CARRILLO, Alejandro Leyva. Development of a videogame to improve communication in children with autism. In: 2016 XI Latin American Conference on Learning Objects and Technology (LACLO). [S. l.: s. n.], 2016. p. 1–6. DOI: 10.1109/LACLO.2016.7751751.

SANTATIWONGCHAI, Settapon; KAEWKAMNERDPONG, Boonserm; JUTHAREE, Wisanu; OUNJAI, Kajornvut. BLISS: Using Robot in Learning Intervention to Promote Social Skills for Autism Therapy. In: PROCEEDINGS of the International Convention on Rehabilitation Engineering & Assistive Technology. Midview City, SGP: Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, 2016. (i-CREATe 2016).

SANTOS, Laura; GEMINIANI, Alice; SCHYDLO, Paul; OLIVIERI, Ivana; SANTOS-VICTOR, José; PEDROCCHI, Alessandra. Design of a Robotic Coach for Motor, Social and Cognitive Skills Training Toward Applications With ASD Children. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, IEEE, v. 29, p. 1223–1232, 2021.

SAURO, Jeff; LEWIS, James R. **Quantifying the user experience: Practical statistics for user research**. [S. l.]: Morgan Kaufmann, 2016.

SCHEMA, XML. Xml Schema. **World Wide Web Consortium. http://www. w3. org/XML/Schema. Acesso em**, v. 5, 2004.

SCHREPP, Martin; HINDERKS, Andreas; THOMASCHEWSKI, Jörg. Construction of a Benchmark for the User Experience Questionnaire (UEQ). **International Journal of Interactive Multimedia and Artificial Intelligence**, IMAI Software - International Journal of Interactive Multimedia e Artificial Intelligence, v. 4, n. 4, p. 40, 2017. ISSN 1989-1660.

SHIBATA, Takanori. An overview of human interactive robots for psychological enrichment. **Proceedings of the IEEE**, IEEE, v. 92, n. 11, p. 1749–1758, 2004.

_____. Therapeutic seal robot as biofeedback medical device: Qualitative and quantitative evaluations of robot therapy in dementia care. **Proceedings of the IEEE**, IEEE, v. 100, n. 8, p. 2527–2538, 2012.

SMITH, Tristram. Discrete trial training in the treatment of autism. **Focus on autism and other developmental disabilities**, Sage Publications Sage CA: Thousand Oaks, CA, v. 16, n. 2, p. 86–92, 2001.

TAN, Chek Tien; HARROLD, Natalie; ROSSER, Daniel. Can You CopyMe? An Expression Mimicking Serious Game. In: SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications. Hong Kong, Hong Kong: Association for Computing Machinery, 2013. (SA '13). ISBN 9781450326339. DOI: 10.1145/2543651.2543657. Disponível em: <https://doi.org/10.1145/2543651.2543657>.

TOUSIGNANT, Steve; VAN WYK, Eric; GINI, Maria. An overview of XRobots: A hierarchical state machine-based language, 2011.

TRUONG, Lang Bach; KANG, Dong-Byeong; JI, Sang-Hoon; JEONG, Gu-Min. A control algorithm for robot simulator in educational platform. In: IEEE. ICTC 2011. [S. l.: s. n.], 2011. p. 735–736.

VALENTIM, Pedro Alves; BARRETO, Fábio; MUCHALUAT-SAADE, Débora C. Possibilitando o Reconhecimento de Expressões Faciais em Aplicações Ginga-NCL. In: SBC. ANAIS Estendidos do XXVI Simpósio Brasileiro de Sistemas Multimıdia e Web. [S. l.: s. n.], 2020. p. 53–56.

VASQUEZ, Biel Piero E Alvarado; MATIA, Fernando. A social robot empowered with a new programming language and its performance in a laboratory. In: IEEE. 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA). [S. l.: s. n.], 2019. p. 1–6.

W3C. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. [S. l.: s. n.], 2008. World-Wide Web Consortium Recommendation.

YANG, Shuo; MAO, Xinjun; GE, Binbin; YANG, Sen. The roadmap and challenges of robot programming languages. In: IEEE. 2015 IEEE International Conference on Systems, Man, and Cybernetics. [S. l.: s. n.], 2015. p. 328–333.

ZUBRYCKI, Igor; GRANOSIK, Grzegorz. Designing an interactive device for sensory therapy. In: IEEE. 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI). [S. l.: s. n.], 2016. p. 545–546.

APÊNDICE A - XML SCHEMA DA LINGUAGEM EVAML

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!--
XML Schema for the EvaML Language
This is EvaML
Copyright : 2022 LABORATORIO MIDIACOM, All Rights Reserved.
See https://www.midiacom.uff.br
PublicURI : https://github.com/midiacom/eva-robot/blob/master/EvaML-EvaSIM-
    source-code/Linux%20Version/EvaML-Schema/evaml_schema.xsd
Author : Marcelo Marques da Rocha
Revision: 2022/08/12
-->
  <!-- ** Data type definitions ** -->
  <xs:simpleType name="idType">
    <xs:restriction base="xs:ID"/>
  </xs:simpleType>

  <xs:simpleType name="modeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ON"/>
      <xs:enumeration value="OFF"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="voiceListType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="pt-BR_IsabelaV3Voice"/>
      <xs:enumeration value="en-US_AllisonV3Voice"/>
```

```
27      <xs:enumeration value="en-US_EmilyV3Voice"/>
28      <xs:enumeration value="en-US_HenryV3Voice"/>
29      <xs:enumeration value="es-LA_SofiaV3Voice"/>
30      <xs:enumeration value="es-ES_LauraV3Voice"/>
31      <xs:enumeration value="es-ES_EnriqueV3Voice"/>
32    </xs:restriction>
33  </xs:simpleType>
34
35  <xs:simpleType name="motionListType">
36    <xs:restriction base="xs:string">
37      <xs:enumeration value="YES"/>
38      <xs:enumeration value="NO"/>
39      <xs:enumeration value="CENTER"/>
40      <xs:enumeration value="LEFT"/>
41      <xs:enumeration value="RIGHT"/>
42      <xs:enumeration value="UP"/>
43      <xs:enumeration value="DOWN"/>
44      <xs:enumeration value="ANGRY"/>
45      <xs:enumeration value="2UP"/>
46      <xs:enumeration value="2DOWN"/>
47      <xs:enumeration value="2RIGHT"/>
48      <xs:enumeration value="2LEFT"/>
49    </xs:restriction>
50  </xs:simpleType>
51
52  <xs:simpleType name="voiceStringType">
53    <xs:restriction base="xs:string"/>
54  </xs:simpleType>
55
56  <xs:simpleType name="voiceType">
57    <xs:union memberTypes="voiceListType voiceStringType" />
58  </xs:simpleType>
59
60  <xs:simpleType name="audioListType">
61    <xs:restriction base="xs:string">
62      <xs:enumeration value="applause-moderate"/>
63      <xs:enumeration value="beep-09"/>
64      <xs:enumeration value="cheering"/>
```

```
65      <xs:enumeration value="fanfare2"/>
66      <xs:enumeration value="laugh-01"/>
67      <xs:enumeration value="mario-end-01"/>
68      <xs:enumeration value="mario-end-02"/>
69      <xs:enumeration value="mario-end-03"/>
70      <xs:enumeration value="mario-fundo"/>
71      <xs:enumeration value="mario-game-intro"/>
72      <xs:enumeration value="mario-sound1"/>
73      <xs:enumeration value="mario-sound2"/>
74      <xs:enumeration value="mario-start-01"/>
75      <xs:enumeration value="mario-start-02"/>
76      <xs:enumeration value="oh-no"/>
77      <xs:enumeration value="song-exodus"/>
78      <xs:enumeration value="song-the-girl-from-ipanema"/>
79      <xs:enumeration value="song-the-imperial-march"/>
80      <xs:enumeration value="song-vivaldi-spring"/>
81      <xs:enumeration value="song-exodus"/>
82    </xs:restriction>
83  </xs:simpleType>
84
85  <xs:simpleType name="audioStringType">
86    <xs:restriction base="xs:string"/>
87  </xs:simpleType>
88
89  <xs:simpleType name="audioFileType">
90    <xs:union memberTypes="audioListType audioStringType" />
91  </xs:simpleType>
92
93  <xs:simpleType name="caseOpType">
94    <xs:restriction base="xs:string">
95      <xs:enumeration value="exact"/>
96      <xs:enumeration value="contain"/>
97      <xs:enumeration value="eq"/>
98      <xs:enumeration value="lt"/>
99      <xs:enumeration value="gt"/>
100     <xs:enumeration value="lte"/>
101     <xs:enumeration value="gte"/>
102     <xs:enumeration value="ne"/>
```

```
103     </xs:restriction>
104   </xs:simpleType>
105
106   <xs:simpleType name="audioBlockType">
107     <xs:restriction base="xs:string">
108       <xs:enumeration value="TRUE"/>
109       <xs:enumeration value="FALSE"/>
110     </xs:restriction>
111   </xs:simpleType>
112
113   <xs:simpleType name="ledAnimationType">
114     <xs:restriction base="xs:string">
115       <xs:enumeration value="HAPPY"/>
116       <xs:enumeration value="SAD"/>
117       <xs:enumeration value="ANGRY"/>
118       <xs:enumeration value="STOP"/>
119       <xs:enumeration value="SPEAK"/>
120       <xs:enumeration value="LISTEN"/>
121       <xs:enumeration value="SURPRISE"/>
122     </xs:restriction>
123   </xs:simpleType>
124
125   <xs:simpleType name="lightStateType">
126     <xs:restriction base="xs:string">
127       <xs:enumeration value="ON"/>
128       <xs:enumeration value="OFF"/>
129     </xs:restriction>
130   </xs:simpleType>
131
132   <xs:simpleType name="lightRgbColorType">
133     <xs:restriction base="xs:token">
134       <xs:pattern value="#[\dA-F | a-f ]{6}([\dA-F | a-f][\dA-F | a-f])?"/>
135     </xs:restriction>
136   </xs:simpleType>
137
138   <xs:simpleType name="lightListColorType">
139     <xs:restriction base="xs:string">
140       <xs:enumeration value="WHITE"/>
```

```
141      <xs:enumeration value="BLACK"/>
142      <xs:enumeration value="RED"/>
143      <xs:enumeration value="PINK"/>
144      <xs:enumeration value="GREEN"/>
145      <xs:enumeration value="YELLOW"/>
146      <xs:enumeration value="BLUE"/>
147    </xs:restriction>
148  </xs:simpleType>
149
150  <xs:simpleType name="lightColorType">
151    <xs:union memberTypes="lightListColorType lightRgbColorType" />
152  </xs:simpleType>
153
154  <xs:simpleType name="counterOpType">
155    <xs:restriction base="xs:string">
156      <xs:enumeration value="="/>
157      <xs:enumeration value="+"/>
158      <xs:enumeration value="*"/>
159      <xs:enumeration value="/"/>
160      <xs:enumeration value="%"/>
161    </xs:restriction>
162  </xs:simpleType>
163
164  <xs:simpleType name="evaEmotionType">
165    <xs:restriction base="xs:string">
166      <xs:enumeration value="NEUTRAL"/>
167      <xs:enumeration value="HAPPY"/>
168      <xs:enumeration value="SAD"/>
169      <xs:enumeration value="ANGRY"/>
170    </xs:restriction>
171  </xs:simpleType>
172
173  <!-- ** EvaML Commands Definitions ** -->
174  <xs:element name="random">
175    <xs:complexType>
176      <xs:attribute name="id" type="idType"/>
177      <xs:attribute name="min" type="xs:nonNegativeInteger" use="required"/>
178      <xs:attribute name="max" type="xs:nonNegativeInteger" use="required"/>
```

```
179     </xs:complexType>
180   </xs:element>
181
182   <xs:element name="wait">
183     <xs:complexType>
184     <xs:attribute name="id" type="idType"/>
185       <xs:attribute name="duration" type="xs:nonNegativeInteger" use="required"/>
186     </xs:complexType>
187   </xs:element>
188
189   <xs:element name="talk">
190     <xs:complexType mixed="true">
191       <xs:attribute name="id" type="idType"/>
192     </xs:complexType>
193   </xs:element>
194
195   <xs:element name="stop">
196     <xs:complexType>
197     </xs:complexType>
198   </xs:element>
199
200   <xs:element name="light">
201     <xs:complexType>
202       <xs:attribute name="id" type="idType"/>
203       <xs:attribute name="state" type="lightStateType" use="required"/>
204       <xs:attribute name="color" type="lightColorType" default="WHITE"/>
205     </xs:complexType>
206   </xs:element>
207
208   <xs:element name="goto">
209     <xs:complexType>
210       <xs:attribute name="target" type="xs:IDREF" use="required"/>
211     </xs:complexType>
212   </xs:element>
213
214   <xs:element name="userEmotion">
215     <xs:complexType>
216       <xs:attribute name="id" type="idType"/>
```

```
217     </xs:complexType>
218   </xs:element>
219
220   <xs:element name="evaEmotion">
221     <xs:complexType>
222       <xs:attribute name="id" type="idType"/>
223       <xs:attribute name="emotion" type="evaEmotionType" use="required"/>
224     </xs:complexType>
225   </xs:element>
226
227   <xs:element name="useMacro">
228     <xs:complexType>
229       <xs:attribute name="macro" type="xs:IDREF" use="required"/>
230     </xs:complexType>
231   </xs:element>
232
233   <xs:element name="listen">
234     <xs:complexType>
235       <xs:attribute name="id" type="idType"/>
236     </xs:complexType>
237   </xs:element>
238
239   <xs:element name="audio">
240     <xs:complexType>
241       <xs:attribute name="id" type="idType"/>
242       <xs:attribute name="source" type="audioFileType" use="required"/>
243       <xs:attribute name="block" type="audioBlockType" use="required"/>
244     </xs:complexType>
245   </xs:element>
246
247   <xs:element name="led">
248     <xs:complexType>
249       <xs:attribute name="id" type="idType"/>
250       <xs:attribute name="animation" type="ledAnimationType" use="required"/>
251     </xs:complexType>
252   </xs:element>
253
254   <xs:element name="counter">
```

```
255     <xs:complexType>
256       <xs:attribute name="id" type="idType"/>
257       <xs:attribute name="var" type="xs:string" use="required"/>
258       <xs:attribute name="op" type="counterOpType" use="required"/>
259       <xs:attribute name="value" type="xs:integer" use="required"/>
260     </xs:complexType>
261   </xs:element>
262
263   <xs:element name="switch">
264     <xs:complexType>
265       <xs:sequence>
266         <xs:element ref="case" minOccurs="1" maxOccurs="unbounded" />
267         <xs:element ref="default" minOccurs="0" maxOccurs="1"/>
268       </xs:sequence>
269       <xs:attribute name="id" type="idType"/>
270       <xs:attribute name="var" type="xs:string" use="required"/>
271     </xs:complexType>
272   </xs:element>
273
274   <xs:element name="motion">
275     <xs:complexType>
276       <xs:attribute name="id" type="idType"/>
277       <xs:attribute name="type" type="motionListType" use="required"/>
278     </xs:complexType>
279   </xs:element>
280
281   <xs:element name="voice">
282     <xs:complexType>
283       <xs:attribute name="tone" type="voiceType" use="required"/>
284     </xs:complexType>
285   </xs:element>
286
287   <xs:element name="lightEffects">
288     <xs:complexType>
289       <xs:attribute name="mode" type="modeType" use="required"/>
290     </xs:complexType>
291   </xs:element>
292
```

```
293 <xs:element name="audioEffects">
294   <xs:complexType>
295     <xs:attribute name="mode" type="modeType" use="required"/>
296     <xs:attribute name="vol" type="xs:string" default="100%"/>
297   </xs:complexType>
298 </xs:element>
299
300 <xs:element name="case">
301   <xs:complexType>
302     <xs:choice minOccurs="0" maxOccurs="unbounded">
303       <xs:element ref="random"/>
304       <xs:element ref="wait"/>
305       <xs:element ref="talk"/>
306       <xs:element ref="stop"/>
307       <xs:element ref="light"/>
308       <xs:element ref="goto"/>
309       <xs:element ref="motion"/>
310       <xs:element ref="userEmotion"/>
311       <xs:element ref="evaEmotion"/>
312       <xs:element ref="useMacro"/>
313       <xs:element ref="listen"/>
314       <xs:element ref="audio"/>
315       <xs:element ref="led"/>
316       <xs:element ref="counter"/>
317       <xs:element ref="switch"/>
318     </xs:choice>
319     <xs:attribute name="op" type="caseOpType" use="required"/>
320     <xs:attribute name="value" type="xs:string" use="required"/>
321   </xs:complexType>
322 </xs:element>
323
324 <xs:element name="default">
325   <xs:complexType>
326     <xs:choice minOccurs="0" maxOccurs="unbounded">
327       <xs:element ref="random"/>
328       <xs:element ref="wait"/>
329       <xs:element ref="talk"/>
330       <xs:element ref="stop"/>
```

```
331        <xs:element ref="light"/>
332        <xs:element ref="goto"/>
333        <xs:element ref="motion"/>
334        <xs:element ref="userEmotion"/>
335        <xs:element ref="evaEmotion"/>
336        <xs:element ref="useMacro"/>
337        <xs:element ref="listen"/>
338        <xs:element ref="audio"/>
339        <xs:element ref="led"/>
340        <xs:element ref="counter"/>
341        <xs:element ref="switch"/>
342      </xs:choice>
343    </xs:complexType>
344  </xs:element>
345
346  <xs:element name="macro">
347    <xs:complexType>
348      <xs:choice minOccurs="0" maxOccurs="unbounded">
349        <xs:element ref="random"/>
350        <xs:element ref="wait"/>
351        <xs:element ref="talk"/>
352        <xs:element ref="stop"/>
353        <xs:element ref="light"/>
354        <xs:element ref="goto"/>
355        <xs:element ref="motion"/>
356        <xs:element ref="userEmotion"/>
357        <xs:element ref="evaEmotion"/>
358        <xs:element ref="listen"/>
359        <xs:element ref="audio"/>
360        <xs:element ref="led"/>
361        <xs:element ref="counter"/>
362        <xs:element ref="switch"/>
363      </xs:choice>
364      <xs:attribute name="id" type="idType" use="required" />
365    </xs:complexType>
366  </xs:element>
367
368  <!-- ** Root Section Definition ** -->
```

```
369 <xs:element name="evaml">
370   <xs:complexType>
371     <xs:sequence>
372       <xs:element ref="settings"/>
373       <xs:element ref="script"/>
374       <xs:element ref="macros" minOccurs="0" maxOccurs="1"/>
375     </xs:sequence>
376     <xs:attribute name="name" type="xs:string" use="required"/>
377   </xs:complexType>
378 </xs:element>
379
380 <!-- ** Settings Section Definition ** -->
381 <xs:element name="settings">
382   <xs:complexType>
383     <xs:all>
384       <xs:element ref="voice" minOccurs="1" maxOccurs="1"/>
385       <xs:element ref="lightEffects" minOccurs="0" maxOccurs="1"/>
386       <xs:element ref="audioEffects" minOccurs="0" maxOccurs="1"/>
387     </xs:all>
388   </xs:complexType>
389 </xs:element>
390
391 <!-- ** Script Section Definition ** -->
392 <xs:element name="script">
393   <xs:complexType>
394     <xs:choice minOccurs="0" maxOccurs="unbounded">
395       <xs:element ref="random"/>
396       <xs:element ref="wait"/>
397       <xs:element ref="talk"/>
398       <xs:element ref="stop"/>
399       <xs:element ref="light"/>
400       <xs:element ref="goto"/>
401       <xs:element ref="motion"/>
402       <xs:element ref="userEmotion"/>
403       <xs:element ref="evaEmotion"/>
404       <xs:element ref="useMacro"/>
405       <xs:element ref="listen"/>
406       <xs:element ref="audio"/>
```

```
407         <xs:element ref="led"/>
408         <xs:element ref="counter"/>
409         <xs:element ref="switch"/>
410       </xs:choice>
411     </xs:complexType>
412   </xs:element>
413
414   <!-- ** Macros Section Definition ** -->
415   <xs:element name="macros">
416     <xs:complexType>
417       <xs:sequence minOccurs="1" maxOccurs="unbounded">
418         <xs:element ref="macro"/>
419       </xs:sequence>
420     </xs:complexType>
421   </xs:element>
422 </xs:schema>
```

Listagem A.1: Código XML Schema de especificação da linguagem EvaML

APÊNDICE B - MONTAGEM FÍSICA DO ROBÔ EVA

B.1 Montagem do Robô

Esta seção apresenta uma visão geral do processo de montagem do robô com algumas imagens reais das conexões das placas acopladas às peças do corpo do robô. Também será apresentada a solução para uma dificuldade que pode ocorrer no processo de montagem dos componentes, a configuração dos IDs (identificadores) dos servomotores. Para informações mais detalhadas sobre os arquivos dos modelos 3D e um passo-a-passo da montagem dos componentes, acesse o repositório[1] do EVA no Github.

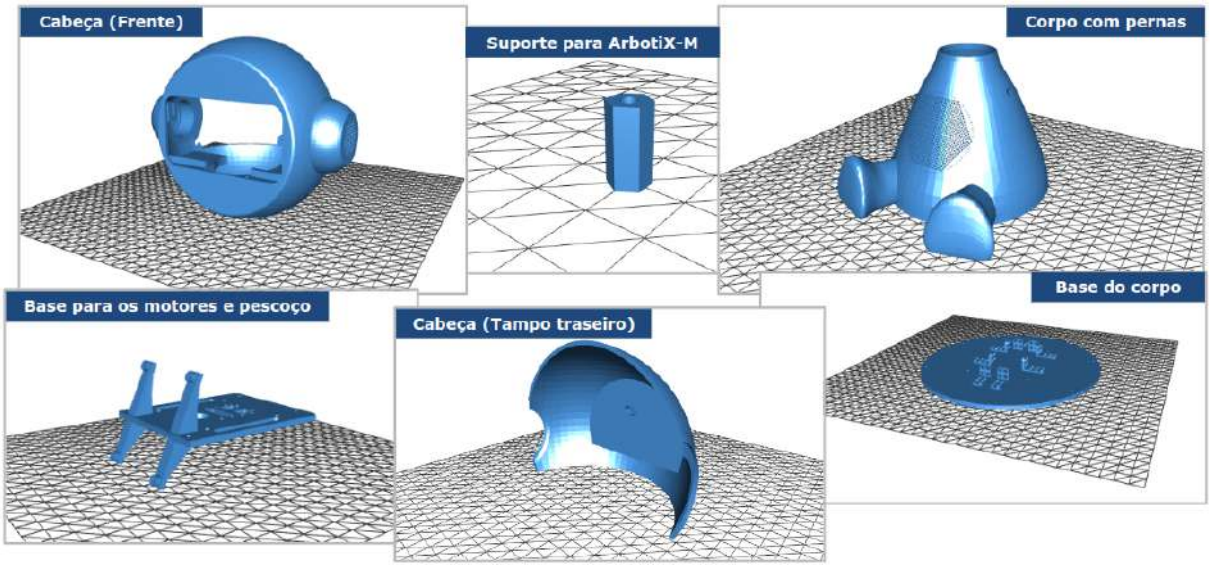

Figura 41: Modelos para a impressão 3D

Todo o corpo do robô foi construído em impressora 3D e os criadores do robô disponibilizam todos os arquivos e modelos para a impressão e construção do EVA. A Figura 41 mostra algumas imagens dos modelos 3D disponíveis no repositório do robô.

[1] https://github.com/eva-social-robot

Com todas as partes do corpo do robô impressas em 3D e todos os componentes de hardware em mãos, pode-se dar início ao processo de junção das partes internas do robô e a conexão das placas, dos cabos, do display e dos dois servomotores. A Figura 42(a) apresenta uma visa frontal da placa Matrix Voice com seus 8 microfones e sua matriz de 18 LEDs RGB. A Figura 42(b) mostra a parte traseira do display dentro da cabeça do robô. Ao display se conectam três cabos, um cabo de energia, um cabo mini HDMI e um cabo USB que é conectado ao Raspberry PI permitindo que o display funcione como dispositivo de entrada através do toque na tela. A Figura 42(c) mostra a Matrix Voice conectada ao Raspberry PI e como esses elementos juntos são fixados em uma peça interna ao robô. Essa peça, que também foi impressa, serve como suporte para todas as placas e também para os dois servomotores.

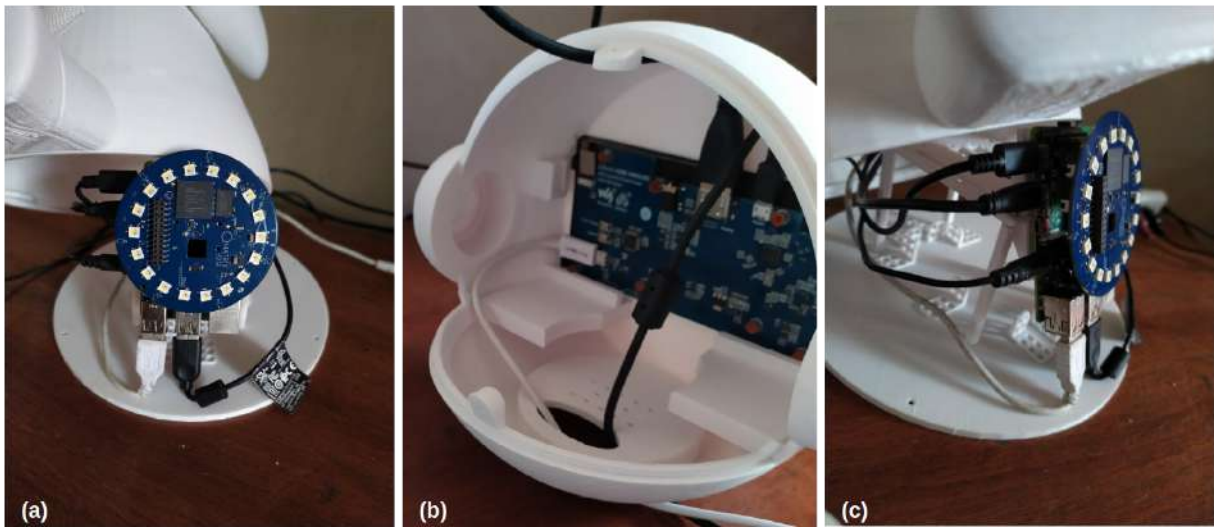

Figura 42: (a) Visão frontal da placa Matrix Voice com os 18 LEDs RGB, (b) Montagem do display de toque de 5.5" e (c) Matrix Voice acoplada ao Raspberry PI

A placa ArbotiX-M é a responsável pelo controle dos dois servomotores que movimentam a cabeça do robô. Essa placa pode controlar até três servomotores e pode ser programada utilizando-se a IDE do Arduino. Ela se conecta ao Raspberry PI através de um cabo FTDI-USB que é usado tanto para a transferência do firmware da placa quanto para a comunicação serial. A placa ArbotiX-M também é usada para programar e definir os IDs dos servomotores. A Figura 43 mostra esses três componentes.

A placa ArbotiX-M possui três conexões destinadas ao controle de três servomotores. O tipo de conexão usada é a *DaisyChain*, isto é, os servomotores são ligados em série, de maneira sequencial. Para ter o controle individual de cada dispositivo, nesse tipo de conexão, os servomotores precisam ter IDs diferentes. Os servomotores AX-12A podem ter esses parâmetros definidos através da alteração dos seus firmwares. Esse processo de

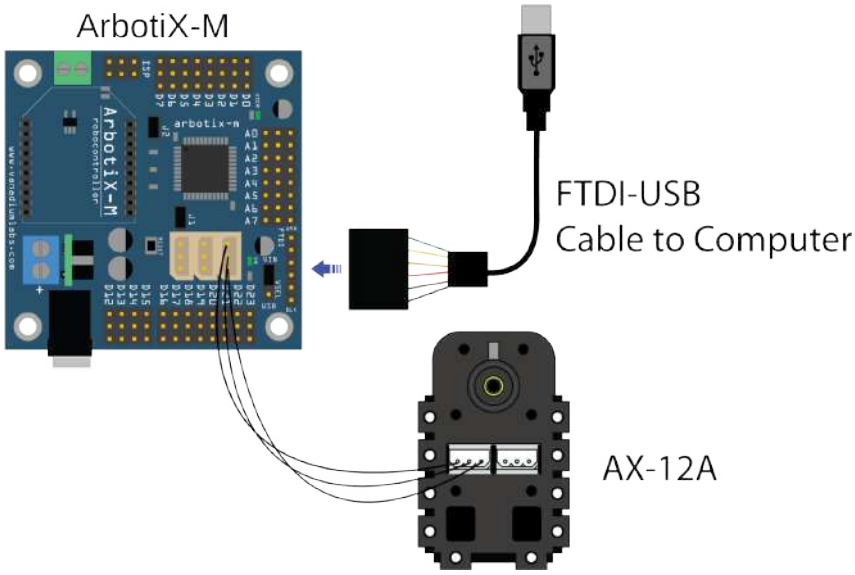

Figura 43: Placa ArbotiX-M, servomotor AX-12A e um cabo FTDI-USB

alteração do firmware do servomotores pode ser feito utilizando-se a placa ArbotiX-M.

A Figura 44 mostra a montagem final dos componentes internos do robô. O item (a) mostra os cabos que conectam os dois servomotores à placa ArbotiX-M. A imagem do centro, item (b), apresenta todo o sistema de placas e os dois servomotores conectados um ao outro. O item (c) mostra o momento em que o robô, com seu hardware totalmente instalado, será fechado e estará pronto para a instalação do seu software de controle.

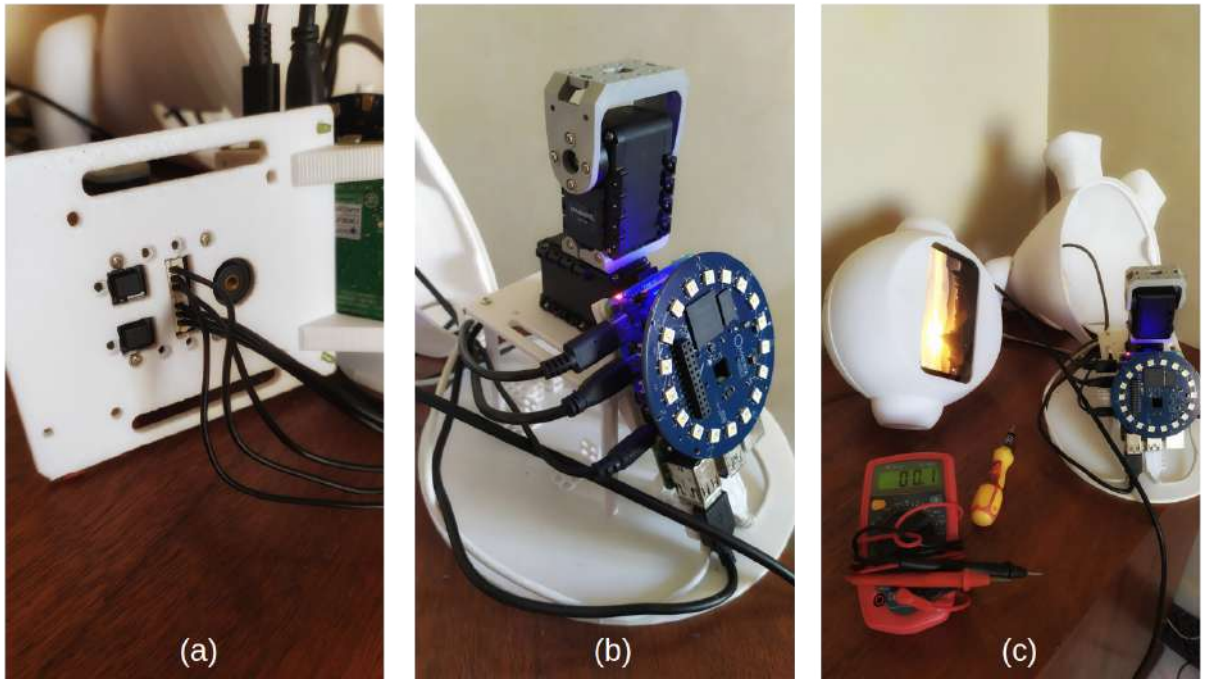

Figura 44: Fixação dos servomotores no suporte na parte interna do Eva

Para instalar o software de controle do EVA, é preciso acessar o terminal de dentro do sistema Raspbian no Raspberry PI, e baixar do repositório do robô no Github os fontes da aplicação. Logo após, seguindo os passos descritos na página do Github, será preciso instalar as dependências do software robô, como o NodeJS, por exemplo. Para que o EVA possa falar, transformando texto em fala, será necessário criar um conta no serviço IBM Watson e baixar uma chave em formato de texto que será usada pela aplicação do robô. O mesmo será necessário para que o robô possa converter fala em texto, utilizando a API do Google.

B.2 Configurando os IDs dos Servomotores AX-12A

O processo, que será apresentado nesta seção, usa a ArbotiX-M junto com a IDE do Arduino, sem utilizar qualquer software intermediário. A ideia é usar uma biblioteca para a ArbotiX-M, que se comunica com o servomotor, e pode escrever e ler nos seus registradores. A Listagem B.1 mostra o código do sketch[2] que deve ser importado e executado na IDE do Arduino. Este pequeno código faz com que o servomotor com ID=1 seja alterado para ID=2.

```
#include <ax12.h>
#include <BioloidController.h>

void setup()
{
  // param1 = ID do servo
  // param2 = n. do registrador
  // param3 = novo ID

  ax12SetRegister(1, 3, 2);
}

void loop() {}
```

Listagem B.1: Código para alterar o ID do servomotor AX-12A

O código a seguir pode ser usado para descobrir o ID de um servomotor conectado a ArbotiX-M e alterá-lo. Pode-se ver o código do sketch para a ArbotiX-M na Listagem

[2]Um sketch é o nome que é usado para se referir a um programa dentro da IDE do Arduino. É a unidade de código que é carregada e executada em uma placa do tipo Arduino.

B.2. Para descobrir o ID dos servomotores, basta fazer o *upload* do código para a placa ArbotiX-M e abrir o monitor da porta serial da IDE do Arduino, como indicado na Figura 45. Após a abertura da janela que monitora a porta serial na IDE, será apresentada uma saída semelhante à da Figura 46, exibindo os IDs dos servomotores conectados à placa ArbotiX-M.

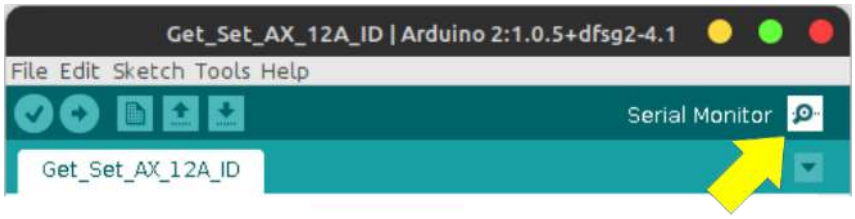

Figura 45: Monitor de porta serial da IDE do Arduino

Figura 46: Saída do programa que mostra o ID de um servomotor na janela de monitoramento da porta serial

```
/*
Autor: Marcelo Marques da Rocha
Laboratorio MidiaCom - Universidade Federal Fluminense (UFF)

Programa para encontrar o ID do servomotor AX-12A
Para alterar o ID de um servomotor:
1) Rode o programa para descobrir o ID do servomotor
```

```
2) Em seguida, descomente a linha com o comando ax12SetRegister(x, y, z);
3) Substitua os parametros da funcao de acordo com a sua necessidade*/

#include <BioloidController.h>
#include <ax12.h>

void setup()
{
  // param1 = ID do servo
  // param2 = n. do registrador
  // param3 = novo ID
  //ax12SetRegister(10, 3, 2);
  Serial.begin(9600);
}

void loop()
{
  delay(2000);
  Serial.println("Iniciando busca pelo ID do servomotor.");
  Serial.println("----------------------------------------------");
  Serial.println("ID\tEncontrado");
  Serial.println("----------------------------------------------");
  int servID = -1;
  for (int i=0; i<=255; i++)
  {
    delay(100);
    // param1 = ID do servo (a verificar)
    // param2 = n. do registrador
    // param3 = numero de bytes lidos
    servID = ax12GetRegister(i, 3, 1);
    Serial.print(i); // imprime a coluna do ID
    if (servID != -1){ // imprime o resultado da busca
      Serial.println("\tOK");
    } else {
      Serial.println("\t---");
    }
  }
  Serial.println("----------------------------------------------");
```

```
  Serial.println("Fim de busca.");
  while(1) ;
}
```

Listagem B.2: Código que mostra o ID de um servomotor

Após a correta configuração dos IDs dos servomotores, o firmware de controle do EVA deve ser enviado para a ArbotiX-M.

APÊNDICE C – ROTEIRO DO TESTE DE AVALIAÇÃO DA EVAML E DO EVASIM

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Universidade Federal Fluminense/IC - Instituto de Computação

Você está sendo convidado a participar de um estudo para avaliar a linguagem EvaML e o software simulador EvaSIM. Trata-se de uma pesquisa realizada pelo aluno de pós-graduação Marcelo Marques da Rocha, orientado pela professora Dra. Débora Christina Muchaluat Saade.

Se desejar continuar participando, continue a leitura e realize as atividades propostas. Muito obrigado!

O robô EVA (*Embodied Voice Assistant*) é um robô social e foi desenvolvido originalmente por pesquisadores do CICESE (*Centro de Investigación Científica y de Educación Superior de Ensenada*), em Baja California no México. O robô EVA é uma plataforma de robótica *open-source* destinada a auxiliar como ferramenta de apoio à pesquisa em Interação Humano-Robô (IHR). A imagem do robô EVA pode ser vista na Figura 47.

O objetivo deste teste é a criação de uma sessão interativa para o robô EVA usando os elementos definidos na linguagem EvaML, baseada em XML. O processo de desenvolvimento dos scripts para o robô EVA será usado como objeto de avaliação da linguagem EvaML. O teste será dividido em 3 tarefas menores que utilizarão os elementos da linguagem de forma progressiva. Ao finalizar as 3 tarefas, a sessão interativa deve ser similar à disponível neste link[1] (idioma do vídeo: inglês). Antes de começar a executar as tarefas

[1] https://www.youtube.com/watch?v=uDkwUEX8IeA

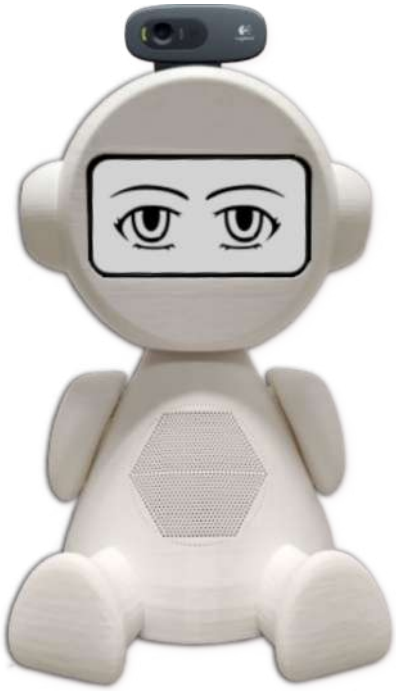

Figura 47: Robô EVA

sugeridas, recomenda-se ler o manual da linguagem EvaML[2] e instalar o parser EvaML e o simulador EvaSIM.

A seguir serão apresentados os passos para a configuração do ambiente adequado para o desenvolvimento do teste:

1. **IDE Visual Studio Code (Linux ou Windows)**

   Você deve baixar e instalar a IDE Visual Studio Code a partir deste endereço: https://code.visualstudio.com/.

2. **XML Language Support by Red Hat instalado no VScode**

   A linguagem EvaML é uma linguagem baseada em XML e exige uma formatação adequada da sua estrutura, elementos e atributos. A utilização do VSCode com o plugin da Red Hat que dá suporte à linguagens baseadas em XML é uma ferramenta de apoio insdispensável para o correto desenvolvimento das tarefas.

   A instalação é muito simples! A Figura 48, através das setas numeradas, indica os elementos da interface do VSCode que devem ser acessados durante o processo de instalação do plugin XML. Para iniciar a instalação, abra a visualização de "Extensões"clicando no ícone "Extensões"**(1)** na barra de atividades que fica na

[2]https://github.com/midiacom/eva-robot/blob/master/EvaML-Reference-Manual/EvaML-Reference-Manual.pdf

lateral esquerda do VSCode ou utilizando o atalho do teclado (Ctrl+Shift+X). Você verá uma lista das extensões mais populares do VSCode no VSCode Marketplace. Na barra de pesquisa **(2)**, digite XML Language Suporte by Red Hat. Para instalar a extensão, selecione-a na lista e clique no botão Instalar **(3)**.

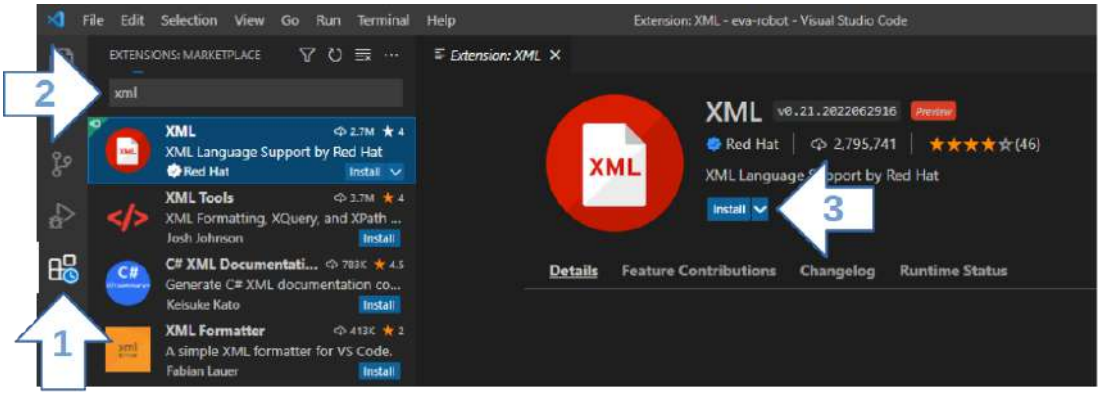

Figura 48: Instalando o plugin XML no VSCode

A partir deste momento o VSCode dará suporte à codificação da linguagem usando como base as regras básicas da linguagem XML e também com base no arquivo XML Schema que define a gramática da linguagem EvaML . Você terá suporte na definição da estrutura do documento, utilização correta dos elementos disponíveis em cada seção do documento e poderá utilizar o recurso de autocompletar do VSCode através do atalho do teclado (Ctrl+Barra de espaço) durante a entrada dos comandos e atributos da linguagem.

3. **Parser EvaML e o simulador EvaSIM**

A linguagem EvaML é uma linguagem para o desenvolvimento de sessões interativas para o robô EVA. Como não é possível neste momento ter um robô físico montado para os testes da linguagem, o código escrito em EvaML será testado usando-se o simulador do robô EVA chamado EvaSIM. O EvaSIM, assim como o robô físico, não executa diretamente um código escrito na linguagem EvaML, o simulador é capaz de executar o código resultante do processo de parser da linguagem EvaML. O código do parser da linguagem se encontra no arquivo "eva_parser.py".

Como dito anteriormente, o simulador EvaSIM é capaz de executar o código resultante do processo de parser da linguagem EvaML e é esse código que deve ser importado e executado no simulador .

Para ter acesso ao simulador EvaSIM e ao parser EvaML baixe o arquivo zip na versão correta referente ao seu sistema operacional a partir deste link[3]. Você será

[3]https://github.com/midiacom/eva-robot/tree/master/EvaSIM%20Testing%20Version

direcionado a uma página contendo as instruções para o download, execução e o teste inicial do EvaSIM.

Para executar o parser EvaML você precisa ter o Python instalado na sua máquina junto com o módulo xmlschema. Para isso, siga os passos descritos a seguir:

i. Caso seu sistema operacional seja o Windows, vá até a página de downloads e baixe a versão do Pyhton adequada ao seu sistema operacional através deste link[4]. Clique no arquivo executável e siga o processo de instalação. Escolha a instalação customizável, como indicado na imagem à esquerda na Figura 49, e em seguida certifique-se que a opção pip esteja selecionada, como indicado na imagem mais à direita, ainda na Figura 49. Siga os diálogos apresentados e finalize o processo instalação do Python.

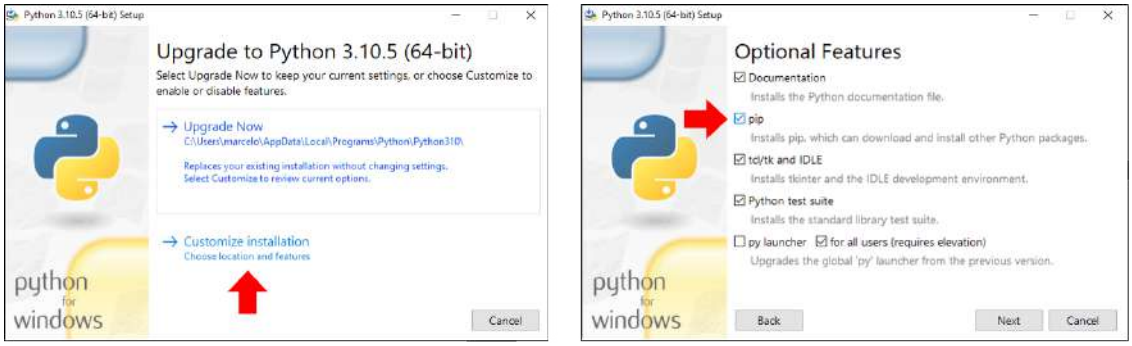

Figura 49: Instalando o Python no Windows

* No sistema Ubuntu Linux o Python já vem instalado por padrão sendo apenas necessário instalar o seu gerenciador de pacotes. Para isso, digite no seu terminal Linux o seguinte comando com permissões de administrador: **apt install python3-pip**.

ii. Agora será preciso instalar os módulos Python xmlschema e requests, para isso, abra um terminal no seu sistema operacional (Windows ou Linux), digite e execute os seguintes comandos: **pip install xmlschema** (para instalar o módulo xmlschema) e **pip install requests** (para a instalação do módulo requests).

Para que a experiência no processo de desenvolvimento da tarefa seja a melhor possível, será apresentada a seguir uma sequência de passos que podem servir como um guia para a execução das tarefas. Nós partiremos do pressuposto que você já tem o ambiente de desenvolvimento corretamente configurado como indicado anteriormente.

[4]https://www.python.org/downloads/

- Você deve abrir o VSCode e no menu "Arquivo" clicar em importar "pasta". Em seguida vá até a pasta "eva_sim" e a selecione. O VSCode mostrará o conteúdo da pasta "eva_sim"com todos os seus arquivos. Caso você não esteja vendo a árvore do diretório no VSCode, use o atalho (Ctrl+Shift+E).

- Observando os arquivos através do explorer do VSCode, você deve clicar no arquivo de template nomeado "my_script_file.xml". Esse arquivo contém uma estrutura básica do documento no padrão EvaML. É neste arquivo que você deve entrar o código referente a execução das tarefas propostas neste teste. Você deve aproveitar o recurso de autocompletar do VSCode! Ele o ajudará na utilização correta dos elementos da linguagem. Por exemplo, ao digitar o caractere «" você verá uma lista dos elementos da linguagem que podem ou devem ser utilizados em cada seção do seu documento EvaML. Da mesma maneira, ao entrar com um atributo também será apresentada uma lista com as opções para cada atributo. Quando há algum erro na estrutura do documento ou há algum erro de digitação que comprometa a boa formação do documento, a IDE procura indicar, da maneira mais precisa possível, em que lugar do documento se encontra o erro. Ela faz isso, geralmente, sublinhando o local onde o erro está ocorrendo com a cor vermelha.

- Para testar o código que você escreveu, como já foi dito, será utilizado o simulador EvaSIM, mas para isso, é preciso transformar o código do script que você escreveu em um código compatível com o simulador. Esse processo é feito utilizando-se o parser EvaML. Para ter acesso à linha de comando crie um novo "terminal" no VSCode através do menu "Terminal" ou através do atalho (Ctrl+Shift+'). Um terminal se abrirá na parte inferior da IDE e já estará no diretório de trabalho, ou seja, no diretório referente à pasta "eva_sim". Para executar o parser do código digite no terminal o comando "python eva_parser.py my_script_file.xml -c". Se o seu código estiver no padrão da linguagem EvaML, você verá no terminal, a indicação de que as 4 etapas do parsing executaram corretamente e o parsing terá criado o arquivo "script01_EvaML.xml". É este o arquivo que deverá ser importado no EvaSIM na próxima etapa.

- Durante a etapa de configuração do ambiente adequado (como foi sugerido) você deve ter baixado e executado o simulador EvaSIM. Você também deve ter ativado o simulador clicando no botão "PowerOn" no menu de botões do mesmo. Para que o simulador execute o código gerado pelo parsing, você precisa importá-lo no EvaSIM. Para isso, você deve cliquar no botão "Import File" no menu de botões

do simulador e selecionar, através da janela de diálogo de abertura de arquivo, o arquivo "script01_EvaML.xml".

- O próximo passo é rodar o script carregado na memória do EvaSIM e isso deve ser feito clicando no botão "Run" do simulador. Durante a execução do script, observe as mensagens no emulador de terminal do EvaSIM, elas podem ajudá-lo na verificação da lógica do seu script. Acompanhe também as tabelas de memória, que ficam à direita na interface do simulador, nelas você pode visualizar o conteúdo da memória do EvaSIM. O simulador foi programado para emitir algumas mensagens de alerta e de erro em tempo de execução. Caso ocorra algum problema na execução do seu código, volte para o VSCode e procure corrigi-lo, salvando-o e voltando novamente aos processos de parsing, importação e execução do script.

Seguindo as dicas indicadas até aqui você será capaz de executar as tarefas propostas a seguir. Boa sorte!

Tarefa 1

Desenvolva uma sessão para o robô EVA que faça ele se apresentar para o usuário. Esta sessão deve usar elementos de luz e som, deve utilizar o display do robô para expressar suas emoções através do olhar e deve utilizar o recurso de *Text to Speech* (TTS) do robô, permitindo que o robô fale. Marque o tempo que levará para executar esta tarefa.

Primeiro, defina o nome do script através do atributo name do elemento <evaml>. Você pode manter o mesmo nome que já veio dentro do template. Em seguida, você deve definir a voz do robô usando o comando <voice>. Para ter acesso a algumas opções de vozes disponíveis, consulte a Tabela 2.3, na Seção 2.3.1 do manual. Certifique-se de ativar os efeitos de luz e áudio na seção <settings>. A sequência de ações a serem executadas pelo robô é a seguinte.

Usando o comando <evaEmotion>, faça com que o olhar do robô fique com a expressão neutra. Com o comando <light>, faça com que o robô acenda a luz na cor branca. Vá até a pasta “sonidos”, que se encontra no diretório do parser EvaML e escolha um arquivo de áudio para ser tocado pelo robô. Utilizando o comando <audio> toque o arquivo de áudio escolhido como um som de abertura, antes do texto de apresentação do robô. Lembre-se de usar o atributo block=”true” para fazer com que o comando seguinte espere pelo fim do play do arquivo de áudio.

Agora, faça com que o robô acenda a luz amarela e fique com a expressão de alegria. Em seguida, o robô deve se apresentar falando um texto. Esse texto deve conter 3 frases distintas, que serão selecionadas aleatoriamente a cada execução do script. Para isso, use o caractere "/" como separador das frases. Agora, acenda a luz rosa, faça o robô ficar com o olhar neutro e finalize o script, fazendo com que a interação fique em pausa por 2 segundos, para isso, use o comando <wait>. Salve o script, execute o parser EvaML, importe e rode o código no simulador EvaSIM. Fique atento pois o atributo name do elemento raiz <evaml> do arquivo de template é "script01", portanto, após a execução do parser será gerado o arquivo "script01_EvaML.xml" e este é o arquivo que deve ser importado no simulador. Se o script não executar como planejado no EvaSIM, revise o seu código XML para resolver qualquer problema. Salve o código XML final da tarefa 1, pois será necessário utilizá-lo para a tarefa 2. Lembre-se de marcar o tempo que levará para executar esta tarefa.

Tarefa 2

Dando continuidade ao script criado na tarefa 1, o objetivo da tarefa 2 é fazer com que o robô interaja com o usuário via voz para obter o seu nome e perguntar se ele deseja jogar um jogo. Marque o tempo que levará para executar esta tarefa.

Primeiro, o robô deverá obter o nome do usuário. Para isso ele deve fazer a pergunta falando e em seguida obter a resposta do usuário. Em seguida, o robô deve saudar o usuário pelo nome obtido anteriormente e dizer que vai apresentar suas emoções. O robô deverá apresentar suas emoções com o olhar, seguido de uma descrição verbal da emoção apresentada. Apresente as 3 emoções (alegria, tristeza e raiva). Após a apresentação das 3 emoções faça com que o olhar do robô fique com expressão neutra.

Vamos, ainda nesta tarefa, fazer a introdução do *Jogo da Imitação*, um Jogo Sério que usa a capacidade de reconhecimento de expressões faciais do EVA. Este jogo será implementado na tarefa 3.

Seguindo com a codificação da sequência de ações, faça com que o robô pergunte ao usuário se ele quer jogar o Jogo da Imitação. A resposta deve ser sim ou não. Obtenha a resposta do usuário. Usando o comando <switch var="$">, insira dois comandos <case> para tratar as respostas sim/não do usuário. Se o usuário responder sim, faça o robô expressar um olhar de alegria e falar uma frase de entusiasmo por conta do usuário ter aceitado jogar o jogo. Caso o usuário diga não, faça o robô expressar um olhar de tristeza despedindo-se do usuário, falando uma frase lamentando o ocorrido e apagando a

luz. Após a fala, ainda no bloco <case>, que trata a resposta não, insira o comando que interrompe a execução do script. Você pode fechar o bloco <switch>, salvar o código e executar o parser EvaML e rodar no simulador EvaSIM. Se tudo estiver certo, você pode partir para a tarefa 3, caso contrário, revise o seu código XML para resolver qualquer problema. Salve o código XML final da tarefa 2, pois será necessário utilizá-lo para a tarefa 3. Lembre-se de marcar o tempo que levará para executar esta tarefa.

Observação 1: Para entender melhor a utilização do caractere $, veja a Seção 2.3.4 do manual e observe a Figura 2.2.

Observação 2: No EvaSIM, use o emulador de terminal para acompanhar as ações sendo executadas pelo "robô", e verifique as tabelas de variáveis para conferir os valores de retorno dos comandos <random> e <listen> que podem ser referenciados utilizando-se o caractere $.

Tarefa 3

A última tarefa implementa o Jogo da Imitação. O jogo deverá funcionar da seguinte maneira. O robô deve explicar ao usuário que irá apresentar algumas emoções. Ao todo, serão 3 emoções apresentadas (tristeza, alegria ou raiva) e elas devem ser apresentadas aleatoriamente. Para as emoções de tristeza, alegria e raiva, acenda a lâmpada nas cores azul, verde e vermelha, respectivamente. Em seguida, o robô deve pedir ao usuário para imitá-lo. O robô, então, deve detectar a expressão facial do usuário e verificar se o usuário conseguiu imitá-lo. Este processo deve ser repetido 3 vezes e, ao final, o robô deve dizer quantas vezes o usuário imitou sua expressão facial corretamente. Marque o tempo que levará para executar esta tarefa.

Para detectar a expressão facial do usuário, use o comando <userEmotion>. Caso o usuário tenha imitado corretamente, o robô deve expressar alegria, falar uma frase de congratulação e um ponto deve ser adicionado ao score do jogo. No caso do usuário não conseguir imitar o robô, o EVA deve expressar tristeza e falar uma frase dizendo que o usuário errou. Em seguida, o robô reinicia o processo, apresentando outra emoção selecionada aleatoriamente. Ao todo, devem ser apresentadas 3 emoções, ou seja, o processo se repete 3 vezes. Após as 3 rodadas de emoções, o robô deve expressar alegria, apresentar (falando) o score do jogo e por fim, falar um texto se despedindo do usuário. O nome do usuário, obtido na tarefa 2, deve ser citado no texto. Para isso use o caractere $ seguido do índice que pode acessar o segundo elemento armazenado na memória do EVA.

Observe que duas variáveis devem ser criadas, a primeira deve conter o score do jogo. A segunda deve controlar o número de rodadas do jogo. Fique atento aos trechos de códigos que podem se repetir nos 3 fluxos aleatórios. Você pode definir algumas macros que podem, além de reduzir a quantidade de código digitado, deixar o script mais legível. As partes do texto sublinhadas indicam as sequências de ações que podem ser definidas em 3 macros.

Após a finalização da codificação da tarefa em XML, execute o parser EvaML, importe o código no simulador EvaSIM e verifique se o código se comporta como o esperado. Se o script não executar como planejado no EvaSIM, revise o seu código XML para resolver qualquer problema. Salve o código XML final da tarefa 3, pois será necessário fazer upload deste código posteriormente. Lembre-se de marcar o tempo que levará para executar esta tarefa.

Observação 3: O arquivo que deve ser enviado é o arquivo "my_script_file.xml".

Questionário de Avaliação

Ao finalizar as 3 tarefas, responda o questionário sobre a linguagem EvaML e o simulador EvaSIM disponível através do link[5].

[5]https://forms.gle/zg3fBnhEbtWFZjnF8

APÊNDICE D – QUESTIONÁRIO: EVAML E EVASIM - TESTE DE USABILIDADE

EvaML and EvaSIM - Usability Test (Teste de Usabilidade)

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Universidade Federal Fluminense/IC - Instituto de Computação

Você está sendo convidado a participar de um estudo para avaliar a linguagem EvaML e o software simulador EvaSIM. Trata-se de uma pesquisa realizada pelo aluno de pós-graduação Marcelo Marques da Rocha, orientado pela professora Dra. Débora Christina Muchaluat Saade.

Para a realização do estudo, é necessário que você responda a cada uma das seções deste questionário e aceite eletronicamente participar da pesquisa como voluntário. Os dados respondidos serão confidenciais e você poderá desistir do estudo a qualquer momento.

Não existem respostas "certas" ou respostas "erradas". Sua opinião pessoal é o que conta!

Obrigado pela sua participação e colaboração.

FREE AND INFORMED CONSENT TERM

Fluminense Federal University / IC - Institute of Computing

You are being invited to participate in a study to evaluate the EvaSIM simulator software. This is a research carried out by the graduate student Marcelo Marques da Rocha, supervised by Professor Dr. Débora Christina Muchaluat Saade.

In order to carry out the study, it is necessary that you answer each of the sections of this questionnaire and electronically accept to participate in the research as a volunteer. The data answered will be confidential and you can withdraw from the study at any time.

There are no "right" answers or "wrong" answers. Your personal opinion is what counts!

Thank you for your participation and collaboration.

*Obrigatório

1. This questionnaire is anonymous. Do you agree to participate? (Este questionário é anônimo. Você concorda em participar?) *

   Marcar apenas uma oval.

   ◯ Yes (Sim)

Questionnaire - EvaML

2. Nickname (Apelido) *

3. Age (Idade) *

4. Indicate your experience with the EVA robot. (Indique seu nível de experiência com o robô EVA.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Never used (Nunca usei)	◯	◯	◯	◯	◯	Expert (Muito experiente)

5. Indicate your experience with XML. (Indique seu nível de experiência com XML.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Never used (Nunca usei)	◯	◯	◯	◯	◯	Expert (Muito experiente)

6. Indicate your experience in programming. (Indique seu nível de experiência em programação.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Beginner (Iniciante)	◯	◯	◯	◯	◯	Expert (Muito experiente)

7. Using EvaML, I can clearly identify the components of my interactive session for the EVA robot. (Usando EvaML, eu consegui identificar claramente os componentes da minha sessão interativa para o robô EVA.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

8. I understand the functionality of the EvaML language elements. (Eu entendo a funcionalidade dos elementos da linguagem EvaML.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

9. Did you successfully complete step 1 of the test? (Você concluiu com sucesso a etapa 1 do teste?) *

Marcar apenas uma oval.

◯ Yes (Sim)
◯ No (Não)

10. How long did it take you to complete step 1 of the test? (Quanto tempo você levou para concluir a etapa 1 do teste?)

11. How easy was it to implement step 1 of the test with the elements of the EvaML language?(O quão fácil foi implementar a etapa 1 do teste com os elementos da linguagem EvaML?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Very difficult (Muito difícil)	◯	◯	◯	◯	◯	Very easy (Muito fácil)

12. Did you successfully complete step 2 of the test? (Você concluiu com sucesso a etapa 2 do teste?) *

Marcar apenas uma oval.

◯ Yes (Sim)
◯ No (Não)

13. How long did it take you to complete step 2 of the test? (Quanto tempo você levou para concluir a etapa 2 do teste?)

14. How easy was it to implement step 2 of the test with the elements of the EvaML language? (O quão fácil foi implementar a etapa 2 do teste com os elementos da linguagem EvaML?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Very difficult (Muito difícil)	◯	◯	◯	◯	◯	Very easy (Muito fácil)

15. Did you successfully complete step 3 of the test? (Você concluiu com sucesso a etapa 3 do teste?) *

Marcar apenas uma oval.

◯ Yes (Sim)
◯ No (Não)

16. How long did it take you to complete step 3 of the test? (Quanto tempo você levou para concluir a etapa 3 do teste?)

17. How easy was it to implement step 3 of the test with the elements of the EvaML language? (O quão fácil foi implementar a etapa 3 do teste com os elementos da linguagem EvaML?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Very difficult (Muito difícil)	◯	◯	◯	◯	◯	Very easy (Muito fácil)

18. Do you consider the EvaML language verbose? (Você considera a linguagem EvaML verbosa?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Not verbose (Não verbosa)	◯	◯	◯	◯	◯	Very verbose (Muito verbosa)

19. Do you consider that the EvaML language greatly restricts the order of creation of the elements of your application? (Você considera que a linguagem EvaML restringe muito a ordem de criação dos elementos da sua aplicação?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Restricts a little (Restringe pouco)	◯	◯	◯	◯	◯	Restricts a lot (Restringe muito)

20. Do you consider that the EvaML language has many hidden dependencies between its elements? (Você considera que a linguagem EvaML possui muitas dependências ocultas entre seus elementos?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Doesn't have (Não possui)	◯	◯	◯	◯	◯	Has many (Possui muitas)

21. Do you think that the syntax of the EvaML language induces you to make mistakes when using its elements? (Você acha que a sintaxe da linguagem EvaML induz você a cometer erros quando usa seus elementos?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Doesn't induce (Não induz)	◯	◯	◯	◯	◯	Induces a lot of mistakes (Induz muitos erros)

22. I think that EvaML language elements with similar semantics are expressed by similar syntaxes. (Eu acho que os elementos da linguagem EvaML com semânticas similares são expressos por sintaxes similares.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

23. Did modifying one element of your code require modifying other elements? (A modificação de um elemento do seu código exigiu a modificação de outros elementos?) *

Marcar apenas uma oval.

	1	2	3	4	5	
Didn't require (Não exigiu)	◯	◯	◯	◯	◯	Required a lot (Exigiu muito)

24. Indicate the level of difficulty in using each element of the EvaML language. *
(Indique o nível de dificuldade no uso de cada elemento da linguagem EvaML.)

Marcar apenas uma oval por linha.

	I didn't use it (Não usei)	1 Very easy (Muito fácil)	2 Easy (Fácil)	3 Moderate (Moderada)	4 Difficult (Difícil)	5 Very difficult (Muito difícil)
<voice>						
<random>						
<wait>						
<talk>						
<stop>						
<light>						
<goto>						
<userEmotion>						
<evaEmotion>						
<useMacro>						
<listen>						
<audio>						
<led>						
<motion>						
<counter>						
<switch>						
<macro>						
<case>						
<default>						

25. Did you have difficulty understanding the structure of a script in the EvaML language? (Você teve dificuldade de entender a estrutura de um script na linguagem EvaML?) *

Marcar apenas uma oval.

	1	2	3	4	5	
No difficulty (Nenhuma dificuldade)	◯	◯	◯	◯	◯	Much difficulty (Muita dificuldade)

26. Give a final grade to the EvaML language. (Dê uma nota final para a linguagem EvaML.) *

Marcar apenas uma oval.

1	2	3	4	5
◯	◯	◯	◯	◯

27. Make comments and give suggestions so that we can improve the EvaML language. (Faça comentários e dê sugestões para que possamos melhorar a linguagem EvaML.) *

28. Upload your XML code for the task performed. (Faça o upload do seu código XML da tarefa realizada.) *

Arquivos enviados:

Questionnaire - EvaSIM

29. Professional background and occupation (Área de atuação) *

30. Years of professional experience (Anos de atuação na área) *

31. I think that I would like to use EvaSIM frequently. (Eu acho que eu gostaria de usar o EvaSIM com frequência.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

32. I think EvaSIM would help me program EVA scripts. (Acho que o EvaSIM me ajudaria a programar scripts para o robô EVA.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

33. I found EvaSIM unnecessarily complex. (Achei o EvaSIM desnecessariamente complexo.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

34. I thought EvaSIM was easy to use. (Achei o EvaSIM fácil de usar.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

35. I think that I would need the support of a technical person to be able to use EvaSIM. (Acho que precisaria do apoio de um técnico para poder usar o EvaSIM.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

36. I found the various functions in EvaSIM were well integrated. (Achei que as várias funções do EvaSIM estavam bem integradas.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

37. I thought there was too much inconsistency in EvaSIM. (Achei que havia muita inconsistência no EvaSIM.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

38. I would imagine that most people would learn to use EvaSIM very quickly. (Eu imagino que a maioria das pessoas aprenderia a usar o EvaSIM muito rapidamente.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

39. I found EvaSIM very cumbersome to use. (Achei o EvaSIM muito complicado de usar.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

40. I felt very confident using EvaSIM. (Eu me senti muito confiante usando o EvaSIM.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

41. I needed to learn a lot of things before I could get going with EvaSIM system. (Eu precisaria aprender muitas coisas antes de começar a usar o EvaSIM.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

42. Messages presented by EvaSIM were difficult to understand. (Uma mensagem apresentada pelo EvaSIM foi difícil de entender.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

43. It was easy to install EvaSIM software components. (Foi fácil instalar os componentes de software do EvaSIM.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

44. EvaSIM interface elements represented the robot's functionalities well. (Os elementos de interface do EvaSIM representaram bem as funcionalidades do robô.) *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly disagree (Discordo fortemente)	◯	◯	◯	◯	◯	Strongly agree (Concordo fortemente)

45. Make comments and give suggestions so that we can improve EvaSIM simulator. (Faça comentários e dê sugestões para que possamos melhorar o simulador EvaSIM.) *

Este conteúdo não foi criado nem aprovado pelo Google.

Google Formulários