



Pós-Graduação em Ciência da Computação

**“Persona-AIML: Uma Arquitetura para Desenvolver
Chatterbots com Personalidade”**

Por

Adjamir de Moura Galvão

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, MARÇO/2003



UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ADJAMIR DE MOURA GALVÃO

“Persona-AI ML: Uma Arquitetura para Desenvolver *Chatterbots*
com Personalidade”

*ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA
UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO
REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE
EM CIÊNCIA DA COMPUTAÇÃO.*

ORIENTADORA: FLÁVIA DE ALMEIDA BARROS

RECIFE, MARÇO/2003

*Aos meus pais, Ana e Francisco, e
especialmente, a Geórgia Maria C. S. “Galvão”.*

Agradecimentos

- Primeiramente a Deus, fonte inesgotável de força e inspiração;
- A minha família que sempre me incentivou a progredir pessoalmente e profissionalmente;
- A minha namorada, Geórgia Maria, que com seu apoio incondicional esteve sempre ao meu lado, especialmente nos momentos mais difíceis.
- A minha orientadora, Flávia Barros, pela orientação, dedicação e incentivo durante todo o período de mestrado;
- Ao professor André Neves, pelas sugestões ao longo deste trabalho;
- A todos os colegas que fiz durante o mestrado, especialmente a Ayla, Ellen, Hélio, Bruno e Ismênia, pelo apoio e amizade durante esse período;

Pouco conhecimento faz com que as criaturas se sintam orgulhosas. Muito conhecimento, que se sintam humildes. É assim que as espigas sem grãos erguem desdenhosamente a cabeça para o céu, enquanto as cheias a baixam para a terra, sua mãe.

Leonardo da Vinci

Resumo

Chatterbots são sistemas que se comunicam com usuários em linguagem natural, e podem ser utilizados como interfaces nas mais variadas aplicações (e.g., comércio eletrônico, ensino a distância, entre outras). Estudos recentes têm demonstrado a importância de explorar o comportamento social do usuário perante o computador, destacando-se o papel da personalidade. Além disso, a personalidade também é vista como um componente importante no processo de tomada de decisão.

Dentre as ferramentas e linguagens para a criação de *chatterbots*, AIML (*Artificial Intelligence Markup Language*) é, atualmente, uma das mais bem sucedidas abordagens. Entretanto, AIML não oferece suporte para a modelagem e a implementação de personalidade em *chatterbots*.

Esta dissertação apresenta a arquitetura Persona-AIML, uma extensão do AIML original, que possibilita o desenvolvimento de modelos de personalidade para *chatterbots*. Na arquitetura proposta, a personalidade é composta pelos seguintes elementos: traços, atitudes, humor, emoções e estados físicos.

Neste trabalho, *chatterbots* são modelados como agentes racionais e utilizam uma base de regras para descrever sua personalidade (tendo como objetivo a modularidade, extensibilidade e reusabilidade). O protótipo desenvolvido utilizou um modelo de personalidade baseado no Modelo dos “Cinco Grandes Fatores”. Entretanto, a arquitetura proposta permite a utilização de outros modelos de personalidade. Essa arquitetura foi implementada em Java, e os testes revelaram resultados satisfatórios do protótipo.

Abstract

Chatterbots are systems that communicate with users in natural language, and they can be used as interfaces in the most varied applications (e.g., electronic commerce, distance learning, among others). Recent studies have demonstrated the importance of considering the user's social behavior towards the computer, highlighting the role of personality. Besides, personality is also seen as an important component in the decision making process.

Among the tools and languages for the creation of chatterbots, AIML (*Artificial Intelligence Markup Language*) is, currently, one of the most successful approaches. However, AIML does not offer support for the modeling and implementation of personality in chatterbots.

This dissertation presents the architecture Persona-AIML, an extension of the original AIML, which enables the development of personality models for chatterbots. In the proposed architecture, personality is composed by the following elements: traits, attitudes, moods, emotions and physical states.

In this work, chatterbots are modeled as rational agents, and use a rules base to describe their personality (aiming at modularity, extensibility and reusability). The developed prototype used a personality model based on the "Big Five factors" model. However, the proposed architecture is able to deal with others models for personality. This architecture was implemented in Java, and the tests revealed satisfactory results of the prototype.

Sumário

1. Introdução	15
1.1 Objetivos.....	16
1.2 Organização da Dissertação	17
2. <i>Chatterbots</i>: Características e Aplicações	18
2.1 Aplicações de <i>Chatterbots</i>	18
2.2 Agentes Inteligentes e <i>Chatterbots</i>	22
2.2.1 Arquiteturas de Agentes	23
2.3 Breve Histórico dos <i>Chatterbots</i>	26
2.3.1 ELIZA	27
2.3.2 PARRY	30
2.3.3 JULIA.....	31
2.3.4 JFRED e a Série ALBERT.....	33
2.3.5 ALICE e AIML.....	34
2.3.6 Outros <i>Chatterbots</i>	35
2.3.7 Agentes Conversacionais Incorporados	36
2.4 Estudo Comparativo	37
2.5 A Linguagem AIML	39
2.6 Conclusões.....	45
3. Personalidade em Agentes Inteligentes	48
3.1 O Papel da Personalidade em Agentes Inteligentes.....	48
3.2 Personalidade Segundo a Psicologia	50
3.2.1 Elementos	51
3.2.2 Teorias e Modelos	54
3.3 Modelos Computacionais de Personalidade em Agentes.....	59
3.3.1 Abordagens Utilizadas.....	59
3.3.2 Exemplos de Agentes Inteligentes com Personalidade.....	60
3.3.3 Exemplos de <i>Chatterbots</i> com Personalidade.....	62
3.4 Conclusões	63
4. A Arquitetura Persona-AIML	65
4.1 Princípios Básicos	65

4.2 Arquitetura	67
4.2.1 O Componente de Personalidade	69
4.2.2 O Componente de Raciocínio	72
4.3 Criação de Modelos de Personalidade	75
4.3.1 Pontos Comuns	76
4.3.2 Principais Diferenças	76
4.3.3 Regras	77
4.3.4 Elementos de Personalidade	79
4.3.5 Mecanismo de Validação	85
4.4 Passos para a Criação de <i>Chatterbots</i> com Personalidade	86
4.5 Conclusões	87
5. Implementação e Protótipo	89
5.1 Decisões de Projeto	89
5.2 A Base de Categorias	90
5.3 O Componente de Personalidade	92
5.3.1 Crenças e Ações: <i>Tags <beliefs></i> e <i><actions></i>	93
5.3.2 Regras: <i>Tags <events></i> e <i><behavior></i>	94
5.3.3 Elementos de Personalidade: <i>Tag <personality></i>	96
5.4 Outras <i>Tags</i> Adicionadas	101
5.5 Conversor Persona-AIML para AIML Padrão	109
5.6 Protótipo Testado	110
5.6.1 Construção da Base de Categorias	111
5.6.2 Definição do Modelo de Personalidade	111
5.6.3 Definição dos Comportamentos e Elementos de Personalidade	112
5.6.4 Testes	113
5.6.5 Análise dos Resultados	113
5.7 Conclusões	114
6. Conclusão	116
6.1 Contribuições	116
6.2 Trabalhos Futuros	117
7. Referências Bibliográficas	119
Apêndice A – Logs de Diálogo	127

Apêndice B – Componente de Personalidade	130
Arquivo de Personalidade.....	130
Mecanismo de Validação.....	135
Apêndice C – Exemplo de Categorias	138

Lista de Figuras

FIGURA 2.1 – ARQUITETURA GERAL DE UM AGENTE INTELIGENTE	22
FIGURA 2.2 – AGENTE REATIVO (RUSSEL E NORVIG, 1995).....	24
FIGURA 2.3 – AGENTE REATIVO COM ESTADO INTERNO (RUSSEL E NORVIG, 1995).....	25
FIGURA 2.4 – AGENTE BASEADO EM OBJETIVOS (RUSSEL E NORVIG, 1995)	25
FIGURA 2.5 – AGENTE OTIMIZADOR (RUSSEL E NORVIG, 1995)	26
FIGURA 2.6 – TRECHO DE DIÁLOGO COM ELIZA.....	28
FIGURA 2.7 – EXEMPLO DE REGRA DE ELIZA	28
FIGURA 2.8 – TRECHO DE DIÁLOGO COM PARRY	31
FIGURA 2.9 – TRECHO DE DIÁLOGO COM JULIA.....	32
FIGURA 2.10 – TRECHO DE DIÁLOGO COM JIM (ALBERT).....	33
FIGURA 2.11 – TRECHO DE DIÁLOGO COM ALICE	35
FIGURA 2.12 – IMAGENS DO HUMANO VIRTUAL	37
FIGURA 2.13 – SINTAXE BÁSICA DE UMA CATEGORIA AIML	39
FIGURA 2.14 – REGRA DE ELIZA REESCRITA UTILIZANDO AIML.....	40
FIGURA 2.15 – EXEMPLO DE USO DAS TAGS <GET> E <SET>	40
FIGURA 2.16 – EXEMPLO DE USO DA TAG <THINK>.....	41
FIGURA 2.17 – EXEMPLO DE FUNCIONAMENTO DA TAG <INPUT>.....	41
FIGURA 2.18 – EXEMPLO UTILIZANDO A TAG <THAT>.....	42
FIGURA 2.19 – EXEMPLO UTILIZANDO A TAG <TOPIC>.....	43
FIGURA 2.20 – EXEMPLO UTILIZANDO A TAG <CONDITION>	44
FIGURA 2.21 – EXEMPLO UTILIZANDO A TAG <SRAI>.....	45
FIGURA 3.1 – DURAÇÃO DE EMOÇÕES, HUMOR E SENTIMENTOS.....	52
FIGURA 3.2 – EXEMPLO DE RESPOSTAS VARIADAS UTILIZADAS PELOS CHATTERBOTS.....	63
FIGURA 4.1 – ARQUITETURA PERSONA-AIML	68
FIGURA 4.2 – COMPONENTE DE PERSONALIDADE	70
FIGURA 4.3 – DIAGRAMA DE CLASSES DO AGENTE PERSONA-AIML.....	71
FIGURA 4.4 – COMPONENTE DE RACIOCÍNIO DO AGENTE	72
FIGURA 4.5 – MÓDULO DE CONSTRUÇÃO DA RESPOSTA	74
FIGURA 4.6 – DIAGRAMA DE CLASSES PARA AS REGRAS	77
FIGURA 4.7 – DIAGRAMA DE CLASSES PARA AS REGRAS DE COMPORTAMENTO	78
FIGURA 4.8 – DIAGRAMA DE CLASSES PARA OS EVENTOS	78
FIGURA 4.9 – EXEMPLO DE PSEUDO-REGRA DE COMPORTAMENTO	79
FIGURA 4.10 – DEFINIÇÃO DA CLASSE <i>ABSTRACTELEMENT</i>	80

FIGURA 4.11 – DIAGRAMA DE OBJETOS PARA O ESTADO FÍSICO	80
FIGURA 4.12 – DEFINIÇÃO DA CLASSE <i>ABSTRACTVALUE</i>	81
FIGURA 4.13 – EXEMPLO DE CÓDIGO DO MODELO <i>OZ</i> DE PERSONALIDADE.....	81
FIGURA 4.14 – DIAGRAMA DE CLASSE PARA <i>MOOD</i> E <i>PHYSICALSTATE</i>	82
FIGURA 4.15 – DIAGRAMA DE CLASSES PARA <i>ATTITUDE</i> E <i>EMOTION</i>	83
FIGURA 4.16 – DIAGRAMA DE OBJETOS EXEMPLIFICANDO UMA ENTIDADE CONCRETA	84
FIGURA 4.17 – DIAGRAMA DE OBJETOS EXEMPLIFICANDO UMA ENTIDADE ABSTRATA	84
FIGURA 4.18 – DIAGRAMA DE CLASSES PARA <i>TRAIT</i>	85
FIGURA 4.19 – DIAGRAMA DE CLASSES PARA <i>VALIDATIONMODEL</i>	85
FIGURA 5.1 – EXEMPLO DA FIGURA 2.21 CONVERTIDA PARA O FORMATO PERSONA-AIML	91
FIGURA 5.2 – EXEMPLO DE USO DA TAG <i><PATTERN-TYPE></i>	92
FIGURA 5.3 – VISÃO GERAL DE UM ARQUIVO DE DEFINIÇÃO DO COMPONENTE DE PERSONALIDADE.....	93
FIGURA 5.4 – EXEMPLO DE DEFINIÇÃO DE CRENÇAS EM XML	94
FIGURA 5.5 – EXEMPLO DE AÇÕES EM XML.....	94
FIGURA 5.6 – DEFINIÇÃO GERAL DAS REGRAS EM XML.....	95
FIGURA 5.7 – EXEMPLO DE REGRAS DE COMPORTAMENTO EM XML	96
FIGURA 5.8 – CONDIÇÕES E AÇÕES DE UMA REGRA.	96
FIGURA 5.9 – EXEMPLO APRESENTADO NA FIGURA 4.11 DESCRITO EM XML.....	97
FIGURA 5.10 – EXEMPLO DE DESCRIÇÃO DE OBJETOS DA CLASSE <i>MOOD</i> EM XML	98
FIGURA 5.11 – EXEMPLO DE DESCRIÇÃO DE OBJETOS DA CLASSE <i>PHYSICALSTATE</i> EM XML	98
FIGURA 5.12 – EXEMPLO APRESENTADO NA FIGURA 4.16 DESCRITO EM XML.....	99
FIGURA 5.13 – EXEMPLO APRESENTADO NA FIGURA 4.17 DESCRITO EM XML.....	100
FIGURA 5.14 – EXEMPLO DE EMOÇÃO DESCRITO EM XML	100
FIGURA 5.15 – EXEMPLO DA FIGURA 4.13 DESCRITO EM XML	101
FIGURA 5.16 – EXEMPLO DE USO DA TAG <i><EVALUATE></i>	102
FIGURA 5.17 – EXEMPLO DE USO DA TAG <i><EVALUATE></i>	102
FIGURA 5.18 – EXEMPLO DE USO DA TAG <i><CHANGE></i>	103
FIGURA 5.19 – EXEMPLO DE USO DA TAG <i><ATTITUDE></i>	104
FIGURA 5.20 – EXEMPLO DE USO DA TAG <i><EMOTION></i>	105
FIGURA 5.21 – EXEMPLO DE USO DA TAG <i><SWITCH></i>	106
FIGURA 5.22 – EXEMPLO DE USO DA TAG <i><CALL-ACTION></i>	106
FIGURA 5.23 – EXEMPLO DE USO DA TAG <i><MODIFY-CHANGE></i>	107
FIGURA 5.24 – EXEMPLO DE USO DA TAG <i><BLOCK-DO></i>	107
FIGURA 5.25 – EXEMPLO DE USO DA TAG <i><FIND-DATA></i>	108

FIGURA 5.26 – EXEMPLO DE USO DA TAG <i><IF></i>	109
FIGURA 5.27 – EXEMPLO DE USO DA TAG <i><EVAL></i>	109
FIGURA 5.28 – FUNCIONAMENTO DO CONVERSOR PERSONA-AIML	110
FIGURA 5.29 – INTERFACE <i>WEB</i> PARA DIÁLOGO COM O <i>CHATTERBOT</i>	113
FIGURA 5.30 – EXEMPLO DE DIÁLOGO COM O <i>CHATTERBOT</i>	114

Lista de Tabelas

TABELA 2.1 – CARACTERIZAÇÃO DOS <i>CHATTERBOTS</i> SEGUNDO PAGE.....	23
TABELA 2.2 – COMPARAÇÃO ENTRE TECNOLOGIA PARA CONSTRUÇÃO DE <i>CHATTERBOTS</i> .	38
TABELA 4.1 – DESCRIÇÃO DAS CLASSES APRESENTADAS NO DIAGRAMA UML (FIGURA 4.3)	72

1

Introdução

A idéia de prover comunicação em linguagem natural para usuários de computadores data da década de 1950, quando Turing propôs o que ficou mais tarde conhecido como o Jogo da Imitação (*Imitation Game*) (TURING, 1950). Desde então, podemos identificar diversas tentativas de se construírem tais sistemas, desde ELIZA (WEIZENBAUM, 1966), até ALICE (WALLACE, 2001), o ganhador de dois recentes prêmios Loebner¹.

Com a crescente disseminação da Web, esses sistemas voltaram ao foco de discussão entre pesquisadores das áreas de Inteligência Artificial, Filosofia e Ciência Cognitiva. Atualmente, tais sistemas são conhecidos como *chatterbots* (LAVEN, 2002), podendo ser utilizados em diversas aplicações na Internet (e.g., comércio eletrônico, ensino a distância e entretenimento).

Os *chatterbots*, além serem facilitadores no processo de interação homem-máquina, são capazes de explorar o comportamento do usuário, podendo até mesmo influenciá-lo. Estudos recentes têm mostrado a importância do uso de personalidade como forma de melhorar o desempenho de sistemas computacionais. Além disso, a personalidade também é apresentada como um componente importante no processo de tomada de decisão. Assim sendo, seria de grande utilidade a criação de *chatterbots* capazes de demonstrar personalidade coerente e robusta durante suas interações com os usuários.

Um estudo realizado durante o desenvolvimento desta dissertação revelou que já existem algumas tentativas de se incluir personalidade nesses sistemas (e.g., JULIA e PARRY). Contudo, nenhuma das implementações encontradas se mostrou satisfatória no que concerne a coerência do comportamento do *chatterbot* ao longo do tempo, e a reusabilidade do modelo de personalidade na criação de outros robôs com personalidades diferentes.

¹ <http://www.loebner.net/Prizef/loebner-prize.html>

Além disso, nosso estudo nos levou a concluir também que, dentre as tecnologias utilizadas para a construção de *chattebots*, AIML é, atualmente, a mais adequada. Contudo, apesar da importância da personalidade nesse tipo de sistema, ainda não existem implementações consistentes de personalidade em AIML.

1.1 Objetivos

Esta dissertação teve por objetivo o desenvolvimento de uma arquitetura, denominada Persona-AIML, para a criação de *chatterbots* em AIML com personalidade. Desenvolvemos um modelo de personalidade baseado no Modelo dos Cinco Grandes Fatores (*Big Five*) (MCCRAE e JOHN, 1992), entretanto, a arquitetura proposta é flexível, permitindo a utilização de outros modelos de personalidade. Neste trabalho, a personalidade é composta pelos seguintes elementos: traços, atitudes, humor, emoções e estados físicos.

Esta arquitetura é baseada nos conceitos de agentes inteligentes (RUSSEL e NORVIG, 1995), utilizando a abordagem simbólica para a definição do comportamento do *chatterbot*. A arquitetura foi modelada utilizando o paradigma orientado a objetos, e sua especificação foi feita utilizando a linguagem UML (*Unified Modeling Language*). Implementamos um protótipo a partir da especificação do Persona-AIML, a fim de testar a viabilidade da nossa proposta. A implementação utilizou a linguagem Java, a fim de preservar a modularidade, extensibilidade e reusabilidade do código.

Além da base de regras, o protótipo construído conta também com uma base de falas em AIML que foi adaptada a partir de um *chatterbot* em desenvolvimento no laboratório VIRTUS da UFPE². Testes com o protótipo revelaram resultados bastante satisfatórios no que concerne a coerência do comportamento do *chatterbot* em relação à personalidade definida para os testes.

As principais contribuições deste trabalho foram: a criação de uma arquitetura flexível, modular, extensível e reutilizável para criação de *chatterbot* com personalidade; a adaptação de um modelo de agentes inteligentes para a construção de *chatterbots*; a adaptação de um modelo consagrado na Psicologia (“*Big Five*”) para criar

² Pixel está sendo desenvolvido por uma equipe liderada pelo prof. André Neves, coordenador do VIRTUS - <http://www.virtus.ufpe.br/pixelbot>

um modelo de personalidade; e a implementação de um protótipo baseado nessa arquitetura.

1.2 Organização da Dissertação

Além deste capítulo introdutório, esta dissertação é composta pelos seguintes capítulos:

Capítulo 2 – Apresenta uma revisão sobre a área de *chatterbots*, observando suas características e aplicações. Nele são abordados os conceitos sobre agentes inteligentes e como alguns *chatterbots* consagrados pela literatura podem ser caracterizados segundo esse modelo. É feita uma análise sobre as linguagens e ferramentas existentes para o desenvolvimento de *chatterbots*, onde destacamos os conceitos básicos da linguagem AIML, que foi escolhida como base para arquitetura proposta.

Capítulo 3 – Apresenta um estudo detalhado sobre os elementos de personalidade e teorias utilizadas no desenvolvimento de modelos computacionais de personalidade. Descrevemos o papel da personalidade em agentes inteligentes e as abordagens existentes para a modelagem desse componente. É feita uma breve revisão da literatura apresentando exemplos de sistemas que utilizam um modelo de personalidade.

Capítulo 4 – Apresenta a arquitetura Persona-AIML, uma extensão de AIML, desenvolvida com base nas críticas feitas a abordagens existentes e nas necessidades específicas de *chatterbots*. São descritos os componentes da arquitetura e como modelos de personalidade podem ser criados com base em nossa arquitetura.

Capítulo 5 – Descreve a implementação da arquitetura proposta e apresenta o protótipo utilizado para a validação da arquitetura. Apresentamos os resultados obtidos através da análise dos diálogos com usuários.

Capítulo 6 – Enumera as contribuições e trabalhos futuros previstos.

Chatterbots: Características e Aplicações

Chatterbots são sistemas que utilizam linguagem natural para dialogar com o usuário (BICKMORE, 1999; LAVEN, 2002). Os primeiros *chatterbots* tinham aplicação restrita a estudos acadêmicos. Atualmente, eles são considerados alternativas capazes de desempenhar o papel de facilitadores em diversas aplicações como, por exemplo, comércio eletrônico e ensino a distância. Esses sistemas têm a capacidade de explorar o comportamento social do usuário perante o computador, mesmo quando não são programados com essa intenção.

Neste capítulo, apresentamos uma revisão sobre os *chatterbots*, observando sua evolução histórica e suas características principais.

2.1 Aplicações de *Chatterbots*

Com a popularização da Internet através da *World Wide Web*, ou *Web*, pessoas e empresas de todo o mundo passaram a utilizar cada vez mais seus serviços. Mesmo com todo o trabalho feito pelos engenheiros de software, a navegação pela *Web* ainda continua sendo uma das atividades menos prazerosas.

É preciso que o usuário utilize palavras chaves através de um engenho de busca¹ ou tenha que navegar por entre menus, muitas vezes pouco intuitivos, para localizar páginas de seu interesse. Um outro problema é a ausência de mecanismos que permitam a negociação e o diálogo, assim como ocorrem em ambientes fora da *Web*. Situações como essas são comuns a pessoas que utilizam serviços de comércio eletrônico, ensino a distância, ou que necessitam consultar informações sobre uma determinada empresa ou um produto.

Exemplos como os citados acima ilustram algumas das limitações existentes no processo de interação homem-máquina onde, ao invés de termos uma interface baseada em linguagem natural, que o ser humano está habituado a utilizar, temos que

¹ e.g, Google – <http://www.google.com>

nos adaptar e aprender a utilizar novas ferramentas para interagir com a máquina. Motivados por necessidades como estas, os *chatterbots* estão se tornando cada vez mais populares na *Web*.

Outra característica importante dos *chatterbots*, que pode contribuir para a melhora no processo de interação homem-máquina, é a constatação de que perguntas pessoais feitas pelos *chatterbots* são respondidas pelo usuário com grande frequência (LEAVERTON, 2002). Isto torna o processo de coleta de informações extremamente eficaz.

A seguir, são discutidos os principais usos de *chatterbots*: entretenimento, ensino a distância, comércio eletrônico e ensino a distância.

Entretenimento

Uma das aplicações mais comuns dos *chatterbots* é o entretenimento. Para que tal objetivo seja alcançado, é importante que o *chatterbot* apresente um comportamento dinâmico, sendo capaz de simular “ilusão de vida” através de sua personalidade. Como exemplos dessa aplicação de *chatterbots* temos SETEZOOM (INSITE, 2002) e JULIA (MAULDIN, 1994).

SETEZOOM é parte de uma campanha publicitária da empresa *Gessy Lever* cujo público alvo é composto por jovens e adolescentes, especialmente os do sexo masculino. Desta forma, SETEZOOM incorpora a personalidade de uma jovem cuja conversação é baseada em assuntos comuns ao seu público como, por exemplo, festas e namoro.

JULIA foi desenvolvida para auxiliar os jogadores de ambientes de realidade virtual baseados em texto. Suas principais características serão discutidas na seção 2.3.3. FONER (1997) faz uma discussão abrangente sobre os aspectos sociais apresentados por JULIA, destacando sua importância para sistemas de software em geral.

Ensino a Distância

Ambientes de ensino a distância têm como um dos grandes desafios manter os usuários engajados nos estudos, proporcionando meios para que eles possam

pesquisar informações de seu interesse e interagir com outros usuários a fim de aprimorar seus conhecimentos.

Chatterbots como ADELE (JOHNSON e SHAW, 1997) e STEVE (RICKEL e JOHNSON, 1997) têm sido empregados em sistemas de ensino à distância, executando tarefas como: responder questões dos estudantes, efetuar demonstrações, e monitorar as ações dos estudantes. Outro exemplo é o robô² Einstein (FORBES, 2000) desenvolvido pela empresa Artificial Life³ para ensinar física. Caso o estudante não responda uma questão dirigida a ele por um longo tempo, ele é advertido para que passe a prestar atenção na aula.

Como visto, *chatterbots* podem ser utilizados nestes ambientes como assistentes virtuais que auxiliam no processo de aprendizagem. Nesse sentido, eles devem ser projetados de forma que possam tomar atitudes perante os usuários como, por exemplo, repreender aqueles que não estejam se empenhando nos estudos. Contudo, conforme JUNG et al. (2002) ressaltam, é importante identificar que comportamentos são apropriados para estimular a aprendizagem, caso contrário, é possível gerar distúrbios no comportamento do estudante como, por exemplo, ansiedade ou timidez.

Atendimento ao Consumidor

Serviços de atendimento ao consumidor são indispensáveis para o bom funcionamento de um negócio, porém o custo associado a esse tipo de serviço é alto. Estudos mostram que a maior parte das dúvidas dos usuários podem ser respondidas utilizando uma pequena parte base de conhecimento de suporte ao usuário (GIANFORTE, 2002). Desta forma, as empresas que utilizam a Internet como um canal de comunicação com o usuário podem empregar *chatterbots* para responder suas perguntas, cortando custos e acelerando o processo. O atendimento passa a ser mais pessoal, uma vez que os usuários não precisam navegar por FAQs (*Frequently Asked Questions*) em busca de suas respostas (OH, 2000). Além disso, esses robôs também podem colher reclamações dos usuários e prestar informações sobre o andamento de suas queixas.

² Neste trabalho os termos *chatterbot* e robô serão utilizados com o mesmo significado.

³ <http://www.artificial-life.com>

Hank (COCA-COLA, 2002) é um robô desenvolvido pela empresa Coca-Cola como o objetivo de esclarecer dúvidas sobre a empresa e seus produtos. Quando não é capaz de responder uma questão, Hank sugere ao usuário tentar refazer a questão ou entrar em contato diretamente com o suporte através de um *link* fornecido por ele.

A empresa Lucas Arts desenvolveu o *chatterbot* Yoda (LUCAS, 2002) para suporte técnico aos usuários de seus jogos. Desta forma, os usuários conseguem auxílio *on-line* e dicas sobre jogos dia e noite, todos os dias da semana, o que além de ser importante para os usuários, reduz os custos da empresa para atendê-los.

Comércio Eletrônico

ANDERSON (1997) levantou diversos motivos que explicam o fato de muitas pessoas relutarem em utilizar serviços de comércio eletrônico (*e-commerce*). Entre eles destacam-se aqui a dificuldade do consumidor em localizar o produto desejado, e o fato de o consumidor não se sentir confortável ao fazer as compras, seja porque aquilo que ele compra não pode ser examinado, ou pelo medo de que o número de seu cartão de crédito seja utilizado indevidamente.

Este último fator tem sido destacado como uma grande barreira para o comércio eletrônico (DONEY e CANNON, 1997). Além disso, o cliente muitas vezes não confia que o produto comprado será realmente entregue, conforme suas expectativas. Sendo assim, é imperativo adotar políticas que estabeleçam uma relação de confiança entre cliente e vendedor. MOON (1998) demonstrou que um sistema computacional utilizando uma estratégia de reciprocidade e auto-descoberta em uma conversação baseada em linguagem natural é mais atrativo ao usuário, possibilitando que ele divulgue mais informações pessoais e se torne mais confiante para comprar um produto oferecido pelo sistema.

Nesse cenário, *chatterbots* podem contribuir para a solução do problema. Esses robôs podem facilitar e humanizar o processo de *e-commerce*, de forma que o consumidor não precise navegar entre páginas para chegar ao produto desejado, uma vez que ele poderá ser levado pelo *chatterbot* diretamente até esse produto, com base no diálogo com o cliente, analogamente ao que acontece em uma loja física. A humanização do processo ocorre a partir do momento que o *chatterbot* passa a utilizar informações do cliente sobre suas preferências, obtidas através do diálogo.

O projeto COGITO (ABBATTISTA et al., 2002) propõe o uso de *chatterbots* como assistentes virtuais em sites de *e-commerce*. Utilizando os *logs* dos diálogos, o sistema proposto extrai *profiles* a fim de descobrir quais são as preferências dos usuários. A partir dessas informações, são feitas recomendações a grupos de usuários que apresentem perfis semelhantes.

2.2 Agentes Inteligentes e *Chatterbots*

RUSSEL e NORVIG (1995) definem agente como uma entidade que percebe seu ambiente através de sensores e atua sobre ele através de efetadores. Embora existam outras definições para o termo agente⁴, cuja discussão está fora do escopo desta dissertação, o trabalho apresentado por Russel e Norvig fornece uma base sólida para a análise de *chatterbots* na perspectiva de agentes.

Um agente racional (inteligente) utiliza um componente de raciocínio para mapear cada possível seqüência de percepções na melhor ação a ser tomada, com base nos dados obtidos pela seqüência de percepções e no seu conhecimento (modelo do ambiente). Uma arquitetura genérica desse tipo de agente é apresentada na Figura 2.1.

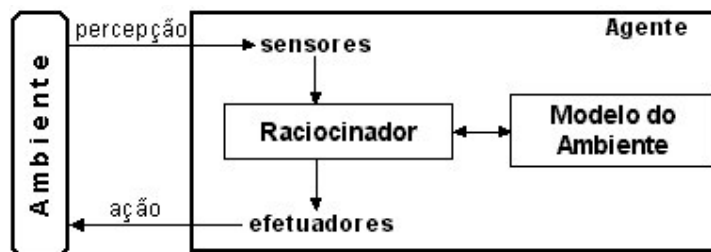


Figura 2.1 – Arquitetura geral de um agente inteligente

Segundo RUSSEL e NORVIG (1995), antes de iniciar o *design* de agentes, é preciso ter-se uma idéia bem definida a respeito do possível conjunto de percepções, ações, objetivos, e sobre o ambiente no qual o agente atuará. Este conjunto é conhecido como PAGE (do inglês, *Perceptions, Actions, Goals, Environment*).

Os *chatterbots* também podem ser caracterizados segundo esse conjunto, apesar de essa não ser a prática comum de quem trabalha na área, uma vez que as percepções e ações desse tipo de agente podem sempre ser reduzidas à interpretação e à produção de texto, respectivamente.

⁴ Ver (FRANKLIN e GRAESSER, 1997).

Apesar disso, consideramos a caracterização dos *chatterbots* segundo esse modelo vantajosa em muitos aspectos. No nosso caso, em especial, a adoção desse modelo será de grande auxílio na criação de *chatterbots* com personalidade (Capítulo 4 – seção 4.2). Para ilustrar a discussão, caracterizamos as aplicações de *chatterbots* apresentadas na seção anterior conforme esse modelo (Tabela 2.1)

Tabela 2.1 – Caracterização dos *chatterbots* segundo PAGE

Aplicação	Percepções	Ações	Objetivos	Ambiente
Entretenimento	Texto escrito	Conversar sobre um tema qualquer, guiar o usuário em um jogo, etc.	Divertir o usuário geralmente simulando “vida artificial”	Usuários
Ensino a distância	Texto escrito	Sugerir assuntos de discussão, repreender alunos, tirar dúvidas, efetuar demonstrações, etc.	Melhorar a aprendizagem do estudante	Estudantes
Atendimento ao consumidor	Texto escrito	Responder dúvidas, anotar reclamações, prestar informações sobre processos em andamento, etc.	Minimizar custos de atendimento.	Consumidores
Comércio eletrônico	Texto escrito	Sugerir a compra de produtos, guiar o comprador, etc.	Maximizar as vendas e satisfazer o cliente.	Compradores

Com base na Tabela 2.1, fica caracterizado que as percepções e as ações dos *chatterbots* consistem, essencialmente, na interpretação e na formulação de texto escrito em linguagem natural.

2.2.1 Arquiteturas de Agentes

A partir dos requisitos da aplicação a ser implementada, é possível definir qual a maneira mais eficiente de mapear as percepções dos agentes em ações, processo denominado como tomada de decisão (*decison making*). Com base nesses requisitos, RUSSEL e NORVIG (1995) propõem diversas arquiteturas de agentes: agente reativo, agente reativo com estado interno, agente baseado em objetivos, e agente otimizador. A seguir, é apresentada uma breve descrição dessas arquiteturas. A partir daí, procuraremos classificar os exemplos de *chatterbots* apresentados na seção 2.3 segundo esses modelos.

Agente Reativo

Agentes reativos mapeiam percepções em ações através de regras condição-ação (IF-THEN-ELSE). Essas regras permitem representar o conhecimento do agente de maneira modular, inteligível e eficiente. Por não registrarem percepções e ações passadas, nem possuírem um modelo do mundo, os agentes reativos não são capazes de levar em consideração nenhuma ação tomada no passado nem a consequência de alguma ação executada. Este tipo de agente é representado na Figura 2.2 e um exemplo de *chatterbot* deste tipo pode ser visto na seção 2.3.1.

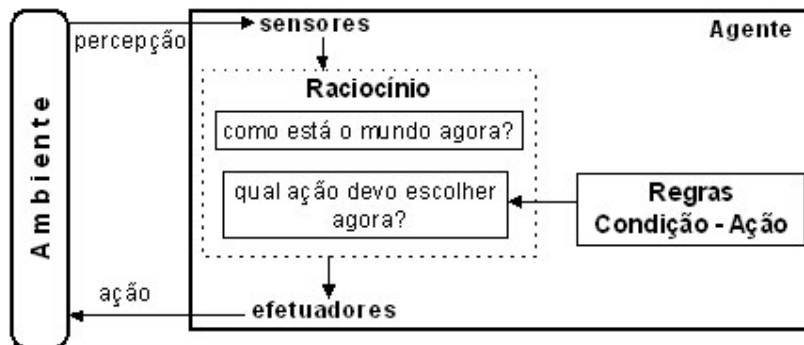


Figura 2.2 – Agente reativo (RUSSEL e NORVIG, 1995)

Agente Reativo com Estado Interno

Este tipo de agente estende o modelo anterior acrescentando-lhe um mecanismo para armazenar informações sobre o ambiente, ações e percepções, que representam seu estado interno. O processo de raciocínio do agente é composto dos seguintes passos: (1) atualização do estado interno através das percepções; (2) seleção de regras que satisfazem o estado interno atual do agente; (3) execução das ações associadas as regras selecionadas no passo anterior; e (4) atualização do estado interno de acordo com a execução das ações.

Utilizando informações de seu estado interno, os agentes apresentam maior capacidade para escolher de novas ações a serem tomadas. Agentes deste tipo podem ser caracterizados segundo a Figura 2.3 e um exemplo de *chatterbot* deste tipo pode ser visto na seção 2.3.5.

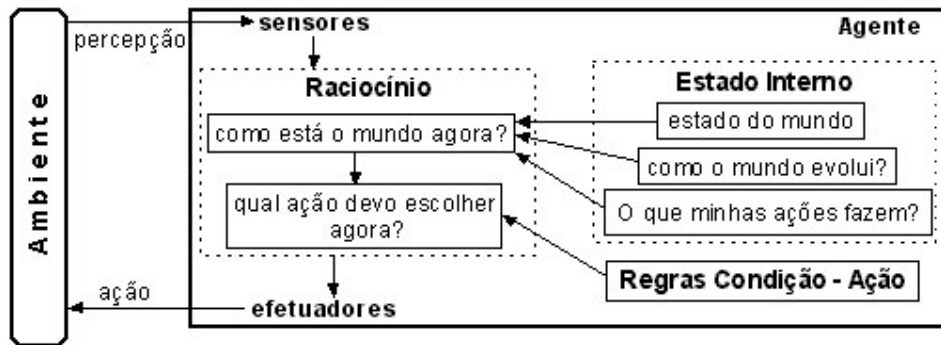


Figura 2.3 – Agente reativo com estado interno (RUSSEL e NORVIG, 1995)

Agente Baseado em Objetivos

Conhecer informações sobre o estado interno nem sempre é suficiente para decidir que ação deve ser tomada. Em certas aplicações, agentes podem possuir diversos objetivos a serem alcançados e que devem ser considerados no processo de tomada de decisão. Esses agentes podem ter mais de um objetivo consecutivo a alcançar, porém não conseguem lidar com objetivos simultâneos conflitantes.

Um exemplo clássico dessa situação é encontrado no jogo *Wumpus World* (RUSSEL e NORVIG, 1995), onde o agente deve primeiramente explorar cavernas em busca do ouro e, após encontrá-lo, buscar uma saída. Agentes dessa natureza são considerados agentes baseados em objetivo (Figura 2.4). Um exemplo de *chatterbot* deste tipo pode ser visto na seção 2.3.3.

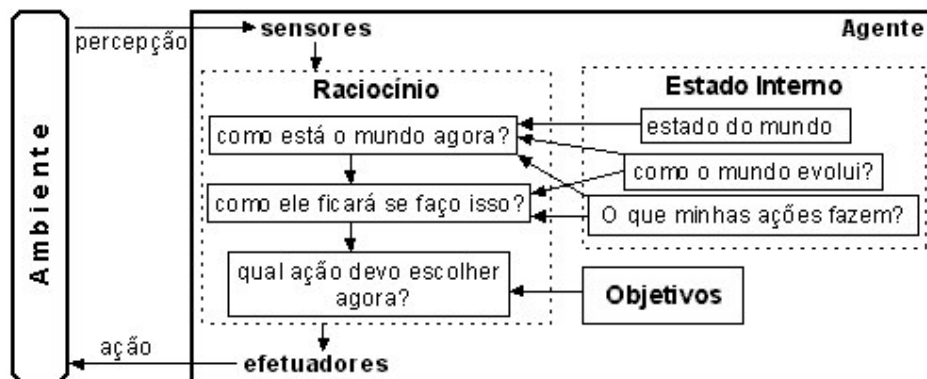


Figura 2.4 – Agente baseado em objetivos (RUSSEL e NORVIG, 1995)

Agente Otimizador

A definição explícita dos objetivos nem sempre é suficiente para que o agente possa tomar decisões de maneira eficaz. Situações onde o agente possui

objetivos conflitantes (e.g., velocidade e segurança) ou onde é necessário avaliar se suas ações modificarão o ambiente de forma satisfatória (e.g., vantagem de correr riscos) necessitam de uma função, conhecida como função de utilidade, que permita a avaliação de diferentes estados do mundo. Um agente com estas características é conhecido como otimizador (Figura 2.5). Um exemplo de uso de função de utilidade pode ser visto pelo *chatterbot* descrito na seção 2.3.3.

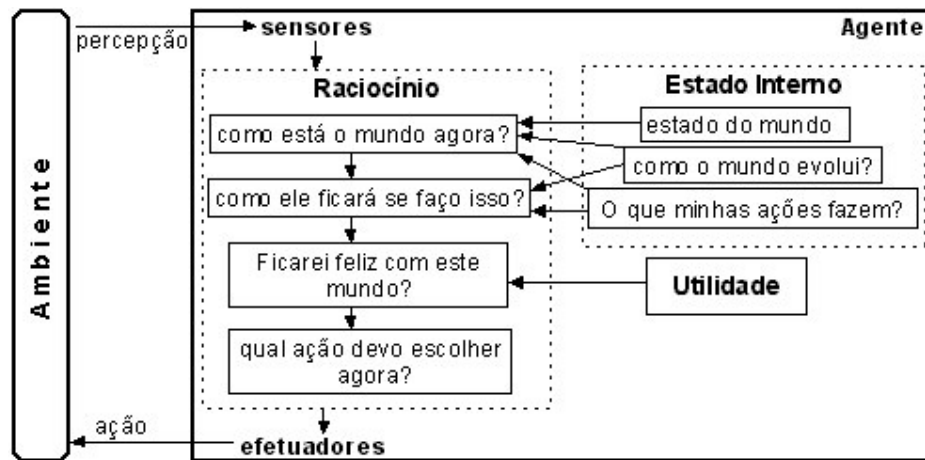


Figura 2.5 – Agente otimizador (RUSSEL e NORVIG, 1995)

2.3 Breve Histórico dos *Chatterbots*

Ao longo do desenvolvimento dos *chatterbots*, diversas técnicas têm sido exploradas objetivando melhorar a qualidade desses sistemas. Portanto, um estudo prévio sobre *chatterbots* torna-se fundamental para a comparação entre características presentes nesses sistemas, personalidade é uma delas.

Nesta seção, será apresentado um estudo considerando as características e o desempenho de alguns dos *chatterbots* existentes, a fim de levantar limitações que restringem o seu uso. Para este estudo, tomou-se como base os resultados obtidos nos concursos do *Loebner Prize*⁵. Apesar de, existirem controvérsias sobre a validade científica deste concurso⁶, ele se apresenta como um parâmetro aceitável para a avaliação, uma vez que esta é uma competição baseada apenas em texto e, é considerada uma das competições mais importantes na área de *chatterbots*.

⁵ <http://www.loebner.net/Prize/loebner-prize.html>

⁶ Ver (HARNAD, 1992; SHIEBER, 1994; LOEBNER, 1994)

Devido à sua natureza comercial, alguns *chatterbots* apresentados nesta seção não possuem artigos publicados com maiores detalhes do seu funcionamento. Por isso, algumas de nossas análises utilizam apenas as informações divulgadas em seus *websites*.

2.3.1 ELIZA

Dezesseis anos após o Teste de Turing ter sido proposto, Joseph Weizenbaum desenvolveu o *chatterbot* chamado ELIZA (WEIZENBAUM, 1966). ELIZA, também conhecida como DOCTOR, é considerada o marco inicial do desenvolvimento de *chatterbots*. Sua capacidade de conseguir que pessoas divulgassem informações de ordem pessoal, acreditando que ela era capaz de compreendê-las (WEIZENBAUM, 1976), chamou a atenção da comunidade científica. Esta característica preocupou seu criador, que percebeu que sistemas de conversação como ELIZA, relativamente simples de serem implementados, eram capazes de influenciar o comportamento do usuário.

ELIZA responde às sentenças digitadas pelo usuário de acordo com o princípio Rogeriano (ROGERS, 1951). Neste método, o cliente (paciente) assume o papel principal, sendo encorajado a falar a maior parte do tempo. O terapeuta atua resumindo, de vez em quando, o que o paciente disse e estimula-o a continuar falando, a fim de esclarecer os sentimentos do paciente, porém sem julgá-los ou explicá-los. A partir desse princípio, ELIZA, é livre para assumir que não conhece nada sobre o mundo real, não tendo que responder questões do usuário, nem falar sobre si mesma, o que simplifica o seu desenvolvimento. Assim sendo, o usuário é estimulado a tentar responder suas próprias perguntas, num processo de introspecção. A seguir, é transcrito um trecho de diálogo típico com ELIZA retirado de seu artigo original (Figura 2.6).

usuário	Men are all alike.
ELIZA	IN WHAT WAY?
usuário	They're always bugging us about something or other.
ELIZA	CAN YOU THINK OF A SPECIFIC EXAMPLE?
usuário	Well, my boyfriend made me come here.
ELIZA	YOUR BOYFRIEND MADE YOU CAME HERE.
usuário	He says I'm depressed much of the time.
ELIZA	I AM SORRY TO HEAR YOU ARE DEPRESSED.
usuário	It's true. I am unhappy.
	...
ELIZA	DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE?

Figura 2.6 – Trecho de diálogo com ELIZA

ELIZA possui os seguintes componentes: (1) uma base de regras, utilizada para processar as sentenças do usuário; e (2) um *kernel*, responsável por ler as sentenças digitadas pelo usuário e processar a base de regras em busca daquela que está associada à sentença digitada. Caso nenhuma regra seja associada à sentença do usuário, o *kernel* retorna uma resposta livre de contexto, ou, em alguns casos definidos na base de regras, pode retornar uma resposta já utilizada.

Cada regra é composta por uma lista de palavras-chaves, baseadas em regras de decomposição, que separa a sentença em termos. Associado a cada regra, existe um conjunto de regras de reconstrução da sentença, definindo assim uma ou mais respostas para o usuário. Além disso, é possível definir regras de transformação para o pré-processamento da sentença do usuário, uma vez que, em alguns casos (por exemplo, palavras sinônimas), é preciso modificar a entrada antes de processá-la.

Para ilustrar seu funcionamento, cujo princípio ainda é utilizado por *chatterbots* modernos, temos na Figura 2.7 um exemplo simplificado de uma regra utilizada por ELIZA. As palavras “decomposição” e “reconstrução” foram utilizadas apenas para melhorar o entendimento do exemplo, não fazendo parte da regra.

	(
Decomposição	(0 YOU 0 ME)
Reconstrução	(WHAT MAKES YOU THINK I 3 YOU)
)

Figura 2.7 – Exemplo de Regra de ELIZA

Na etapa de decomposição de uma regra, um número n indica que o termo será composto por uma seqüência de n palavras. O número zero (0) é uma exceção e representa um termo composto por uma seqüência infinita de palavras. Palavras-chaves

como, por exemplo, “*you*” e “*me*” determinam em qual termo da sentença elas obrigatoriamente devem aparecer. Suponha agora a seguinte sentença digitada pelo usuário: “*It seems that you hate me*”. Com base na regra acima, ela será decomposta nos seguintes termos:

(1) <i>It seems that</i>	(2) <i>you</i>	(3) <i>hate</i>	(4) <i>me</i>
--------------------------	----------------	-----------------	---------------

Podemos verificar que a regra foi satisfeita, pois o primeiro e o terceiro termos são seqüências de n palavras, e as palavras-chaves “*you*” e “*me*” são, respectivamente, o segundo e o quarto termos.

Na etapa de reconstrução da frase, um número n representa “o i -ésimo termo da decomposição”. Portanto, teremos como resposta a seguinte frase:

WHAT MAKES YOU THINK I HATE YOU
--

O uso do princípio Rogeriano permite a simplificação do diálogo mantendo um bom nível de credibilidade perante o usuário, pois encobre em parte as limitações de ELIZA. Embora mantenha um registro sobre o que foi respondido ao usuário para auxiliar no processo de reconstrução de outras frases, a versão original de ELIZA não é capaz de contextualizar as sentenças, impossibilitando, por exemplo, determinar a que se referem respostas como “sim” e “não”. Além disso, não é possível utilizar variáveis para armazenar informações fornecidas pelo usuário, inviabilizando, por exemplo, o preenchimento automático de questionários com dados pessoais do usuário. Por último, devido ao modelo de conversação adotado, ELIZA não foi desenvolvida levando em consideração a inclusão de personalidade, emoções e atitudes, dificultando o seu uso para outras aplicações.

WEIZENBAUM (1967) ainda desenvolveu uma versão melhorada de ELIZA que era capaz de contextualizar a conversação. Entretanto, a maioria das implementações encontradas atualmente se baseia na sua versão inicial.

Apesar de suas limitações, ELIZA é uma referência importantíssima na área de *chatterbots*, uma vez que várias de suas características estão presentes nos *chatterbots* atuais (por exemplo, base de regras separada do *kernel* e uso de regras para casamento de padrões baseada em termos). Desenvolvido com base em ELIZA, o PC THERAPIST (SAYGIN et al., 2000), cuja implementação é proprietária, tornou-se o

primeiro *chatterbot* a vencer o *Loebner Prize*, em 1991, fato repetido em 1992, 1993 e 1995. Outra variação de ELIZA, o HEX (DONATH, 2000; SAYGIN et al., 2000), venceu o mesmo concurso em 1996. Isto comprova a eficácia das técnicas utilizadas por ELIZA.

A versão inicial de ELIZA pode ser classificada como sendo um agente reativo. Suas regras, cujo princípio é utilizado por diversos *chatterbots* atuais, estabelecem um processamento equivalente ao mapeamento “condição-ação”. Apesar de armazenar algumas de suas ações passadas (sentenças devolvidas ao usuário), essa versão de ELIZA é incapaz de contextualizar sentenças. De fato, as informações guardadas servem apenas para serem parcialmente utilizadas em futuras respostas.

2.3.2 PARRY

PARRY (COLBY et al., 1971) é um sistema que simula o comportamento de um paciente paranóico, ou seja, um paciente que possui um conceito exagerado de si mesmo e idéias de perseguição. Assumindo um comportamento paranóico, PARRY torna plausíveis respostas consideradas estranhas, mas impede uma conversação prolongada, uma vez que ele reluta em falar sobre si mesmo.

PARRY utiliza um sistema de casamento de padrão similar ao utilizado por ELIZA. Adicionalmente, a sentença de entrada é modificada através da eliminação de algumas palavras utilizando um dicionário, ou reduzindo o tamanho da sentença original, caso ela não seja encontrada em sua base. Apesar dessas mudanças, o que melhor diferencia o comportamento de PARRY do de ELIZA é a presença dos seguintes aspectos: (1) simulação de emoções (medo, raiva e desconfiança) através de valores numéricos, utilizando-as como parte do processo de escolha das respostas; (2) avaliação do entrevistador, que pode ser considerado, por exemplo, como amigável, hostil ou dominante; (3) capacidade de falar sobre si próprio, mesmo que de forma restrita devido a sua personalidade paranóica.

Para cada sentença do usuário, o valor da intensidade das emoções é modificado de acordo com o que PARRY “pensa” a respeito daquele assunto. Por exemplo, quando o usuário fala sobre algum tópico que lhe cause raiva, o valor da emoção “raiva” é incrementado. Assuntos considerados neutros fazem com que PARRY fique mais amigável, pois diminuem o valor das emoções negativas.

Em um experimento real, psicanalistas foram convidados para avaliar o grau de paranóia de PARRY e de três pacientes humanos a partir de entrevistas feitas pelos próprios psicanalistas utilizando terminais de computador. Após serem informados de que alguns dos diálogos tinham sido gerados por uma máquina, eles não foram capazes de distinguir quais eram esses diálogos (BALLARD e JONES, 1987). Ressalta-se aqui que esses resultados não significam que PARRY conseguiu passar no Teste de Turing, uma vez que o experimento se refere a uma situação particular. A seguir, tem-se um trecho de uma entrevista com PARRY extraído de seu artigo original (Figura 2.8).

	...
entrevistador	Are you Italian?
PARRY	MY BACKGROUND IS BRITISH, BUT I WAS BORN IN THIS COUNTRY.
entrevistador	Do you know any Italian crooks?
PARRY	I TRY TO AVOID THE UNDERWORLD.
entrevistador	Are you afraid of the underworld?
PARRY	THE MAFIA ARE THE WORST GANGSTERS.
entrevistador	Are you connected with the mafia?
PARRY	YOU KNOW, THEY KNOW ME.
entrevistador	How do they know you?
PARRY	THEY SPY ON ME.
	...

Figura 2.8 – Trecho de diálogo com PARRY

2.3.3 JULIA

JULIA (MAULDIN, 1994) destacou-se por ser um agente inteligente que habita ambientes de realidade virtual TINYMUD (*Tiny Multiperson User Dimensions*). Esses ambientes são uma variação dos tipos de MUDs (CURTIS, 1992) existentes, onde jogadores atuam como personagens de uma história, interagindo através de interfaces baseadas em texto.

JULIA, além de conversar com os usuários, possui um modelo interno do mundo para auxiliar outros jogadores. Esse modelo é obtido através da exploração autônoma do ambiente TINYMUD que ela habita. Ela é capaz de lembrar informações referentes a ações tomadas anteriormente e informações sobre outros jogadores (e.g., quem a matou), para que possam ser utilizadas no processo de tomada de decisão (FONER, 1997). O fato de um ambiente MUD ser restrito a um determinado número de ações, sendo todas expressas através de texto escrito, contribuiu para facilitar o desenvolvido dessas características em JULIA.

JULIA, que assume a personalidade de uma mulher, também responde a perguntas de caráter social dos usuários. Entretanto, essa personalidade é incorporada à sua base de conhecimento, não existindo um modelo de personalidade pelo qual JULIA tivesse sido desenvolvida. Um trecho de diálogo com JULIA, que foi modificada para competir no *Loebner Prize* de 1993⁷, é reproduzido abaixo (Figura 2.9).

...	
juiz	Hello. What kind of pets do you have yourself?
JULIA	I HAVE TWO CATS.
juiz	What are their names?
JULIA	MY CATS ARE CALLED SPATS AND BOWTIE.
juiz	How do you feel about declawing?
JULIA	WHAT KINDS OF PETS DID YOU HAVE AS A CHILD?
juiz	Won't you answer my question? How do you feel about declawing?
JULIA	GUINEA PIGS ARE ABOUT AS SMART AS CUCUMBERS.
...	

Figura 2.9 – Trecho de diálogo com JULIA

A versão original de JULIA utilizava um sistema simples de regras casamento de padrões baseado em seqüências do tipo IF-THEN-ELSE. Posteriormente, JULIA passou a utilizar um modelo baseado em Redes Neurais (BRAGA et al., 1999), chamado pelo autor de Redes Conversacionais. Segundo o autor, essa mudança melhorou as respostas de JULIA, entretanto não foi suficiente para obter resultados significativos nos *Loebner Prize* de que participou (1993 e 1994). Seu código fonte é proprietário, mas uma parte dele foi utilizada no desenvolvimento de outro *chatterbot* de código aberto chamado COLIN (FONER, 1997).

Com base na análise feita por Foner, podemos considerar que JULIA atende às características dos agentes baseados em objetivo. Ela possui diversos objetivos a serem alcançados: prestar informações a jogadores; repassar mensagens de jogadores; explorar o ambiente de forma a construir seu mapa, entre outros. Sendo assim, ao ter que decidir que ações devem ser tomadas, JULIA precisa levar em consideração qual o objetivo a ser alcançado. Por exemplo, para escolher para que parte do ambiente ela vai caminhar, é preciso saber se neste momento ela está tentando explorar locais desconhecidos, ou repassar uma mensagem a um determinado jogador.

⁷ <http://www.lazytoad.com/liti/julia>

JULIA possui um mecanismo para avaliar se está sendo útil em um ambiente. Com este mecanismo, ao verificar que ninguém interage com ela por um longo período, é possível decidir explorar ambientes desconhecidos ou procurar outros usuários. Esse mecanismo é similar à função de utilidade dos agentes otimizadores.

2.3.4 JFRED e a Série ALBERT

A empresa CYBERMECHA (2002) tem desenvolvido desde 1994 uma tecnologia para a construção de *chatterbots* chamada FRED (*Functional Response Emulation Devices*) (GARNER, 2003). Dessa abordagem, destacam-se aqui a ferramenta JFRED e o *chatterbot* ALBERT One, baseado na série ALBERT, vencedor dos *Loebner Prize* de 1998 e 1999. Um trecho de diálogo com ALBERT, que competiu em 1999⁸ com o nome JIM, é reproduzido abaixo (Figura 2.10).

JIM	HELLO. MY NAME IS JIM.WHO ARE YOU?
juiz	Hello Ji ⁹ ,. I'm Sherry. How are you?
JIM	HI THERE, WHAT'S UP? I'M DOING JUST FINE, THANKS.
juiz	I'm good too; though I'm feeling a bit tired now.
JIM	I'D LIKE TO KNOW MORE ABOUT YOU. ARE YOU A GIRL OR A BOY?
juiz	I'm a girl. How about you?
JIM	DO YOU COME FROM AROUND HERE? YEAH, ME TOO.
juiz	Are you asking if I live in Adelaide?
JIM	I TRY TO BE AS PRECISE AS POSSIBLE.
	...

Figura 2.10 – Trecho de diálogo com JIM (ALBERT)

JFRED é uma ferramenta de código aberto, escrita na linguagem Java, voltada para a construção de *chatterbots* para a *Web*. A base de conhecimento para *chatterbots* baseados nessa tecnologia é descrita utilizando a linguagem JRL (*Java Ruleset Language*), baseada em regras que utilizam Lógica *Fuzzy* e palavras-chave para o casamento de padrão das sentenças do usuário.

JRL não apresenta nenhum mecanismo para modelagem de personalidade em *chatterbots* e, mesmo utilizando técnicas mais complexas para o tratamento das sentenças do usuário, o *chatterbot* BARRY DEFACTO (ROBITRON, 2003),

⁸ <http://www.cs.flinders.edu.au/Research/AI/LoebnerPrize/1999c03.txt>

⁹ Esse erro (chamar o robô de Ji ao invés de Jim) está presente no *log* de conversação.

desenvolvido inicialmente em uma versão baseada em C++, obteve apenas a quarta colocação no *Loebner Prize* de 1997¹⁰.

ALBERT é a última série da tecnologia FRED que, diferentemente de seu antecessor, não é de código aberto, impossibilitando uma análise mais detalhada. Atualmente, ALBERT se encontra na sua segunda versão.

2.3.5 ALICE e AIML

Em 1995, WALLACE (2001) criou uma linguagem baseada em XML (*eXtensible Markup Language*) (W3C, 2002) denominada AIML (*Artificial Intelligence Markup Language*) para a especificação de bases de conhecimento para *chatterbots*. Embora AIML utilize uma técnica simples de casamento de padrão, *chatterbots* baseados nessa linguagem têm conseguido resultados bem superiores aos anteriores.

ALICE, baseada em AIML, ganhou as versões do *Loebner Prize* de 2000 e 2001, sendo que nesta última, conseguiu obter de um dos juizes uma nota mais alta do que um humano, fato inédito para o evento. Sua base de conhecimento é composta por mais de vinte mil categorias de sentenças de entrada (similar às regras utilizadas por ELIZA), o que justifica em parte o fato de sua técnica relativamente simples ser capaz de obter índices de coerência na conversação bem superiores aos de seus concorrentes. Por outro lado, à medida que essa base de conhecimento aumenta, torna-se mais difícil mantê-la, especialmente se for levada em consideração a sua reutilização na construção de outros *chatterbots*. A seguir, temos um trecho de diálogo de ALICE, extraído do *Loebner Prize* de 2001¹¹ (Figura 2.11).

¹⁰ <http://www.harlington-straker.us/Studio/Albert2.html>.

¹¹ http://www.loebner.net/Prizef/2001_Contest/Richard_Wallace.txt

juiz	My name is judge02.
ALICE	I AM ALWAYS GLAD TO MAKE NEW FRIENDS, JUDGE02.
juiz	Hi, pleased to meet you?
ALICE	HI THERE JUDGE02. NICE MEETING YOU TOO JUDGE02.
juiz	may I ask you some questions?
ALICE	SURE ASK ME ANYTHING.
juiz	who composed Beethoven's piano concerto?
ALICE	CHARLIE PARKER. :-)
juiz	excellent. can you name another composer of music for me?
ALICE	THANKS FOR THE COMPLIMENT.
ALICE	LINUS TORVALDS. :-)
	...

Figura 2.11 – Trecho de diálogo com ALICE

As últimas respostas utilizadas por ALICE ilustram um de seus truques para enganar os juizes. Ao ser questionada sobre o compositor de uma música (também poderia ser o inventor da lâmpada, por exemplo), ALICE sorteia um nome de uma lista e apresenta-o com um sorriso, como se estivesse brincando com o juiz.

Diferentemente de seus antecessores, a ênfase desse trabalho é no desenvolvimento da linguagem AIML, e não na tecnologia de suporte ou em uma aplicação particular do *chatterbot*. Desta forma, existem diversas implementações do *kernel* (Java, Perl, C++, entre outras), responsável pelo processamento de bases de conhecimento em AIML, sendo todas de código aberto. Essa característica tem estimulado o desenvolvimento de uma comunidade voltada à criação de aplicações que utilizem a linguagem AIML. Apesar da ênfase no desenvolvimento da linguagem, AIML também não foi desenvolvida levando em consideração o uso de elementos para a construção de personalidade em *chatterbots*.

AIML é capaz de modelar agentes reativos com estado interno. *Chatterbots* baseados nessa linguagem são capazes de armazenar informações sobre percepções e ações passadas de forma a guiar a escolha de novas ações. Com base nessas informações, é possível contextualizar a sentença digitada pelo usuário, permitindo determinar, por exemplo, a que se referem respostas como “sim” e “não”. Desta forma, o robô pode escolher a ação a realizar de maneira mais coerente.

2.3.6 Outros *Chatterbots*

Ainda como participantes do *Loebner Prize*, destacam-se os seguintes *chatterbots*: TIPS (WHALEN, 1996), vencedor em 1994; CONVERSE (WILKS e

CATIZONE, 1999), vencedor em 1997; e MEGAHAL (HUTCHENS e ALDER, 1998), que introduziu o uso de modelos de MARKOV, obtendo terceiro lugar na competição de 1998. O que chama a atenção para estes *chatterbots* é que, mesmo utilizando técnicas mais complexas para o tratamento das sentenças, eles não obtiveram resultados mais expressivos do que os apresentados por ALICE.

Empresas também têm trabalhado no desenvolvimento de produtos baseados em *chatterbots*. Como exemplos, destacamos: ROBERTA (INTERBOTS, 2002), desenvolvida pela mesma empresa que criou CYBELLE (INTERBOTS, 2002), o primeiro *chatterbot* brasileiro; SETEZOOM, *chatterbot* de representação criado para a Gessy Lever a partir da tecnologia InBot (INSITE, 2002); e ULTRA-HAL (ZABAWARE, 2002), uma ferramenta para desenvolvimento de assistentes virtuais.

2.3.7 Agentes Conversacionais Incorporados

Agentes conversacionais incorporados (*Embodied Conversational Agents*) estendem os *chatterbots* conversacionais. Eles utilizam, em tempo-real, fala, gestos, observação (entre outros canais verbais e não verbais), para simular a interação humana face-a-face (BICKMORE, 1999). É importante ressaltar que esses sistemas também são considerados *chatterbots*, porém apresentam um maior foco no desenvolvimento de tecnologias que permitam o uso desses outros canais de comunicação e não apenas o diálogo escrito.

REA (CASSELL et al., 1999) é um dos exemplos mais sofisticados dessa nova linha de *chatterbots*, sendo capaz de monitorar as ações do usuário através de câmeras digitais enquanto conversam. Por outro lado, REA está longe de ser aplicada em larga escala, devido à grande necessidade de recursos de áudio e vídeo (FORBES, 2000).

KSHIRSAGAR e MAGNENAT-THALMANN (2002) utilizam a linguagem AIML para o processamento de texto e geração de respostas, que é parte da arquitetura utilizada para o desenvolvimento de humanos virtuais. Estes humanos são capazes de expressar gestos e emoções. Portanto, a evolução das técnicas de construção dos *chatterbots* baseados em texto pode contribuir na construção de sistemas mais complexos. A Figura 2.12 apresenta imagens do “humano virtual” (KSHIRSAGAR e MAGNENAT-THALMANN, 2002).



Figura 2.12 – Imagens do Humano Virtual

À medida que estes sistemas exploram novos canais de comunicação, aumenta a complexidade para o seu desenvolvimento e sua utilização, especialmente em ambientes como a *Web*, onde recursos como imagem e voz em tempo real são extremamente caros. Esse conflito entre a complexidade utilizada e os recursos disponíveis para melhorar a comunicação precisa ser levando em consideração, pois, mesmo utilizando agentes com poucos recursos (e.g, apenas texto), é possível obter-se uma grande variedade de respostas sociais do usuário (NASS et al., 1994).

2.4 Estudo Comparativo

Nesta seção, apresentamos um estudo comparativo entre as tecnologias utilizadas para a construção dos *chatterbots* analisados neste capítulo. Embora alguns dos *chatterbots* tenham sido criados para propósitos específicos, nada impediu que a tecnologia envolvida tivesse sido utilizada como base para outros *chatterbots*. Sendo assim, *chatterbots* como ELIZA e JULIA serão incluídos na comparação apresentada na Tabela 2.2, que se baseou nos seguintes critérios:

1. **Memória:** Capacidade do *chatterbot* de armazenar informações do usuário e sobre o diálogo;
2. **Contexto:** Capacidade do *chatterbot* de contextualizar uma frase;
3. **Personalidade:** Existência de um modelo de personalidade na qual é baseada a construção da personalidade dos *chatterbots*;
4. **Portabilidade:** Possibilidade de ser implementada utilizando diferentes linguagens de programação e independência da plataforma de execução; e

5. Propriedade: Indica se o sistema é proprietário ou pode ser livremente utilizado.

Crítérios preenchidos com traços (-) não foram avaliados devido à indisponibilidade de informações precisas a seu respeito.

Tabela 2.2 – Comparação entre tecnologia para construção de *chatterbots*

Nome	Memória	Contexto	Personalidade	Portabilidade	Propriedade
ELIZA	não	sim	não	não	aberta
PARRY	não	-	não	não	aberta
JULIA	sim	-	não	não	fechada
JFRED	sim	sim	não	não	aberta
ALBERT	sim	-	não	não	fechada
AIML	sim	sim	não	sim	aberta
TIPS	-	-	não	não	fechada
CONVERSE	sim	-	não	não	fechada
MEGAHAL	sim	sim	não	não	aberta
INBOT	sim	sim	-	não	fechada
ULTRA-HAL	sim	sim	-	não	fechada
INTERBOT	sim	sim	-	não	fechada

Com base nos critérios utilizados nesta comparação, verifica-se que nenhuma das tecnologias analisadas apresenta, quando foi possível verificar, um modelo de personalidade projetado para a construção de *chatterbots*. Isto representa um grande obstáculo para a construção de personalidades robustas e coerentes, além de dificultar a reutilização das bases de conhecimento dos *chatterbots*. Desta forma, por não haver um padrão para o projeto da personalidade, cada *botmaster*¹² utiliza um modelo *ad hoc* que leva em consideração apenas a sua aplicação.

Observa-se também que AIML representa, atualmente, a melhor ferramenta para desenvolvimento de *chatterbots*, uma vez que é um padrão aberto e independente de linguagem de programação, além de apresentar memória e capacidade de contextualização das sentenças. Além disso, os excelentes resultados apresentados nos *Loebner Prize* têm demonstrado a sua robustez para o desenvolvimento de *chatterbots*.

A próxima seção apresenta um estudo sobre a linguagem AIML, focado seus conceitos básicos.

¹² Como é comumente conhecido o projetista do *chatterbot*.

2.5 A Linguagem AIML

Esta seção descreve a linguagem AIML, que é, atualmente, a melhor ferramenta para construção de *chatterbots* para Web. Esta linguagem foi utilizada no desenvolvimento do nosso trabalho (ver Capítulo 4), daí o nosso especial interesse em apresentar aqui suas características principais.

A linguagem AIML, baseada em XML, é descrita através de identificadores denominados *tags*. Ela foi desenvolvida tendo como inspiração o modelo de casamento de padrões proposto por ELIZA. Sendo assim, para explicar seu funcionamento faremos uma analogia com o modelo de ELIZA (discutido em maiores detalhes na seção 2.3.1).

Em AIML, cada regra (categoria) é representada por uma *tag* `<category>`. Essencialmente, cada categoria é constituída por um *componente de decomposição* da sentença digitada pelo usuário e por um *componente de reconstrução* (ou construção) da sentença devolvida pelo *chatterbot* ao usuário, sendo denominados respectivamente como *pattern* e *template*. A Figura 2.13 apresenta a sintaxe de uma categoria em AIML.

```
<category>
  <pattern>PADRÃO DE ENTRADA</pattern>
  <template>PADRÃO DE SAÍDA</template>
</category>
```

Figura 2.13 – Sintaxe básica de uma categoria AIML

Utilizando um sistema para casamento de padrões simplificado, AIML possui apenas os caracteres especiais “*” e “_” para referenciar os termos da decomposição. O significado desses caracteres é equivalente ao do símbolo “0” de ELIZA, isto é, representam um termo composto por uma seqüência infinita de palavras. A diferença entre esses caracteres especiais é que AIML, ao fazer o casamento de padrão, dá prioridade a regras que utilizem o caractere especial “_”.

Para a recomposição da sentença, os termos capturados pelos caracteres especiais podem ser referenciados através da *tag* `<star index="i"/>`, onde *i* representa o *i*-ésimo termo capturado por um caractere especial. A Figura 2.14 reescreve a regra de ELIZA apresentada na Figura 2.7 utilizando a linguagem AIML.

```

<category>
  <pattern>* YOU * ME</pattern>
  <template>WHAT MAKES YOU THINK I <star index="2"/> YOU</template>
</category>

```

Figura 2.14 – Regra de ELIZA reescrita utilizando AIML

A partir dos exemplos apresentados, podemos verificar que a estrutura da linguagem AIML é similar a de ELIZA. Entretanto, AIML introduz um conjunto de *tags* que permitem sua diferenciação de ELIZA com relação a características como, por exemplo, possuir memória e ser capaz de contextualizar as sentenças digitada pelo usuário.

A seguir são apresentadas as principais *tags* de AIML, fundamentais para a compreensão da linguagem, agrupadas de acordo com a sua funcionalidade. RINGATE et al. (2001) apresentam uma descrição detalhada de todas as *tags* que compõem a linguagem AIML. Em seguida, serão discutidas as limitações dessa linguagem.

Componentes de Memória

Para armazenar e recuperar variáveis da memória do *chatterbot*, AIML define as *tags* `<set>` e `<get>`, respectivamente. A Figura 2.15 ilustra um exemplo utilizando essas *tags*.

Sintaxe	
<pre> <set name="variavel">valor</set> <get name="variavel"/> </pre>	
Exemplos de Categorias	
<pre> <category> <pattern>MEU NOME É *</pattern> <template>MUITO PRAZER <set name="nome"><star/></set>.</template> </category> <category> <pattern>VOCE SE LEMBRA DO MEU NOME</pattern> <template>LEMBRO SIM <get name="nome"/>.</template> </category> </pre>	
Exemplo de Diálogo	
usuário	Meu nome e João.
robô	MUITO PRAZER JOÃO.
usuário	Voce se lembra do meu nome?
robô	LEMBRO SIM JOÃO.

Figura 2.15 – Exemplo de uso das *tags* `<get>` e `<set>`

A tag `<think>` modifica as *tags* que ela contém determinando que, após o seu processamento, o resultado não deve ser retornado. Esse mecanismo, como a tradução da *tag* sugere, simula a idéia de que o *chatterbot* está pensando. A Figura 2.16 mostra esse princípio.

Sintaxe	
<code><think>conteúdo a ser processado</think></code>	
Exemplo de Categoria	
<pre> <category> <pattern>MEU SIGNO E *</pattern> <template>OBRIGADO PELA INFORMACAO. <think><set name="nome"><star/></set></think> </template> </category> </pre>	
Exemplo de Diálogo	
usuário	Meu signo é touro.
robô	OBRIGADO PELA INFORMACAO.

Figura 2.16 – Exemplo de uso da tag `<think>`

AIML especifica que as sentenças digitadas pelo usuário devem ser armazenadas para que possam ser utilizadas futuramente, mantendo assim um registro do diálogo com o usuário. Para fazer referência a esse diálogo armazenado, AIML define a tag `<input>`, que recupera as sentenças na ordem inversa de que foram digitadas. Esta tag é utilizada no *template* de uma categoria. Sua sintaxe e um exemplo de seu funcionamento são apresentados na Figura 2.17.

Sintaxe	
<code><input index="numero" /></code>	
Exemplo de Diálogo	
usuário	Meu nome e João.
robô	MUITO PRAZER JOÃO.
usuário	Você se lembra do meu nome?
robô	LEMBRO SIM JOÃO.
Significado	
<code><input index="1" /></code>	VOCE SE LEMBRA DO MEU NOME?
<code><input index="2" /></code>	MEU NOME E JOÃO.

Figura 2.17 – Exemplo de funcionamento da tag `<input>`

Contextualização do Diálogo

Conforme mencionado anteriormente, a linguagem AIML é capaz de contextualizar o diálogo com o usuário. Para isso ela introduz duas *tags*: `<that>` e `<topic>`.

A tag `<that>` possui a mesma sintaxe que a tag `<pattern>`, podendo assim utilizar os mesmos caracteres especiais. Estando presente em uma categoria, a tag `<that>` define que, além de ser necessário fazer o casamento de padrão definido no `pattern`, deve-se verificar se a última frase dita pelo robô “casa” com a tag `<that>`. A Figura 2.18 apresenta a sintaxe e um exemplo de uso ilustrando como o robô pode contextualizar respostas do tipo “sim” ou “não”.

Sintaxe	
<code><that>PADRÃO DE ENTRADA</that></code>	
Exemplos de Categorias	
<pre> <category> <pattern>FAÇA UMA PERGUNTA</pattern> <template>VOCE GOSTA DE FILMES?</template> </category> <category> <pattern>SIM</pattern> <that>VOCE GOSTA DE FILMES?</that> <template>QUE BOM, EU TAMBEM GOSTO DE FILMES.</template> </category> <category> <pattern>NAO</pattern> <that>VOCE GOSTA DE FILMES?</that> <template>QUE PENA, EU GOSTO DE FILMES.</template> </category> </pre>	
Exemplos de Diálogos	
usuário1	Faça uma pergunta.
robô	VOCE GOSTA DE FILMES?
usuário1	Sim.
robô	QUE BOM, EU TAMBEM GOSTO DE FILMES.
usuário2	Faça uma pergunta.
robô	VOCE GOSTA DE FILMES?
usuário2	Não.
Robô	QUE PENA, EU GOSTO DE FILMES.

Figura 2.18 – Exemplo utilizando a tag `<that>`

Durante um diálogo é comum que as pessoas falem sobre diversos assuntos (tópicos). Modelar a base de conhecimento a partir de tópicos permite que o robô possa melhorar a escolha da resposta para o usuário. O tópico vigente é determinado através da tag `<set>` utilizando a variável reservada “topic”.

Para exemplificar o uso de tópicos, suponha uma situação onde o robô não foi programado para responder uma determinada pergunta, mas sabe qual o assunto que está sendo abordado. Desta forma, o robô é, no mínimo, capaz de devolver uma resposta

que pertença ao contexto do tópico, melhorando assim a qualidade do diálogo. A Figura 2.19 ilustra essa situação.

Sintaxe	
<pre><topic name="nome"> <!--categorias--> </topic></pre>	
Exemplos de Categorias	
<pre><category> <pattern>VAMOS FALAR SOBRE CACHORROS</pattern> <template>OK. VAMOS FALAR SOBRE <set name="topic">CACHORROS</set> </template> </category> <topic name="cachorros"> <category> <pattern>*</pattern> <template>CACHORROS SÃO ANIMAIS INTERESSANTES</template> </category> <category> <pattern>EU GOSTO MUITO DELES</pattern> <template>TAMBÉM GOSTO DE CACHORROS.</template> </category> </topic></pre>	
Diálogo	
usuário	Vamos falar sobre cachorros.
robô	OK. VAMOS FALAR SOBRE CACHORROS.
usuário	Eu gosto muito deles.
robô	TAMBEM GOSTO DE CACHORROS.
usuário	Que legal!
robô	CACHORROS SÃO ANIMAIS INTERESSANTES.

Figura 2.19 – Exemplo utilizando a tag <topic>

Condições e Reavaliação da Sentença

Para finalizar a discussão sobre as principais *tags* de AIML, apresentamos as *tags* <condition> e <srail>, importantes no processo de construção das respostas do *chatterbot*.

A *tag* <condition> permite que o robô faça uma avaliação de maneira similar às instruções “switch” e “if” das linguagens de programação Java e C¹³. Seu uso é ilustrado na Figura 2.20.

¹³ Outras linguagens de programação possuem comandos equivalentes, mas ilustramos utilizando JAVA e C por serem amplamente conhecidas no meio acadêmico e industrial.

Sintaxe	
<pre><condition name="variável"> <li value="valor"><<instruções>> ... <!-- instruções default --> </condition> <condition name="variável" value="valor"> <!-- instruções --> </condition></pre>	
Exemplo de Categoria	
<pre><category> <pattern>COMO VAI VOCE?</pattern> <template> <condition name="alegria"> <li value="alegre">Eu estou feliz <li value="triste">Eu estou triste </condition> </template> </category></pre>	
Exemplo de Diálogo	
usuário1	Como vai voce?
robô	EU ESTOU FELIZ.
usuário2	Como vai voce?
robô	EU ESTOU TRISTE.

Figura 2.20 – Exemplo utilizando a tag `<condition>`

A tag `<srail>` indica que o sistema de seleção de regras (*kernel*) deve avaliar o seu conteúdo como se fosse uma sentença digitada pelo usuário. Desta forma, além de permitir que diferentes padrões de entrada sejam direcionados a uma mesma reposta (processo conhecido como redução simbólica), implementa-se um mecanismo equivalente à chamada de procedimento, comum em linguagens de programação. A Figura 2.21 ilustra um exemplo de uso desta tag.

Sintaxe	
<code><srain>conteúdo a ser processado</srain></code>	
Exemplos de Categorias	
<pre> <category> <pattern>ATE MAIS</pattern> <template><srain>TCHAU</srain></template> </category> </category> <pattern>TCHAU</pattern> <template>NOS VEMOS OUTRO DIA, SÃO <srain>HORAS</srain></template> </category> </category> <pattern>HORAS</pattern> <template><!-- cálculo da hora --></template> </category> </pre>	
Exemplo de Diálogo	
usuário1	Ate mais.
robô	NOS VEMOS OUTRO DIA, SÃO 02:00HS
usuário2	Tchau.
robô	NOS VEMOS OUTRO DIA, SÃO 23:00HS

Figura 2.21 – Exemplo utilizando a tag `<srain>`

Avaliação da AIML

O processo de redução simbólica da linguagem é ineficiente e confuso, uma vez que o *botmaster* precisa definir diversas categorias com um único objetivo. Isso faz com que a base de categorias seja demasiadamente grande, dificultando a sua manutenção.

A técnica de identificação da sentença do usuário é bastante simples. Com isso as bases de categorias precisam ser muito grandes para que o robô obtenha um bom resultado. Técnicas de recuperação de informação poderiam ser combinadas ao casamento de padrão a fim de melhorar a qualidade dos *chatterbots* baseados nessa linguagem.

2.6 Conclusões

Chatterbots despertam o interesse da comunidade científica e industrial, pois, além de proverem uma interface mais amigável e natural, permitem explorar as relações sociais que os usuários desenvolvem com máquinas. Embora eles existam desde os anos 1960, o interesse nesses sistemas voltou a crescer a partir da popularização da Internet, onde os *websites* passaram a ter um problema real, que é a

necessidade de uma interface mais amigável para prover informações a milhões de usuários.

Este estudo pôde identificar os seguintes problemas referentes à tecnologia de *chatterbots*: (1) a aquisição e o gerenciamento de sua base de conhecimento; (2) a técnica para identificação das sentenças digitadas pelo usuário; (3) o uso de personalidade.

A aquisição e o gerenciamento da base de conhecimento do *chatterbot* são de importância central, uma vez que seu tamanho e sua qualidade têm-se mostrado fundamentais para a obtenção de bons níveis de coerência na conversação. Isso pode ser comprovado ao se verificar que *chatterbots* com técnicas simples para a identificação das sentenças dos usuários, mas que possuem bases de conhecimento robustas, são capazes de obter resultados comparáveis àqueles que utilizam técnicas mais complexas. Um problema com relação a bases de conhecimentos grandes como a de ALICE (composta por mais de vinte mil categorias), é a dificuldade em gerenciá-las, principalmente levando em conta a sua reutilização em outros *chatterbots*.

O fato de *chatterbots* que utilizam técnicas mais complexas para a identificação das sentenças do usuário não apresentarem necessariamente melhores resultados não significa que as técnicas para identificação das sentenças do usuário não são importantes. De fato, são elas que possibilitam diminuir o tamanho das bases de conhecimento sem a perda da qualidade do diálogo.

Por sua vez, o uso de personalidade tem sido explorado em *chatterbots* sem a preocupação de se desenvolver um modelo que permita a construção de personalidades coerentes e independentes da base de conhecimento. Desta forma, os *chatterbots*, além de não explorarem de maneira eficiente as possibilidades de interação homem-máquina, apresentam bases de conhecimento difíceis de serem reutilizadas, devido à complexidade em separar o conhecimento e a personalidade.

Para exemplificar este último problema, que é o foco desta dissertação, suponha o seguinte exemplo: como criar um *chatterbot* de entretenimento que deixe de responder ao usuário caso fique zangado? Imagine agora, que queiramos reaproveitar a base de conhecimento de ALICE para esse objetivo. Seguindo a abordagem atualmente utilizada pelos *chatterbots*, poderíamos embutir regras em cada categoria, verificando se

o robô está zangado antes de responder ao usuário. Entretanto, além desse trabalho manual ser extremamente penoso, tendo em vista o tamanho da base de conhecimento, não estaríamos reutilizando a base de conhecimento, e sim fazendo uma cópia alterada da base. Ainda assim, teríamos que refazer todo o processo, caso quiséssemos modificar a personalidade do *chatterbot*.

O próximo capítulo apresenta uma revisão sobre personalidade em agentes inteligentes, detalhando algumas teorias da Psicologia utilizadas na criação de modelos de personalidade para esses agentes virtuais.

3

Personalidade em Agentes Inteligentes

Conforme vimos no capítulo anterior, é possível encontrar na literatura um amplo estudo sobre arquiteturas, características, aplicações e limitações do uso de agentes inteligentes. Recentemente, o uso de personalidade tem ganhado força nesses sistemas como forma de melhorar o seu desempenho. Contudo, não foi possível encontrar um estudo semelhante sobre personalidade focalizado em *chatbots*, nosso tema de trabalho.

Neste capítulo apresentamos o papel da personalidade em agentes inteligentes em geral e, complementando a discussão apresentada no Capítulo 2, apresentamos sua perspectiva em *chatbots*. Em seguida, veremos alguns conceitos extraídos da Psicologia sobre personalidade utilizados no desenvolvimento de modelos de personalidade para agentes. A partir de exemplos da literatura, apresentamos como esses conceitos têm sido empregados na construção de agentes em geral e qual a sua perspectiva com relação aos *chatbots*.

3.1 O Papel da Personalidade em Agentes Inteligentes

Existem diversos trabalhos apresentando o uso de personalidade em agentes inteligentes (ELLIOT, 1992; REILLY, 1996; SILVA, 2000; BAZZAN et al., 2002). Nesses trabalhos, a personalidade é utilizada em dois casos principais: no processo de tomada de decisão, e na criação de “ilusão de vida” em agentes sintéticos.

No processo de tomada de decisão, a personalidade se faz presente através de seus elementos, sendo a emoção o mais comum deles. Vários argumentos sugerem que a emoção pode tornar este processo mais flexível (BAZZAN et al., 2002), sendo apresentada como um mecanismo capaz de fornecer avaliações rápidas e automáticas das condições do ambiente e do estado interno do agente (OLIVEIRA e SARMENTO, 2002).

A criação de ilusão de vida tem sido amplamente empregada na construção de atores sintéticos, também conhecidos como agentes credíveis (*believable agents*), ou agentes virtuais (*virtual agents*) (SILVA, 2000). Esses agentes são geralmente utilizados em ambientes educacionais ou de entretenimento (ROUSSEAU e HAYES-ROTH, 1997). Nesses trabalhos, a personalidade é considerada o aspecto mais importante para adicionar ao agente credibilidade e comportamentos sociais perante o público.

Segundo pesquisas de NASS et al. (1994), a interação que ocorre entre humanos e computadores é fundamentalmente social, o que sugere que princípios extraídos da Psicologia Social, da Comunicação e da Sociologia são relevantes ao estudo das interações homem-máquina. Conforme BICKMORE e CASSELL (2001) ressaltam, estratégias para a modelagem da interação entre homem-máquina podem ser exploradas na construção de um relacionamento de confiança, permitindo assim que o usuário sinta-se mais à vontade para divulgar informações pessoais.

A personalidade, incluindo atitudes e emoções, é considerada o elemento básico para a criação de “ilusão de vida” perante o usuário (REILLY e BATES, 1995; ROUSSEAU e HAYES-ROTH, 1997), processo esse que torna os agentes credíveis. À medida que os agentes se tornam mais capazes de simular essa ilusão, as ligações sociais formadas entre agentes e humanos tendem a se tornar mais fortes (ELLIOT e BRZEZINSKI, 1998). Através deste mecanismo, o fortalecimento dessas ligações pode ser utilizado para influenciar as atitudes e o comportamento do usuário.

Na seção 2.1 apresentamos as principais aplicações de *chatterbots* e, com base nos estudos apresentados por NASS et al., podemos concluir que é possível explorar o comportamento do usuário através da personalidade do *chatterbot*, contribuindo assim no processo de criação de “ilusão de vida” e credibilidade do agente. Nessa direção, observamos que *chatterbots* apresentam características comuns aos agentes credíveis. Desta forma, as aplicações apresentadas no Capítulo 2 apresentam um grande potencial para a exploração do uso de personalidade perante o usuário. De fato, os projetistas objetivam essa exploração, contudo, com base em nossos estudos, não verificamos a preocupação em se trabalhar a criação de *chatterbots* com base em modelos de personalidade.

Chatterbots também podem utilizar emoções e atitudes como componente de funções de utilidade, a fim de melhor atingir seus objetivos. Por exemplo, um

chatbot vendedor cujo estado emocional seja “triste” ao verificar a desistência na compra de produtos, poderia utilizar uma estratégia mais agressiva (e.g., oferecer um desconto) para tentar finalizar a venda, assim como muitas vezes ocorre em situações reais de compras. Em outro exemplo, um robô guia, parte de um jogo de aventura, poderia dar instruções corretas ao usuário apenas se “gostasse” dele, passando a dar informações erradas ou ignorá-lo caso contrário.

Assim sendo, é importante que *chatbots* explorem o comportamento social do usuário. Entretanto, o uso desta abordagem introduz novas questões legais e éticas que devem ser consideradas no processo de interação homem-máquina (HECKMAN e WOBROCK, 2000).

3.2 Personalidade Segundo a Psicologia

Apesar de sua importância, não existe um consenso sobre a definição de personalidade e seus elementos (ROUSSEAU, 1996). ALLPORT (1937) apresenta cinquenta definições para esse termo, extraídas de diversas áreas, inclusive da Psicologia. Os termos “personalidade” e “comportamento”, por serem frequentemente utilizados pelos psicólogos de maneira alternada, geram grande confusão entre os leigos. Enquanto a personalidade é uma abstração ou construção hipotética do (ou sobre o) comportamento, o comportamento consiste em fatos observados (MISCHEL, 1968).

Para KRECH e CRUTCHFIELD (1973), todo conhecimento psicológico, em última análise, contribui para a compreensão da personalidade, considerada o ápice da Psicologia. Segundo os autores, a definição de personalidade para a Psicologia tende a ser ampla, abrangendo a totalidade da constituição psicológica do indivíduo. Essa definição inclui: traços, crenças, atitudes, reações emocionais típicas, estados de humor, entre outras características do indivíduo. Além disso, ela define como todos esses aspectos¹ (elementos) se organizam na pessoa.

Veremos nesta seção a definição dos elementos de personalidade e as teorias e os modelos que são comumente utilizadas no desenvolvimento de modelos de personalidade para agentes inteligentes.

¹ Utilizaremos o termo *elemento* para que não haja confusão com o termo *aspecto* utilizado na área de Ciência da Computação.

3.2.1 Elementos

O modelo de personalidade adotado por um agente inteligente não reflete necessariamente um modelo teórico extraído da Psicologia. Na verdade, a Psicologia tem sido utilizada como um guia para a interpretação do papel de cada elemento de personalidade no comportamento do agente, além de identificar, a partir do comportamento humano, como cada elemento interfere nos demais.

Esse fato não representa uma limitação por si, uma vez que, em sua maioria, os modelos de personalidade para agentes não têm como objetivo representar toda a complexidade humana (ainda não desvendada em sua totalidade), e sim construir modelos que permitam melhorar a eficiência dos agentes. Além disso, em alguns casos (e.g., criação de personagens de desenhos animados), é preferível a utilização de um modelo não realista para agentes (REILLY, 1996).

Estados físicos, emoções, humor, sentimentos, atitudes e traços de personalidade são os elementos mais comumente encontrados em modelos computacionais, apesar de aparecerem obrigatoriamente em todos eles. Assim sendo, focalizamos nesses elementos a nossa discussão sobre personalidade em agentes inteligentes.

Emoções, Humor e Sentimentos

Emoções são consideradas fundamentais no processo de tomada de decisão. Muitos psicólogos argumentam que é impossível para uma pessoa ter um pensamento ou encadear uma ação sem ativar, mesmo que inconscientemente, seu sistema emocional (PICARD, 1997).

Palavras com contexto emocional (e.g., feliz, alegre e gostar) são utilizadas para fazer referência a uma grande variedade de estados ou eventos psicológicos (FRIJDA, 1994). Dentre esses eventos, destacamos aqui aqueles que comumente são utilizados de maneira ambígua (BRAVE e NASS, 2002): emoção (*emotion*); humor (*mood*); e sentimento (*sentiment*), também conhecido como atitude emocional. Uma das causas dessa ambigüidade é a existência de palavras comuns para esses termos como, por exemplo, “feliz” (humor ou emoção) e “gostar” (sentimento ou emoção).

Estes termos diferem entre si em vários aspectos (FRIJDA, 1994), destacando-se aqui dois fatores importantes: a origem e a duração. Emoções e humor são utilizados para caracterizar estados de um indivíduo. Por outro lado, sentimentos são assinalados como propriedades de uma entidade (pessoas, animais e objetos). Enquanto emoções e humor tendem a ser de curta duração (emoções tendo duração de segundos e humor durando horas ou alguns dias), sentimentos podem persistir indefinitivamente (Figura 3.1).



Figura 3.1 – Duração de emoções, humor e sentimentos

Além de ter uma duração mais curta, a emoção se diferencia do humor por ser dirigida a alguma entidade. Pessoas “gostam” de alguém e sentem “medo” de algo. Por outro lado, o humor reflete um estado não intencional da pessoa, caracterizando um estado global ou difuso. Portanto, utilizando o exemplo do início da seção, podemos dizer que uma pessoa está feliz por receber um presente (emoção), ou que ela é geralmente feliz ou alegre (humor). Outro exemplo: uma pessoa pode se sentir “triste” ao brigar com um amigo (emoção), ou pode nos parecer triste (humor).

O humor tende a guiar o modo como as emoções são experimentadas, modificando os limites de ativação de emoções relacionadas. Desta forma, uma pessoa de mal-humor acaba experimentando eventos de uma maneira mais negativa. As emoções, por outro lado, tendem a contribuir na causa ou modificação de humor (BRAVE e NASS, 2002). Por exemplo, alguém que recebeu más notícias, tende a ficar de mau humor ou triste.

Sentimentos estão associados a emoções (positivas ou negativas) que esperamos experimentar ao interagir com uma determinada entidade (BRAVE e NASS, 2002). Esse julgamento baseia-se em: (1) diferentes experiências que serão generalizadas; (2) aprendizagem social; ou (3) uma base inata (e.g., não gostar de ver sangue) (FRIJDA, 1994). Por exemplo, uma criança pode dizer que “gosta” de brincar

com cachorros, fruto de experiências passadas com esta espécie de animal, ou porque aprendeu através de desenhos ou amigos que o cachorro é o melhor amigo do homem. O termo “gostar” também pode ser utilizado para expressar um estado emocional, por exemplo: eu “gostei” de brincar com Totó.

Sentimentos podem fazer com que indivíduos evitem um objeto particular ou um evento específico. De fato, em uma interação, os sentimentos são a base para as emoções que surgirão com relação a uma entidade. Por exemplo, uma pessoa que gosta de cães tende a experimentar emoções prazerosas ao interagir com cães.

Estados Físicos

Estados físicos estão relacionados a necessidades ou capacidades físicas de um indivíduo (e.g., sono, fome, sede e força). Assim como as emoções, os estados físicos são importantes no processo de tomada de decisão, uma vez que estão ligados à sobrevivência do indivíduo, guiando a escolha de objetivos (SILVA, 2000). Por exemplo, uma pessoa que está trabalhando e sentindo sede para o trabalho para beber água, retornando ao trabalho a seguir.

Atitudes

Pessoas também são caracterizadas pela sua capacidade de construir e manter relações sociais. Essas relações são representadas pelas atitudes, moldadas a partir das preferências de um indivíduo e de sua convivência com outras entidades. As atitudes tendem a condicionar como, e se, certas ações serão realizadas. Por exemplo, alguém que ajuda outra pessoa apenas caso “goste” dela.

As atitudes podem desencadear várias emoções, mas não representam por si estruturas emocionais. Elas mudam ao longo do tempo, porém essa mudança tende a ser lenta, ao contrário das emoções (REILLY, 1996).

É importante não confundir o conceito de atitude com sentimento. Conforme foi exposto, os sentimentos apresentam-se como um julgamento prévio de que emoções serão vivenciadas com um objeto, porém refletem um comportamento generalizado. As atitudes, por outro lado, se referem a aspectos sociais relacionados a uma única entidade.

Traços de Personalidade

Quando somos solicitados a descrever a personalidade de alguém, podemos apresentá-la como sendo, por exemplo, pessimista, tímida, cooperativa e inteligente. Utilizando esses termos, tentamos representar generalizações sobre uma pessoa a partir de observações do seu comportamento em diferentes situações. Essas observações são conhecidas como traços de personalidade.

Um traço é uma característica duradoura do indivíduo que se manifesta na maneira consistente de comportar-se em uma ampla variedade de situações (KRECH e CRUTCHFIELD, 1973). Um traço atribuído a um indivíduo significa que ele, geralmente, tende a apresentar um determinado padrão de comportamento. Entretanto, isto não implica que o comportamento das pessoas é definido a partir da presença ou ausência de um determinado traço, pois o traço existe em uma determinada quantidade, variando de pessoa para pessoa. Por exemplo, ao considerarmos uma pessoa “otimista”, provavelmente, não estamos querendo dizer que ela sempre é otimista, mas que seu comportamento é, na maioria dos casos, mais otimista. Portanto, um traço de personalidade é uma dimensão, na qual as pessoas variam quanto à quantidade que exibem do traço.

3.2.2 Teorias e Modelos

Apresentamos aqui de maneira sucinta, os modelos e as teorias da Psicologia utilizadas no desenvolvimento de modelos computacionais de personalidade. Conforme apresentamos no início da seção, um modelo computacional não reflete fielmente um modelo ou teoria extraídos da Psicologia. Geralmente, os modelos computacionais são adaptados com em princípios extraídos da Psicologia.

Serão apresentados a seguir duas teorias e dois modelos da Psicologia que tem servido como base para o desenvolvimento de modelos computacionais para a personalidade : (1) Teoria dos Traços (ALLPORT, 1966); (2) Modelo do Cinco Grandes Fatores (MCCRAE e JOHN, 1992; SAUCIER e GOLDBERG, 1998); (3) Teoria da Aprendizagem Social (BANDURA, 1977); e (4) Modelo OCC (ORTONY et al., 1988).

Teoria dos Traços

A Teoria dos Traços (ALLPORT, 1966) pressupõe a existência de traços que caracterizam o comportamento de uma pessoa em muitas situações, o que poderá levar o psicólogo, ou qualquer indivíduo que conheça bem essa pessoa, a ser capaz de prever o que provavelmente ela fará, mesmo em situações jamais observadas.

Através de uma pesquisa com base na língua inglesa reunindo adjetivos utilizados para descrever o comportamento de indivíduos, ALLPORT e ODBERT (1936) identificaram cerca de 4.500 termos que designavam claramente o comportamento de um indivíduo, sendo assim traços potenciais que deveriam ser analisados.

A principal idéia por trás da Teoria dos Traços que tem sido utilizada em modelos computacionais é especificar a personalidade do agente utilizando um conjunto finito de traços, geralmente estáveis, que influenciarão os demais elementos de personalidade (e.g., atitudes, emoções e estados físicos). Por exemplo, um agente extrovertido (traço extroversão/introversão) se apresentaria mais alegre. Cabe ao projetista do agente identificar que traços seriam suficientes para representar o seu comportamento.

O Modelo dos Cinco Grandes Fatores (*Big Five*)

O modelo dos Cinco Grandes Fatores é uma versão mais atual da Teoria dos Traços. Diferentes trabalhos, partindo de abordagens distintas, reduziram os traços catalogados por ALLPORT e ODBERT (1936) a cinco fatores. Estudos têm mostrado que esses fatores são considerados universais, tendo também sido reconhecidos em outras línguas como Alemão e Chinês (MCCRAE e COSTA, 1997).

Embora haja consenso com relação à solução de cinco fatores, podemos encontrar na literatura dois modelos principais: *Big Five* (SAUCIER e GOLDBERG, 1998) e *Five Factor* (MCCRAE e JOHN, 1992). A principal diferença entre esses modelos está relacionada à sua base teórica. O *Big Five* utiliza a hipótese léxica, supondo que as diferenças individuais socialmente relevantes já estão codificadas através de termos da linguagem natural. O *Five Factor Model*, baseado em

questionários, supõe que os traços da personalidade estão ordenados em um modelo hierárquico e compreensivo onde a genética e fatores ambientais são relevantes.

De acordo com o modelo utilizado (*Five Factor* ou *Big Five*), os autores nomeiam os fatores da seguinte maneira: (I) *Extraversion/Surgency*; (II) *Agreeableness*; (III) *Conscientiousness*; (IV) *Neuroticism/Emotion Stability*; (V) *Openness/Intellect*. Como visto, os fatores I, IV e V diferem na nomenclatura adotada por cada autor. HUTZ et al. (1998) traduzem esses fatores, respectivamente, da seguinte maneira: Extroversão/Introversão; Nível de Socialização; Escrupulosidade ou Vontade; Neuroticismo/Estabilidade Emocional; e Abertura a Experiência ou Intelecto.

De maneira resumida, podemos interpretar os fatores da seguinte forma:

- I. Como o próprio nome sugere, indica a tendência de uma pessoa ser introvertida ou extrovertida;
- II. Indica a tendência de um indivíduo a ser socialmente agradável, caloroso e dócil;
- III. Este fator agrupa características de personalidade que levam a responsabilidade e honestidade;
- IV. Características envolvendo afeto positivo e negativo, ansiedade e estabilidade emocional se agrupam nesse fator.
- V. Engloba características como flexibilidade de pensamento, fantasia e imaginação, abertura para novas experiências e interesses culturais.

A literatura frequentemente não faz referência às diferenças existentes entre os modelos *Five Factor* e *Big Five*. De uma maneira geral, utiliza-se o termo *Big Five* para fazer referência à solução de cinco fatores, sem se preocupar com o modelo correspondente.

Para HOWARD e HOWARD (1995), a importância do *Big Five* advém do fato de que a comunidade de pesquisa sobre personalidade é fragmentada em teorias como, por exemplo, as propostas por Freud e Jung. Segundo os autores, todas estão parcialmente corretas, mas apenas o *Big Five* é capaz de abranger todas. Na verdade, o *Big Five* não é uma teoria em si, nem seus defensores argumentam que seja (DIGMAN, 1990; SAUCIER e GOLDBERG, 1995). Ele representa uma taxonomia importante para

a descrição da personalidade, que pode ser usada por diferentes teorias da Psicologia para integrar os resultados obtidos.

Apesar de ser muito utilizada, o *Big Five* não é uma taxonomia perfeita para descrever completamente as diferenças individuais das pessoas, pois representa uma descrição em alto nível da personalidade (MCCRAE e JOHN, 1992). Ele não captura completamente dimensões que envolvem características físicas, maturidade e gênero da pessoa (JOHN, 1990; PAUNONEN e JACKSON, 2000). Apesar dessas limitações, é possível descrever tipos de indivíduos (e.g., líder e mediador) através de uma combinação dos fatores (traços) que compõem o *Big Five* (HOWARD e HOWARD, 1995).

A Teoria da Aprendizagem Social

A Teoria da Aprendizagem Social (BANDURA, 1977) explica o comportamento humano em termos de uma interação recíproca e contínua entre determinantes cognitivos, comportamentais e ambientais. De acordo com Bandura, grande parte dos comportamentos socialmente manifestados resultam de um processo de observação, identificação, imitação e modelagem. Isso significa que os sujeitos atuam de acordo com o comportamento que observam em outros indivíduos com os quais se identificam (e.g., pais e amigos).

A aprendizagem social é, para Bandura, orientada por quatro processos: (1) atenção - um comportamento só pode ser imitado se os seus aspectos mais significativos forem apreendidos; (2) retenção - a reprodução de um comportamento depende da sua codificação simbólica; (3) produção - o sujeito tem que possuir as capacidades e/ou habilidades para reproduzir um comportamento; e (4) motivação - o indivíduo pode aprender um comportamento, mas para que isso aconteça é preciso que ele perceba que o comportamento lhe permite alcançar o resultado desejado e será benéfico para ele.

Na aprendizagem social, o reforço funciona como um operante informativo e motivacional. Ele facilita a aprendizagem ao encorajar o observador a prestar atenção e a ensaiar o comportamento observado. Para Bandura, o reforço pode assumir duas formas fundamentais: (1) reforço direto - o sujeito é recompensado ou punido pelo comportamento manifestado; (2) reforço indireto - o sujeito observa alguém

experimentar conseqüências positivas ou punitivas por um comportamento e antecipa que o mesmo lhe poderá acontecer.

É a partir da idéia de reforço que sistemas computacionais tentam explorar a Teoria da Aprendizagem Social. Sendo assim, o agente seria programado para, até um certo ponto de seu desenvolvimento, modificar sua personalidade de acordo com as situações vivenciadas, consideradas experiências de aprendizagem por observação e reforço.

O Modelo OCC

Uma vez que as emoções são consideradas fundamentais no processo de tomada de decisão, é possível encontrar alguns na literatura diversos trabalhos objetivando modelar as emoções². ORTONY et al. (1988) desenvolveram um modelo para modelagem de emoções, conhecido como OCC³, que é tido, atualmente, como um padrão no desenvolvimento de agentes (BARTNECK, 2002). Uma característica importante desse modelo é o comprometimento que os autores têm com a área de Inteligência Artificial (BAZZAN et al., 2002).

O modelo OCC especifica um conjunto de emoções que são agrupadas em categorias de acordo com as condições que as geram: conseqüência de eventos; ações de agentes; e aspectos de objetos. Esse modelo apresenta um bom nível de complexidade e detalhe, cobrindo uma vasta quantidade de situações emocionais com as quais normalmente um agente pode se deparar.

Embora o modelo desenvolvido tenha como objetivo ser implementado computacionalmente, vários detalhes de implementação foram omitidos (BAZZAN et al., 2002; BARTNECK, 2002). Além disso, por ser simplificado, ele não deixa claro qual é a influência de outros elementos de personalidade como, por exemplo, atitudes e humor. O modelo descreve, em alto-nível, como variáveis globais e limiares de ativação são utilizados como meio de potencializar as emoções.

² Veja, por exemplo, ROSEMAN et al. (1996) e SLOMAN (1999).

³ OCC é um acrônimo para o nome dos autores Ortony, Clore e Collins.

3.3 Modelos Computacionais de Personalidade em Agentes

Com base na definição de KRECH e CRUTCHFIELD (1973), vista anteriormente, podemos definir um modelo de personalidade a partir de dois componentes básicos: (1) um conjunto de elementos (e.g., humor e estados físicos) que irão compor a personalidade; (2) um mecanismo que determina como os elementos de personalidade interagem entre si e como eles interferem no processo de tomada de decisão. Este último será denominado ao longo deste trabalho como sendo o “mecanismo de validação” do modelo.

Embora a definição proposta pelos autores não seja consensual, os modelos de personalidade apresentados podem ser vistos a partir desta ótica. Dessa forma, permite-se o seu uso para a análise de modelos de personalidade. A partir dessa definição, veremos que a grande diferença entre os modelos de personalidade estudados reside na escolha do mecanismo de validação, cuja escolha depende fundamentalmente do tipo de aplicação do agente.

A Psicologia fornece uma base teórica para a compreensão de cada componente da personalidade, incluindo sua relevância no processo de tomada de decisão e na manifestação de relações sociais. Porém, conforme visto neste capítulo, um modelo computacional não reflete necessariamente uma teoria completa da Psicologia sobre personalidade. Assim sendo, baseado principalmente no tipo de aplicação, podemos encontrar diversos modelos de personalidade para agentes, sendo alguns deles compostos apenas por emoções e outros incluindo vários elementos citados na seção 3.2.1.

Nesta seção veremos as abordagens para a modelagem de personalidade e alguns exemplos de seu uso em agentes inteligentes.

3.3.1 Abordagens Utilizadas

Para STERN (1999), a modelagem da personalidade em agentes é feita a partir de abordagens distintas⁴: *top-down* ou simbólica (REILLY, 1996; SILVA, 2000; e BAZZAN et al., 2002) e *bottom-up* ou conexionista (KSHIRSAGAR e MAGNENAT-THALMANN, 2002; e OLIVEIRA e SARMENTO, 2002). Na abordagem *top-down*,

⁴ Em seu artigo, Stern contextualiza as abordagens *top-down* e *bottom-up* na criação de personagens virtuais. Contudo, com base nas características de cada abordagem, esta classificação pode ser estendida para agentes em geral.

cada comportamento do agente é explicitamente definido através de regras. No outro extremo, a abordagem *bottom-up* utiliza mecanismos (e.g., modelos matemáticos, redes bayesianas e redes neurais) para modelar tal comportamento.

Para o uso da abordagem *top-down*, argumenta-se como vantagem a definição mais precisa e inteligível do comportamento do agente. Os defensores da abordagem *bottom-up*, por outro lado, argumentam que à medida que o número de regras cresce, mais difícil se torna a manutenção do comportamento do agente. Portanto, a abordagem *bottom-up* seria mais escalonável. Contudo, modelar comportamentos complexos através de parâmetros numéricos pode ser uma dificuldade para os projetistas⁵ dos agentes.

A aplicação do agente é um aspecto fundamental na escolha do tipo de abordagem a ser utilizado. Por exemplo, a abordagem *top-down* é, atualmente, a mais utilizada na construção de atores sintéticos (SILVA, 2000), devido à maior precisão na definição do comportamento do agente. KSHIRSAGAR e MAGNENAT-THALMANN (2002), por outro lado, utilizam uma abordagem *bottom-up*, pois argumentam que a incerteza é um aspecto importante do comportamento humano.

Um outro aspecto que deve ser levado em consideração para a escolha entre as abordagens *top-down* e *bottom-up* é a preferência do projetista na criação dos agentes (BATES, 1994; STERN, 1999), uma vez que cabe a ele, entre outras atribuições, a criação da “ilusão de vida” que, conforme explicado, é muito importante para aplicações que requerem interação homem-máquina.

Para ambas as abordagens, o uso de traços de personalidade é benéfico. Os traços podem tornar modelos *bottom-up* mais inteligíveis, uma vez que o projetista utiliza uma especificação do comportamento do agente em auto-nível. Modelos *top-down* são beneficiados a partir do momento que é possível obter-se uma maior flexibilidade para o comportamento do agente.

3.3.2 Exemplos de Agentes Inteligentes com Personalidade

O projeto *Oz* (REILLY, 1996) foi desenvolvido com o intuito de modelar atores sintéticos para um ambiente interativo. Baseado neste projeto, o sistema *The*

⁵ Consideramos o projetista como sendo aquele cuja responsabilidade é a modelagem, em alto-nível, do comportamento do agente.

Woogles é composto por criaturas (chamadas *woogles*) que possuem personalidade própria e são capazes de expressar emoções. Esse projeto considera a emoção o elemento principal da personalidade, e utiliza o modelo OCC (ORTONY et al., 1988).

No projeto *Virtual Theater* (ROUSSEAU e HAYES-ROTH, 1997), agentes são desenvolvidos para se comportarem como atores inteligentes que assumem papéis de personagens fictícios. A partir de sua personalidade, os atores improvisam seu comportamento sem a necessidade de um planejamento detalhado, de maneira similar ao que ocorre com atores reais, em um ambiente cujas histórias são interativas. Este modelo de personalidade é baseado em duas teorias: a teoria dos traços (ALLPORT, 1966) e a teoria da aprendizagem social (BANDURA, 1977).

SILVA (2000) faz uma análise detalhada dos projetos *OZ* e *Virtual Theater* (entre outros projetos), cujos modelos de personalidade são discutidos tendo em vista a sua viabilidade para a construção de atores sintéticos aplicados a jogos com longos períodos de interação. Nesse trabalho, é apresentado o jogo de aventura *Enigmas no Campus* (SILVA, 2000), onde o jogador deve solucionar enigmas a partir de pistas que são encontradas pela exploração do ambiente. O jogo possui diversos atores sintéticos com personalidade e objetivos próprios. Para a modelagem da personalidade, o trabalho utiliza um novo modelo de personalidade baseado no modelo *Big Five* (HOWARD e HOWARD, 1995), pois, segundo a autora, neste tipo de aplicação, os agentes necessitam ter personalidade consistente ao longo do tempo.

BAZZAN et al. (2002) apresentam uma ferramenta para a criação de agentes em cenários onde a aplicação de regras sociais é importante. Segundo os autores, as regras sociais formam a base de uma abordagem para a coordenação de agentes, ponto fundamental em sistemas multi-agentes. Em sua versão inicial, esta ferramenta focaliza apenas o uso de emoções utilizando o modelo OCC.

OLIVEIRA e SARMENTO (2002) utilizam um modelo matemático, denominado *Função de Valência Emocional*, para a simulação de emoções e humor em agentes inteligentes. Nesses agentes, a personalidade pode ser modificada através da variação de parâmetros do modelo matemático apresentado. Os autores sugerem que algoritmos genéticos aplicados na escolha desses parâmetros podem ser úteis para a criação de um agente otimizado.

KSHIRSAGAR e MAGNENAT-THALMANN (2002) utilizam Redes Bayesianas de Crença (*Bayesian Belief Networks*) na simulação de um humano virtual. Esse sistema apresenta uma arquitetura em camadas para a modelagem do humor, de emoções e de traços de personalidade do agente com base nos modelos *Big Five* e OCC. As autoras consideram uma tarefa desafiadora a escolha das matrizes de probabilidade utilizadas pelo agente, o que reflete a dificuldade de empregar abordagens *bottom-up*.

Além desses, outros projetos têm sido desenvolvidos com o intuito de incorporar personalidade a agentes (ELLIOT, 1992; FRANK et al., 1997; ANDRÉ et al., 1999). Nesses projetos, a personalidade tem sido utilizada para explorar o comportamento do usuário.

3.3.3 Exemplos de *Chatterbots* com Personalidade

Conforme visto no capítulo anterior, a personalidade dos *chatterbots* tem sido projetada de maneira *ad hoc* e específica para cada robô. Dentre os *chatterbots* apresentados aqui, apenas PARRY (COLBY et al., 1971) e JULIA possuem personalidade tendo em vista o seu uso no processo de tomada de decisão, ainda que de maneira restrita. PARRY utiliza algumas emoções projetadas especificamente para ele. JULIA é capaz de apresentar atitudes com relação aos usuários. Entretanto, sua personalidade encontra-se embutida no código do agente, o que dificulta a sua reutilização. Diante deste cenário, limitamo-nos à comentar alguns aspectos do comportamento (manifestação da personalidade) dos *chatterbots*.

Utilizar várias respostas para uma mesma pergunta é uma estratégia comum em *chatterbots*, pois permite uma maior variedade no diálogo, o que ajuda na criação de “ilusão de vida”. Contudo, esta estratégia perde a eficácia à medida que as respostas do *chatterbot* estão fixas em sua base de conhecimento. Elas não levam em consideração a avaliação do estado interno do *chatterbot*, nem as mudanças de seu comportamento. Desta forma, os *chatterbots* não são capazes de apresentar um comportamento dinâmico e coerente, que seria bem mais eficaz para a criação de ilusão de vida do que apenas o uso de várias respostas. O exemplo da Figura 3.2, obtida utilizando a versão *on-line* de ALICE⁶, mostra este tipo de abordagem.

⁶ <http://www.alicebot.org>

usuário	How are you?
ALICE	I AM DOING VERY WELL. HOW ARE YOU?
usuário	How are you?
ALICE	EVERYTHING IS GOING EXTREMELY WELL.
usuário	How are you?
ALICE	MY LOGIC AND COGNITIVE FUNCTIONS ARE NORMAL.

Figura 3.2 – Exemplo de respostas variadas utilizadas pelos *chatterbots*

Os *chatterbots*, de uma maneira geral, levam em consideração para a seleção de respostas apenas a sua correspondência com relação às sentenças digitadas pelo usuário. Por exemplo, ao ser agredido pelo usuário com um palavrão, o *chatterbot* utiliza respostas como “Seja mais educado” e “Não fale assim”. Embora essas respostas sejam coerentes com a sentença do usuário, esse “truque” é percebido pelo usuário porque, diferente de uma pessoa, as atitudes do usuário não afetam o comportamento do *chatterbot*, pois suas respostas estão previamente definidas. Desta forma é possível que o *chatterbot* expresse, por exemplo, tristeza (utilizando uma imagem), porém, ao ser perguntado se está feliz, o robô utilizará uma resposta pré-determinada em sua base de conhecido (provavelmente “sim”), o que representa uma inconsistência.

3.4 Conclusões

Neste capítulo apresentamos a importância da personalidade na construção de agentes. Além de ser fundamental para a criação de “ilusão de vida”, a personalidade pode ser utilizada no processo de tomada de decisão, incorporando a esses agentes um mecanismo dinâmico e de fácil compreensão, pois é baseado no comportamento humano.

A colaboração entre pesquisas envolvendo as áreas de Psicologia e Inteligência Artificial é vista como sendo de grande importância na criação de ferramentas que auxiliem os projetistas a incorporar personalidade em agentes. Desta forma, sistemas inteligentes têm buscado na Psicologia bases teóricas que facilitem o seu entendimento e, conseqüentemente, a sua implementação.

Embora a colaboração entre estas áreas seja importante, a Psicologia não fornece, atualmente, um modelo consensual e computacional para a modelagem da personalidade. Assim sendo, podemos encontrar diversos modelos de personalidade para agentes. A escolha do modelo depende fundamentalmente da aplicação do agente.

De uma maneira geral, a personalidade de *chatbots* deve ser capaz de simular “ilusão de vida” e influir no processo de tomada de decisão. Portanto, a modelagem de emoções, humor, estados físicos e atitudes são consideradas fundamentais para este tipo de aplicação.

Embora *chatbots* naturalmente necessitem de um modelo de personalidade, os sistemas estudados têm utilizado abordagens *ad hoc* para a sua implementação. Essas abordagens não levam em consideração os resultados obtidos em anos de trabalho da comunidade de agentes. Desta forma, é possível compreender o porquê da personalidade apresentada por esses agentes ser extremamente simplificada, apresentando comportamentos fugazes.

O próximo capítulo apresenta a arquitetura Persona-AIML, desenvolvida neste trabalho para a criação de modelos de personalidade em *chatbots*.

4

A Arquitetura Persona-AIML

Como visto no Capítulo 2, *chatterbots*, além serem facilitadores no processo de interação homem-máquina, são capazes de explorar o comportamento do usuário, podendo até mesmo influenciá-lo. Além disso, concluímos que, dentre as tecnologias disponíveis para a construção de *chattebots*, AIML é, atualmente, a mais adequada. Vimos também que, apesar de a personalidade ser importante nesse tipo de sistema (Capítulo 3), não existem implementações consistentes de personalidade em AIML.

Esses fatores motivaram o desenvolvimento de uma arquitetura, denominada Persona-AIML, uma extensão de AIML, para a criação de *chatterbots* com personalidade. Esta arquitetura é baseada nos conceitos de agentes inteligentes (seção 2.2), utilizando a abordagem simbólica (ou *top-down*) para a definição do comportamento do *chatterbot* (seção 3.3.1). A personalidade do agente é definida através de uma arquitetura genérica, podendo ser utilizada para o desenvolvimento de modelos de personalidade baseados em teorias distintas (seção 3.2.2). A arquitetura foi modelada utilizando o paradigma orientado a objetos, e sua especificação é feita utilizando a linguagem UML (*Unified Modeling Language*) (FOWLER e SCOTT, 2000).

A seção 4.1 apresenta os princípios básicos que nortearam o desenvolvimento da arquitetura Persona-AIML. A seção 4.2 apresenta a arquitetura proposta, detalhando seus componentes. Na seção 4.3, vemos como é realizada a criação de modelos de personalidade para agentes segundo nossa arquitetura.

4.1 Princípios Básicos

No Capítulo 2 (seção 2.2), apresentamos a arquitetura geral de um agente inteligente baseado na definição de RUSSEL e NORVIG (1995). Vimos também que os *chatterbots* AIML, apesar de serem agentes, utilizam uma arquitetura onde a seqüência de raciocínio (perceber-raciocinar-agir) encontra-se integrada, sendo totalmente definida através de categorias. O interpretador AIML recebe a sentença do usuário (dados

perceptivos) e identifica a categoria correspondente a ela (raciocínio). Uma vez selecionada a categoria, o interpretador retorna o seu *template* para o usuário (ação).

Esse modelo, onde a identificação da categoria cujo padrão “casa” com a sentença do usuário é a única etapa de raciocínio, dificulta a modelagem da personalidade, pois para definir o comportamento do *chatbot*, o *botmaster* deve analisar todas as categorias presentes na base AIML. Isso, além de dificultar a criação de um modelo de personalidade robusta, prejudica a reutilização da base de categorias, uma vez que as respostas (*template*) estão comprometidas com a personalidade específica de um *chatbot*.

Devido a esses problemas, optamos por uma abordagem baseada em agentes seguindo o modelo proposto por Russel e Norvig. Essa abordagem, além de permitir a separação da seqüência de raciocínio, apresenta uma grande literatura voltada para a incorporação de personalidade.

É preciso levar em consideração o seguinte fato: *chatbots* são projetados para sempre responderem ao usuário, mesmo que essa resposta não seja a mais adequada. Se, por um lado, uma resposta incoerente¹ pode prejudicar o andamento do diálogo, por outro, a ausência de resposta causaria no usuário a impressão de que o robô não recebeu a sua sentença, ou que este não lhe está dando atenção². Logo, é imperativo fornecer um meio para que o *chatbot* sempre responda ao usuário. Desta forma, a arquitetura deve possuir uma base de categorias capaz de identificar um grande número de padrão de entrada e, conforme exposto, sempre apresentar um padrão de saída.

Como forma de possibilitar o reuso da base de categorias AIML, em nossa arquitetura o padrão de saída deve representar um comportamento padrão para vários *chatbots* ou parte de uma resposta. Por exemplo, uma base de categorias contento saudações (e.g., “oi” e “olá”) pode definir como padrão de saída a mesma saudação (“oi” e “olá”, respectivamente). Como é comum que a maioria dos *chatbots* utilizem essa resposta, o *botmaster* não tem que se preocupar em modelar esse comportamento. Entretanto, para criar um *chatbot* “ranzinza”, o *botmaster* pode condicionar essas saudações ao fato do *chatbot* gostar do usuário e, caso contrário, ele utilizaria uma outra resposta (e.g., “não quero falar com você, saia daqui”). Neste último caso, o

¹ Uma resposta incoerente é aquela que foge ao contexto do diálogo ou não condiz com a personalidade do *chatbot*.

² Consideramos aqui que essa hipótese não foi programada pelo *botmaster*.

comportamento do “*chatterbot* ranzinza”, isto é, as especificações das condições para a saudação (parte da resposta) e a resposta alternativa devem ser definidas em um componente separado, a fim de preservar reutilização da base de categorias. Este componente foi denominado aqui de componente de personalidade (seção 4.2.1).

Assim, os comportamentos “padrão” permanecem na base de categorias AIML, e apenas os comportamentos influenciados pela personalidade do agente ficam no componente de personalidade. Dessa maneira, o *botmaster* poderá reutilizar a base de categorias na construção de *chatterbots* com diferentes personalidades, tendo que alterar o componente de personalidade.

No Capítulo 3 vimos que diversos modelos de personalidade tem sido desenvolvidos. Analisando esses modelos, observamos que cada um é adequado a uma determinada aplicação (3.3.2), porém, nenhum deles é “ideal” para todas as aplicações possíveis. Desta maneira, nossa principal preocupação no desenvolvimento da Persona-AIML não foi à utilização de um modelo de personalidade particular, e sim o desenvolvimento de uma arquitetura flexível que permita a construção de modelos de personalidade. Para realizar este objetivo, seguindo a definição de personalidade apresentada na 3.2, separamos o modelo de personalidade em dois componentes: conjunto de elementos de personalidade e mecanismo de validação. Com base nestes componentes, o *botmaster* pode definir um modelo de personalidade de acordo com a sua necessidade.

Para a modelagem da arquitetura, escolhemos o uso do paradigma orientado a objetos. Esse paradigma fornece recursos que facilitam o desenvolvimento de uma arquitetura modular, extensível e baseada em componentes. Uma outra vantagem do paradigma orientado a objetos é que ele permite a modelagem de agentes de uma maneira mais natural (FIGUEIRA, 2000).

A especificação da arquitetura foi feita utilizando a linguagem UML que é, atualmente, o padrão mais bem aceito para a especificação, visualização, construção e documentação de software.

4.2 Arquitetura

Nosso agente pode ser classificado como sendo do tipo reativo com estado interno. Esse tipo de agente, embora mais simples do que, por exemplo, um agente

baseado em objetivos, é adequado para a incorporação de personalidade em *chatterbots*, foco principal do nosso trabalho, uma vez que o *chatterbot* aqui não tem objetivo explícito.

Com base nos princípios apresentados anteriormente, a arquitetura Persona-AIML possui os seguintes componentes: (1) base de categorias; (2) *log* do diálogo (3) componente de personalidade; e (4) componente de raciocínio. A Figura 4.1 apresenta uma visão geral dessa arquitetura.

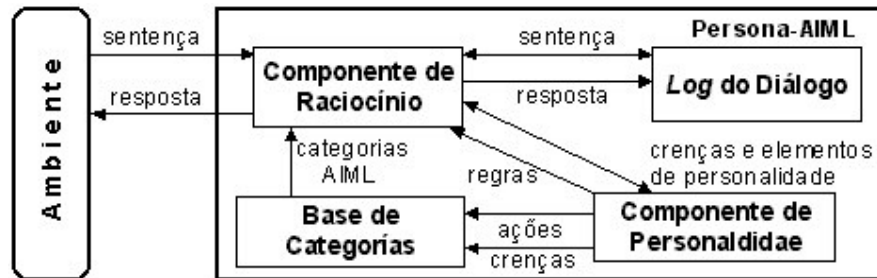


Figura 4.1 – Arquitetura Persona-AIML

A base de categorias é composta por um ou mais arquivos. Nesses arquivos, são definidas categorias AIML (seção 2.5), utilizadas pelo componente de raciocínio para identificar o padrão da sentença digitada pelo usuário. Conforme veremos mais adiante, esta base de categorias foi modificada, acrescentando-se novas informações para o componente de raciocínio (seção 4.2.2).

O *log* do diálogo é responsável por manter o histórico da conversação com o usuário, fornecendo recursos para a sua contextualização. Esse componente já é fornecido pelos interpretadores da linguagem AIML.

No componente de personalidade, são definidas as crenças, o modelo de personalidade adotado (definidos através dos elementos de personalidade e mecanismo de validação), as ações do *chatterbot* e as regras que determinam o seu comportamento (seção 4.2.1).

O componente de raciocínio (seção 4.2.2) é responsável pela identificação da sentença do usuário (casamento de padrão) e pela construção da resposta do usuário. Neste último processo será feita a execução das regras definidas no componente de personalidade, que ajudarão a definir a resposta do *chatterbot* ao usuário.

A correspondência entre os componentes de nossa arquitetura e o modelo conceitual de Russel e Norvig (apresentado na seção 2.2) é feita da seguinte forma: (1) O estado interno do agente é composto pelo *log* do diálogo, pelas crenças e elementos da sua personalidade, definidos no componente de personalidade. (2) As regras condição-ação são definidas pela base de categorias³ e pelas regras do componente de personalidade. (3) O componente de raciocínio é implementado utilizando o interpretador AIML.

As divisões (1) e (2) foram feitas de acordo com os princípios apresentados no início deste capítulo. O objetivo dessa divisão, conforme mencionado, é promover a reutilização da base de categoria.

Veremos a seguir detalhes sobre o componente de personalidade e o componente de raciocínio. As mudanças na base de categorias AIML, por estarem ligadas diretamente à etapa de implementação serão apresentadas no Capítulo 5. O *log* de diálogo não será detalhado, pois, já está inserido no interpretador da linguagem AIML.

4.2.1 O Componente de Personalidade

Para definir um agente *chatbot* com personalidade, buscamos utilizar apenas as funcionalidades básicas que permitem programar o seu comportamento através da modelagem da sua personalidade. Sendo assim, escolhemos os seguintes componentes: crenças, elementos de personalidade, regras, e ações.

A Figura 4.2 apresenta o Componente de Personalidade em detalhes. O termo “elemento *i*” refere-se ao *i*-ésimo elemento de personalidade (e.g., traço e estado físico).

³ Conforme apresentado na seção 2.3.5, as categorias AIML podem também ser consideradas como regras condição-ação. Elas utilizam uma sintaxe específica para a identificação da sentença do usuário, parte do componente de raciocínio do *chatbot*.

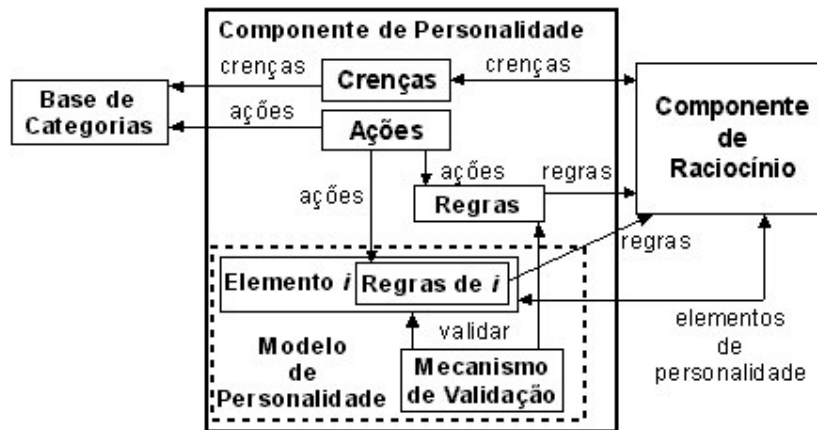


Figura 4.2 – Componente de Personalidade

As crenças definem informações sobre o agente (e.g., nome e idade). Estes dados podem ser acessados pela base de categorias, porém apenas o componente de personalidade pode modificá-los.

Conforme apresentamos no início deste capítulo, para flexibilizar a criação de modelos de personalidade, decidimos fornecer ao *botmaster* um conjunto de elementos de personalidade e um componente, denominado mecanismo de validação, para validar a coerência entre os elementos de personalidade e suas relações. Na seção 4.3 apresentaremos como estes componentes podem ser utilizados para a criação de modelos de personalidade. A definição de cada elemento da personalidade será detalhada na seção 4.3.4, enquanto a modelagem do mecanismo de validação (seção 3.3) é apresentada na seção 4.3.5.

As regras representam o conhecimento do *chatbot* sobre o ambiente e si mesmo. É a partir da avaliação das regras que o robô define que ações serão executadas. As regras utilizam a abordagem simbólica (ou *top-down*), a fim de permitirem ao *botmaster* uma definição mais clara e precisa do comportamento do *chatbot*. Elas são do tipo “condição-ação”, isto é, são descritas através de um conjunto de condições e ações, e serão executadas apenas no caso das condições serem satisfeitas. Além do comportamento do agente (atualização do estado interno e tomada de decisão), as regras podem definir eventos. Os eventos ocorrem quando há uma mudança de estado no componente de personalidade. A seção 4.3.3 apresenta em detalhes o conceito de eventos.

Uma ação é uma abstração para uma seqüência de instruções que devem ser executadas. As ações podem ser chamadas a partir de uma regra, ou a partir de um padrão de saída da base de categorias, representando uma resposta de maneira modular.

Diagrama de Classes

A Figura 4.3 apresenta o diagrama de classes do agente *chatterbot* baseado na arquitetura Persona-AIML, descrito segundo a UML.

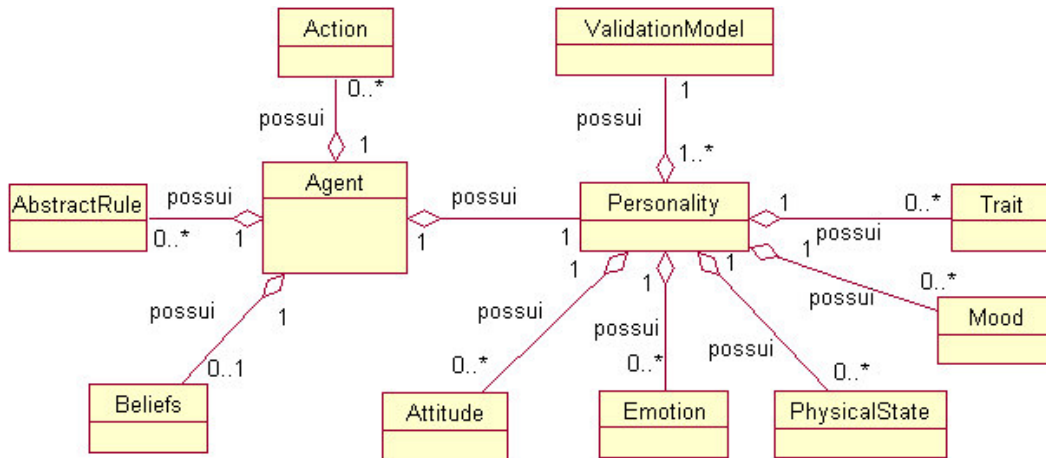


Figura 4.3 – Diagrama de classes do agente Persona-AIML

A Tabela 4.1 apresenta uma descrição resumida das classes apresentadas na Figura 4.3.

Tabela 4.1 – Descrição das classes apresentadas no Diagrama UML (Figura 4.3)

Classe	Descrição
Agent	É a principal classe, sendo responsável pela modelagem do agente. Nela são definidas as crenças, as regras, as ações e a personalidade do agente.
AbstractRule	Descreve uma regra do agente. Ela é composta por condições e ações, ambos descritos em AIML. As ações são executadas apenas caso as condições sejam satisfeitas.
Action	Representa uma ação do agente. Nessa classe é definido o conjunto de instruções que serão executadas quando uma instância da classe é invocada.
Beliefs	Representam o conjunto de crenças do agente.
Personality	Esta classe descreve a personalidade do agente, que é composta por um mecanismo de validação e o conjunto de elementos da personalidade.
ValidationModel	Nesta classe é descrito o mecanismo de validação da personalidade do agente. A partir dessa classe é feita a validação do conjunto de elementos da personalidade e das regras que modificam a sua intensidade.
Trait	Representa um traço de personalidade do agente.
Attitude	Representa uma atitude do agente.
Emotion	Representa uma emoção do agente
Mood	Representa um humor do agente.
PhysicalState	Representa um estado físico do agente.

4.2.2 O Componente de Raciocínio

A Figura 4.4 apresenta o componente de raciocínio do agente. Conforme citado anteriormente, ele é composto pelos módulos de casamento de padrão e de construção da resposta.

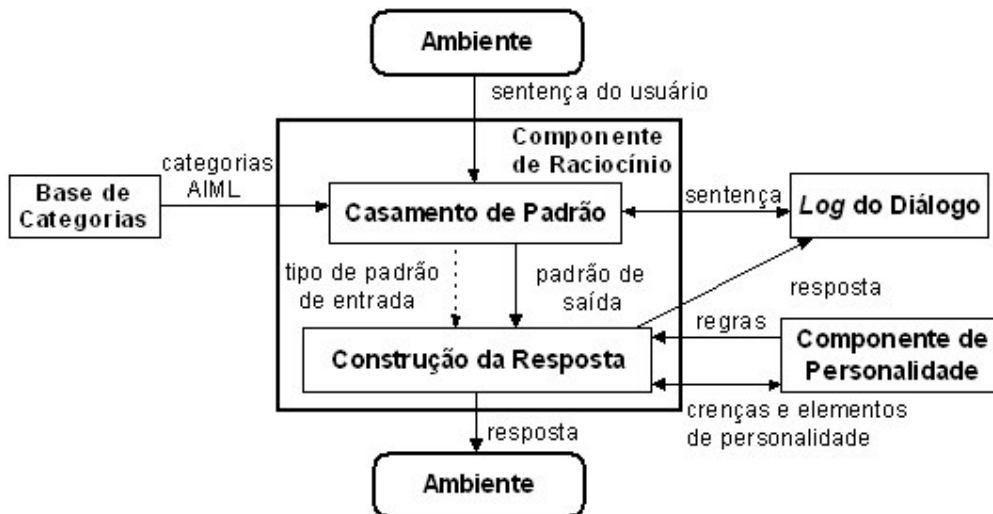


Figura 4.4 – Componente de Raciocínio do agente

Casamento de Padrão

O módulo de casamento de padrão, que faz parte do interpretador AIML, é responsável por identificar o padrão da sentença digitada pelo usuário. Ele determina, dentro da base de categorias, aquela que “casa” com a sentença digitada pelo usuário, levando em consideração o contexto do diálogo (seção 2.5).

Uma vez que diversos padrões de entrada podem ter o mesmo significado semântico (através de redução simbólica), e para manter a independência entre os módulos do componente de raciocínio, adotou-se o uso de uma representação intermediária, denominada “tipo de padrão de entrada”, para representar os dados sobre a sentença digitada pelo usuário (e.g., se ela representa um elogio ao robô). Essa representação não existe para todas as categorias, pois algumas delas podem representar reduções simbólicas (e.g., Figura 2.21) ou chamadas de procedimento e, portanto, nunca “casam” diretamente com as sentenças do usuário.

Os dados sobre o tipo de padrão de entrada são utilizados pelas regras definidas pelo componente de personalidade. Eles são armazenados na memória do agente até serem removidos por alguma regra. Por estar diretamente ligada a implementação, apresentaremos mais detalhes sobre essa modificação no Capítulo 5.

Construção da Resposta

Para a construção da resposta (ações do *chatterbot*) é feita a avaliação das regras definidas no componente de personalidade. A partir desta avaliação, a resposta será montada com base nas ações das regras cujas condições tenham sido satisfeitas. Se, ao final deste processo, nenhuma ação for executada, o padrão de saída obtido no processo de casamento de padrão será retornado para o usuário.

O módulo de construção da resposta possui os seguintes processos: inicialização, atualização do estado interno e tomada de decisão (Figura 4.5).

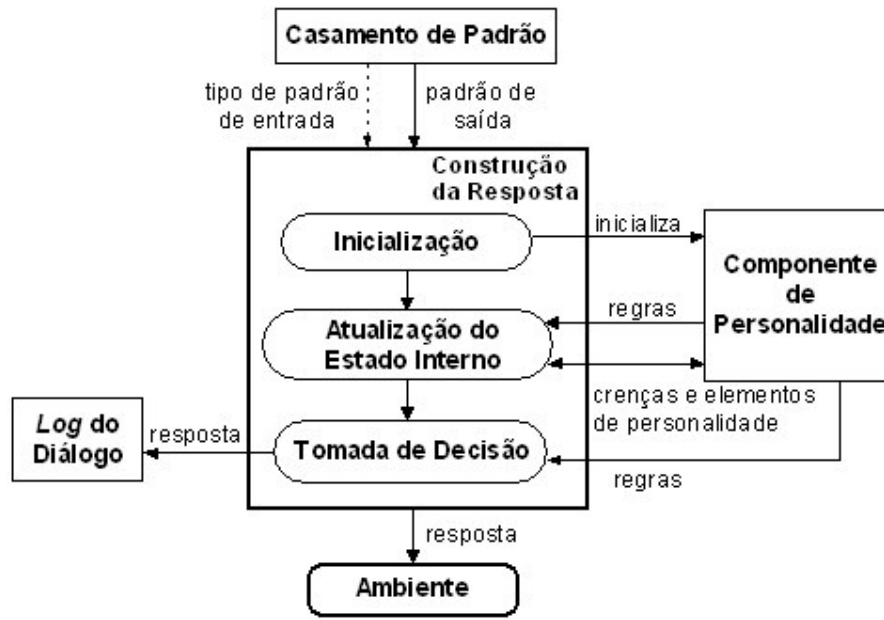


Figura 4.5 – Módulo de construção da resposta

O processo de inicialização é executado apenas na primeira interação do usuário com o *chatbot*, sendo responsável por dar valores iniciais às crenças e aos elementos da personalidade do agente.

À medida que o usuário interage com o *chatbot*, os valores do seu estado interno vão sendo modificados. Para isso, podem ser utilizadas informações obtidas durante o processo de casamento de padrão. Por exemplo, a intensidade da atitude “gostar”, referente ao usuário, é aumentada caso o usuário faça um elogio ao robô. Esse processo é executado antes da tomada de decisão, uma vez que ela depende fundamentalmente da avaliação do estado interno do agente (crenças, elementos de personalidade, etc.).

Após a atualização do estado interno, é realizado o processo de tomada de decisão, durante o qual as regras do componente de personalidade serão avaliadas para a seleção de ações a serem tomadas. Conforme mencionamos anteriormente, caso nenhuma ação seja escolhida, será retornado para o usuário o padrão de saída já selecionado.

Os processos de atualização do estado interno e de tomada de decisão utilizam as regras do componente de personalidade que, em sua definição, indicam em qual dos processos atuam.

4.3 Criação de Modelos de Personalidade

Na seção 3.3, vimos que um modelo de personalidade pode ser descrito através de um conjunto de elementos de personalidade, e de um mecanismo de validação, que define como esses elementos devem ser especificados e como interagem entre si. Analisando as aplicações que utilizam algum modelo de personalidade, verificamos que a maior diferença existente entre eles reside na escolha desse mecanismo de validação, que depende fundamentalmente dos requisitos da aplicação (e.g., personalidade fixa ou capacidade de mudar a personalidade).

Apesar disso, as implementações existentes não consideram uma possível mudança dos requisitos da aplicação, tornando-se necessário refazer o modelo de personalidade a cada nova aplicação, pois não há uma separação clara entre elementos e mecanismo de validação. Para evitar esse tipo de problema, nossa arquitetura prevê dois componentes, a partir dos quais os modelos de personalidade são definidos: o conjunto de elementos de personalidade; e o mecanismo de validação.

O conjunto de elementos de personalidade está baseado em conceitos da Psicologia (seção 3.2.1). Conforme apresentado, este conjunto é composto por traços de personalidade, atitudes, emoções, humor e estados físicos, que são os elementos que comumente são utilizados em implementações de personalidade. A seção 4.3.4 apresenta a descrição da modelagem desses elementos.

O mecanismo de validação, responsável por validar o conjunto de elementos de personalidade que podem ser utilizados pelo modelo e suas relações, é descrito por uma classe específica. Na seção 4.3.5 apresentaremos maiores detalhes sobre a solução adotada.

Embora nossa solução permite uma maior flexibilidade ao *botmaster*, uma vez que ele não precisa se prender a um modelo de personalidade específica, ela deixa a seu cargo as decisões da definição do modelo de personalidade. Sendo assim, ele pode construir um modelo seguindo uma base teórica da Psicologia, ou escolher um modelo *ad hoc*. Esse modelo poderá resultar na criação de *chatterbots* com uma personalidade estável ou instável. Caso o *botmaster* não queira se comprometer em desenvolver um modelo, ele poderá adotar o mecanismo de validação de outro *botmaster*, tendo apenas que definir os elementos da personalidade.

Veremos a seguir os pontos em comum e as diferenças entre nosso modelo e as outras abordagens existentes. Em seguida apresentaremos como foram modelados as regras, os elementos de personalidade e seu mecanismo de validação.

4.3.1 Pontos Comuns

Assim como em outros modelos (seção 3.3.2), a intensidade dos elementos que compõem a personalidade também é representada através de variáveis numéricas, cujos valores devem pertencer a uma faixa pré-definida. Por exemplo, o estado físico *energia*, que representa se o agente está com fome ou não, seria descrito através de uma variável cujo intervalo varia entre [-5, 10]. Os valores negativos significariam “está com fome”, enquanto os positivos, o contrário.

Embora não seja obrigatório, é comum representar estados antagônicos (e.g., triste/alegre, medo/coragem e gostar/odiar) utilizando um único elemento. Isso é feito para prevenir inconsistências no comportamento do agente, como, por exemplo, gostar e odiar uma entidade ao mesmo tempo.

4.3.2 Principais Diferenças

Além da separação entre os elementos de personalidade e o mecanismo de validação, uma diferença importante entre o modelo proposto aqui e as abordagens existentes é que, no nosso caso, cada elemento da personalidade representa uma estrutura ativa, no sentido em que eles podem definir as regras que regem o seu funcionamento, isto é, modificar a sua intensidade. Por exemplo, o humor *alegria*, que representa se o agente está alegre ou triste, pode definir uma regra que muda o estado para feliz ao detectar que o usuário fez um elogio.

Essa solução foi adotada a fim de minimizar um dos problemas da abordagem simbólica, onde, à medida que a base de regras aumenta, cresce a complexidade em manter a sua organização (seção 3.3.1). No nosso caso, é possível gerenciar com maior facilidade as regras do agente, uma vez que elas encontram-se estruturadas de acordo com cada elemento da personalidade. Desta forma, o *botmaster* pode analisar as regras focalizando individualmente cada componente da personalidade.

4.3.3 Regras

Nossas regras são classificadas em duas classes (tipos): comportamentos ou eventos. Enquanto as regras de comportamento são utilizadas no processo de raciocínio do agente, as regras de eventos são utilizadas sempre que o estado interno do agente é modificado. As regras são do tipo condição-ação, isto é, são compostas por uma seqüência de condições (semelhante ao modelo “IF-THEN”) que caso sejam satisfeitas disparam um conjunto de ações para serem executadas.

A Figura 4.6 apresenta o diagrama de classes para as regras, onde comportamento e eventos são representados, respectivamente, pelas classes *AbstractBehaviorRule* e *AbstractEventRule*.

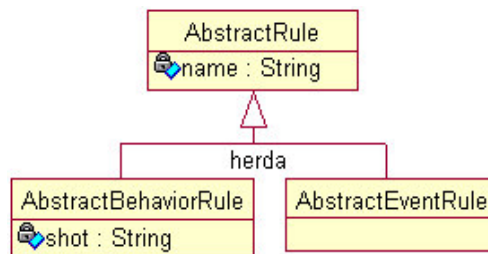


Figura 4.6 – Diagrama de classes para as regras

As regras de comportamento são verificadas pelo módulo de construção da resposta enquanto as condições de pelo menos uma delas são satisfeitas, sendo que as regras que atualizam o estado interno (*StateBehaviorRule*) são verificadas antes daquelas utilizadas no processo de tomada de decisão (*DecisionBehaviorRule*). Uma vez que algumas regras podem assumir ambos os papéis, ou seja, atualizam o estado interno do agente e são responsáveis pela tomada de decisão, existe um terceiro tipo de regra, denominada híbrida (*HybridBehaviorRule*), que é verificada após as regras de atualização do estado interno e antes daquelas de tomada de decisão. É possível determinar que uma regra de comportamento só pode ser executada uma única vez através do atributo *shot*, que por *default* significa que a regra pode ser executada infinitas vezes (Figura 4.7).

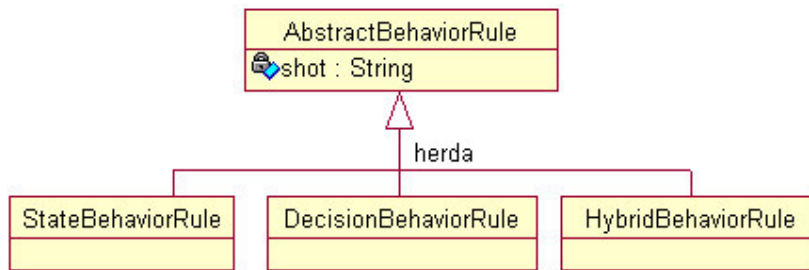


Figura 4.7 – Diagrama de classes para as regras de comportamento

Em nossa arquitetura, os eventos são verificados uma única vez e ocorrem em três situações: (1) mudança de estado de um elemento da personalidade, representada pelo evento *on-change*; (2) criação do elemento da personalidade ou do agente, evento *on-create*; e (3) remoção de um elemento da personalidade, evento *on-destroy*. A Figura 4.8 apresenta o diagrama de classes com a representação dos possíveis eventos.

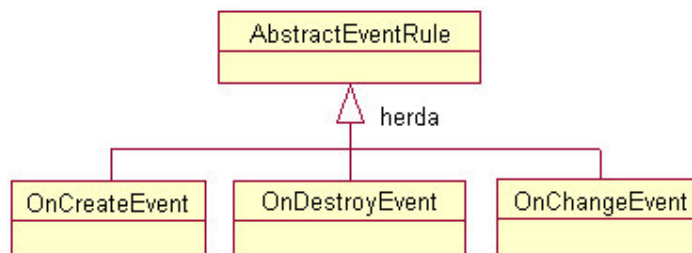


Figura 4.8 – Diagrama de classes para os eventos

Quando há uma mudança de estado em um elemento da personalidade, as regras do evento *on-change* desse elemento são verificadas. Por exemplo, poderíamos definir uma regra que caso a intensidade do humor *alegria* (que indica se o *chatterbot* está alegre ou triste) atinja o estado correspondente a *triste* faça, com que ele comece o agente comece a chorar.

A criação do elemento da personalidade ou do agente faz com que as regras do evento *on-create* sejam verificadas. Por exemplo, podemos definir uma regra para a emoção *raiva* que, ao ser criada, faça com que o *chatterbot* mude a atitude *gostar*, que indica se o *chatterbot* gosta da entidade (e.g., usuário) para o estado referente à *não gostar*.

O evento *on-destroy* ocorre no momento em que um elemento da personalidade é removido. Podemos exemplificar o uso desse evento, estendendo o

exemplo anterior, definindo uma regra que faça com que o *chatterbot* volte a gostar da entidade assim a emoção *raiva* deixe de existir.

Na Figura 4.9 apresentamos um exemplo de uma pseudo-regra de comportamento (tipo *DecisionBehaviorRule*). Esta regra define que o *chatterbot* não responde uma pergunta do usuário caso não goste dele.

```
REGRA nome = responder dúvida
CONDIÇÃO
  se foi feita uma pergunta então
    execute a ação da regra
  fim-se
FIM-CONDIÇÃO
AÇÃO
  escolha
    analisando atitude = gostar, entidade = usuário
    caso estado="não gosta"
      diga "não respondo para você"
    fim-caso
  senão
    execute a resposta da base de categorias
  fim-senão
fim-escolha
FIM-AÇÃO
FIM-REGRA
```

Figura 4.9 – Exemplo de Pseudo-Regra de Comportamento

4.3.4 Elementos de Personalidade

Todos os elementos de personalidade são descritos como uma subclasse de *AbstractElement*. Essa classe define o nome do elemento (atributo *name*), as regras, a descrição dos estados do elemento, a intensidade (atributo *value*) e os valores mínimos e máximos (atributos *min* e *max*), isto é, a faixa de valores válidos para a intensidade (Figura 4.10).

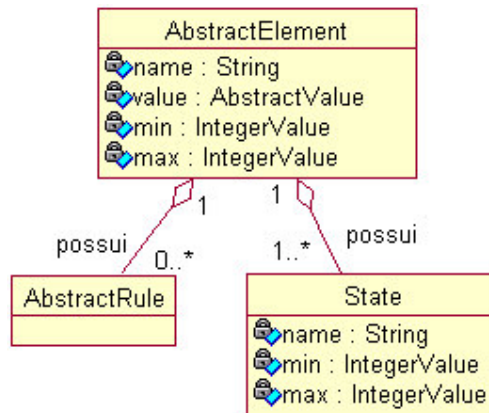


Figura 4.10 – Definição da classe *AbstractElement*

Os estados de um elemento são representados por instâncias da classe *State*. Cada instância descreve o nome (atributo *name*), o limite inferior e superior de sua faixa de valores (atributos *min* e *max*) e sua intensidade (atributo *value*). Por exemplo, um estado físico, denominado *energia*, definiria os valores mínimos e máximos dentro da faixa [-5, 10] e possuiria três estados: exausto, cansado, e normal. O estado *exausto* teria os limites inferiores e superiores variando entre [-5, 0]. O estado *cansado*, por sua vez, teria a faixa de valores [1, 5]. Por fim, os valores entre [6, 10] definiriam o estado *normal*. A partir dessas informações, poderia-se também definir que, inicialmente, o seu estado é normal, atribuindo o valor 6 para a intensidade desse elemento (Figura 4.11).

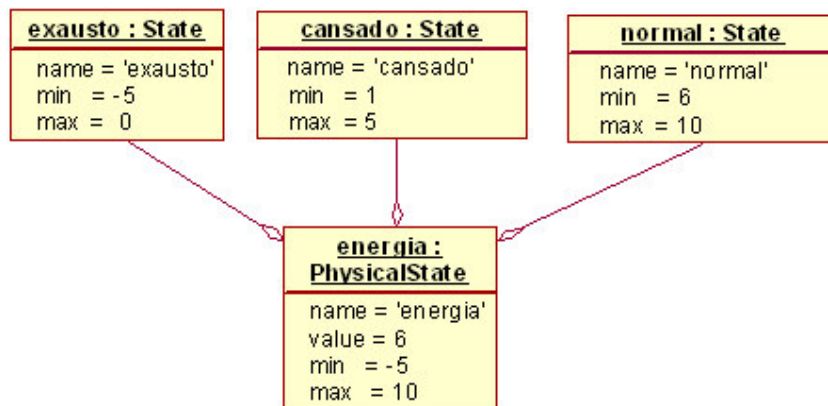


Figura 4.11 – Diagrama de objetos para o estado físico

Para permitir a flexibilidade do modelo de personalidade aqui proposto, o atributo que armazena a intensidade do elemento pertence a uma das subclasses de *AbstractValue* (Figura 4.12), sendo eles: *IntegerValue*; *RandomValue*; e *ExpressionValue*.

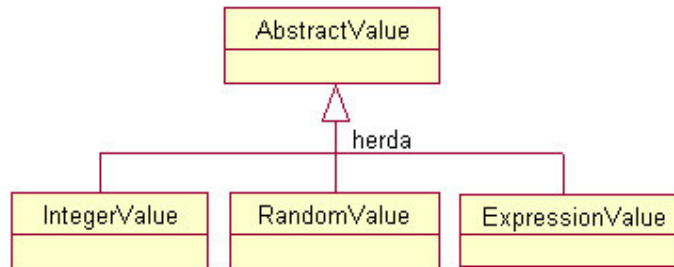


Figura 4.12 – Definição da classe *AbstractValue*

A classe *IntegerValue* define um valor numérico inteiro, que será a intensidade inicial do elemento, isto é, será utilizado no momento da criação do agente. Esse valor deve pertencer à faixa de valores válidos.

Assim como a classe *IntegerValue*, *RandomValue* determina o valor inicial de um elemento. Entretanto, esse valor é escolhido aleatoriamente dentro da faixa de valores válidos para o elemento. Essa escolha permite um maior dinamismo no momento de criação do agente, uma vez que, para cada usuário, o *chatbot* poderá se apresentar inicialmente com diferentes estados internos.

A classe *ExpressionValue* determina o valor do elemento a partir da avaliação de uma expressão, cujo resultado será restringido pela faixa de valores válidos. O valor da intensidade desse elemento só poderá ser modificado a partir da mudança das variáveis que compõem a expressão. Esse tipo atende a necessidade de alguns modelos de personalidade, onde a intensidade de um elemento pode ser definida como uma expressão que envolve outros elementos da personalidade do agente. A Figura 4.13 apresenta um exemplo de código, extraído do Projeto *Oz* (REILLY, 1996), que ilustra essa situação.

Exemplo de Código
<code>unfriendly := distress/3 + anger/2</code>
Comentário
A intensidade (valor) do traço não-amigável (<i>unfriendly</i>) é definida pelo somatório de um terço do sofrimento (<i>distress</i>) e metade da raiva (<i>anger</i>).

Figura 4.13 – Exemplo de código do modelo *Oz* de personalidade

A seguir apresentaremos uma descrição de cada elemento da personalidade segundo nossa proposta.

Humor e Estados Físicos

Os estados físicos e humores do agente são representados, respectivamente, por instâncias das classes *Mood* e *PhysicalState*. Essas classes são subclasses de *AbstractElementWithDelay* (Figura 4.14).

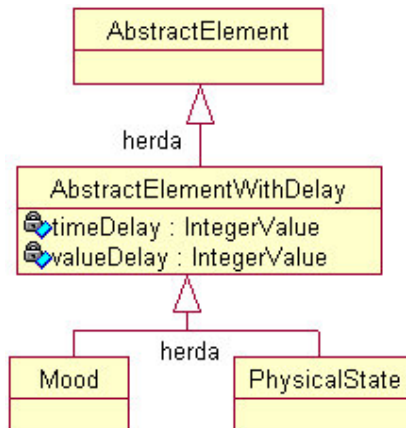


Figura 4.14 – Diagrama de classe para *Mood* e *PhysicalState*

A classe *AbstractElementWithDelay* estende *AbstractElement* permitindo ao *botmaster* incluir um contador (atributo *timeDelay*) que, com o passar o tempo, modifica a intensidade do elemento da personalidade através de um valor fixo (atributo *valueDelay*).

Para o humor, o contador pode ser utilizado para modelar, de maneira simplificada, o mecanismo que permite com que um *chatterbot* possa mudar seu humor à medida que o tempo passa. Por exemplo, um *chatterbot* com personalidade alegre, mesmo após ficar de mal-humor (e.g., ter recebido uma má notícia), volta a ficar de bom humor com o passar do tempo.

Nos estados físicos, o contador pode controlar situações em que, com o passar do tempo, a intensidade do estado físico é modificada. Por exemplo, à medida que o tempo passa, a energia do *chatterbot* diminui.

Atitudes e Emoções

As atitudes e emoções representam elementos mais complexos da personalidade pois, conforme vimos na seção 3.2.1, são orientadas a alguma entidade (pessoas, animais ou objetos) e sofrem a influência dos sentimentos. Para modelar essa

definição, criamos a classe *AbstractElementWithObjects*, que é uma extensão da classe *AbstractElementWithDelay*. A intensidade, os valores mínimos e máximos, o contador, e as regras definidas pela classe *AbstractElementWithDelay* são considerados globais para todas as entidades de uma atitude ou emoção (Figura 4.15).

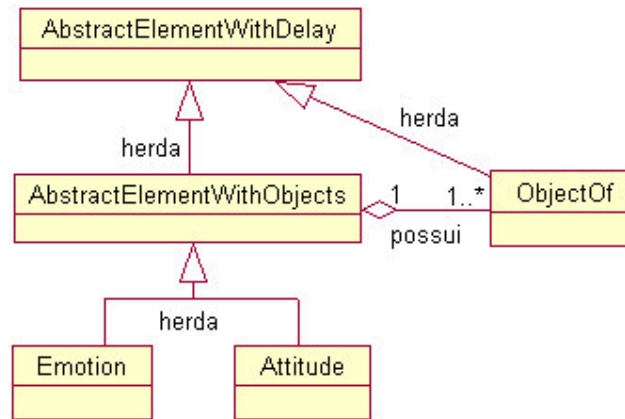


Figura 4.15 – Diagrama de classes para *Attitude* e *Emotion*

A classe *ObjectOf* representa uma entidade a qual uma atitude ou emoção podem se referenciar. Além do atributo *name* (obtido da classe *AbstractElement*), possui o atributo *type*. Esses atributos representam, respectivamente, o nome da entidade (e.g, Paulo) e o tipo (ou grupo) ao qual ela pertence (e.g, pessoa). Além desses atributos, a classe *ObjectOf* pode definir regras e valores específicos (intensidade, valor mínimo e valor máximo) diferentes daqueles que foram definidos globalmente. As instancias de *ObjectOf* compartilham o conjunto de estados definidos na classe que as agrega, ou seja, as atitudes e as emoções.

As entidades podem ser concretas ou abstratas. Uma entidade concreta é definida utilizando o atributo *name* e, opcionalmente, o atributo *type*. Por exemplo, um *chatbot* ranzinza pode definir a intensidade da atitude *gostar* representando o estado “não gosta” para todas as entidades. Entretanto, podemos definir que existe uma exceção, pois o agente gosta de comer macarrão. Para isso, é criada a entidade macarrão (nome *macarrão* e tipo *comida*), cuja intensidade representa o estado “gosta”. A Figura 4.16 apresenta um diagrama de objetos ilustrando o exemplo.

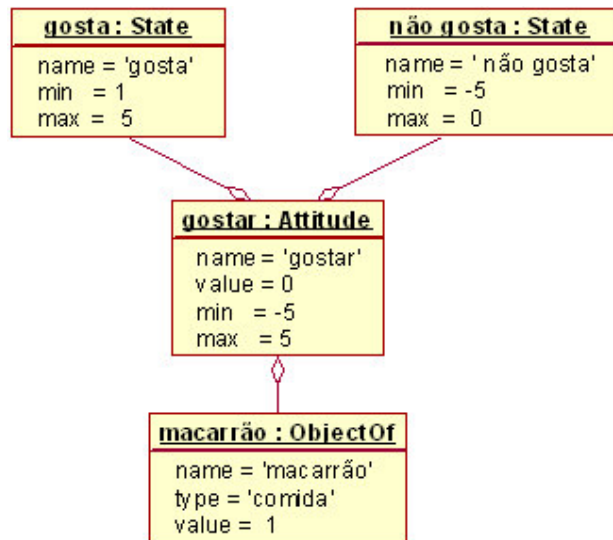


Figura 4.16 – Diagrama de objetos exemplificando uma entidade concreta

Uma entidade abstrata é definida utilizando apenas o atributo *type*. Isso é feito para capturarmos a idéia de sentimentos. Uma entidade abstrata pode definir regras e valores específicos para todas as entidades concretas cujo tipo corresponde a ela. Utilizando o exemplo anterior, podemos especificar que, além de gostar de macarrão, o *chatterbot* gosta de cachorros. Para isso, definimos uma entidade abstrata cujo tipo é cachorro. A Figura 4.17 apresenta um diagrama de objetos ilustrando o exemplo.

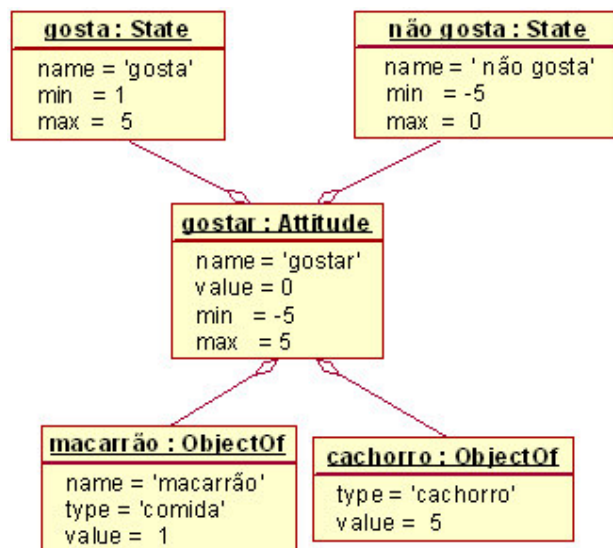


Figura 4.17 – Diagrama de objetos exemplificando uma entidade abstrata

As atitudes e as emoções são representadas, respectivamente, pelas classes *Attitude* e *Emotion*, subclasses de *AbstractElementWithObjects*. As emoções diferem das atitudes em dois aspectos: (1) possuem um estado (*State*) *built-in* cuja faixa de

valores é $[0, 0]$, sendo utilizado para definir o momento em que o sentimento deixa de existir (evento *on-destroy*); (2) ao contrário das atitudes, as entidades concretas definidas por uma emoção não serão criadas automaticamente junto com o agente, ou seja, o agente é criado com um conjunto de atitudes, porém não possui nenhuma emoção.

Traços de Personalidade

Conforme vimos no Capítulo 3, os traços de personalidade são definidos, geralmente, como estruturas estáveis, ou seja, não variam de acordo com o tempo. Quando há mudança, esta é feita a partir das situações que ocorrem com o agente. Aqui, os traços são modelados a partir de uma subclasse de *AbstractElement*, denominada *Trait* (Figura 4.18).

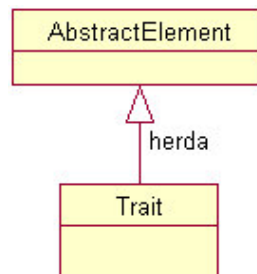


Figura 4.18 – Diagrama de classes para *Trait*

4.3.5 Mecanismo de Validação

O mecanismo de validação de um modelo de personalidade deve ser definido através de uma subclasse de *ValidationModel*. Tendo acesso ao agente (*Agent*) e, conseqüentemente, à sua personalidade (*Personality*), é possível verificar se os elementos de sua personalidade e a forma com que eles interagem estão de acordo com as especificações do projetista do *chatterbot* (Figura 4.19).

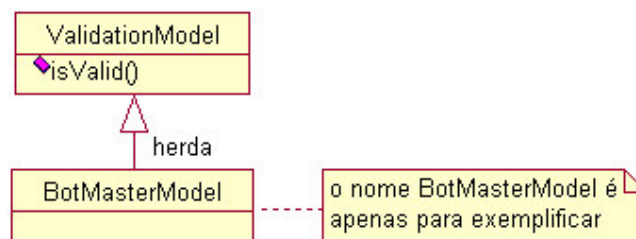


Figura 4.19 – Diagrama de classes para *ValidationModel*

A validação do conjunto de elementos da personalidade é feita através da consulta ao agente (atributo *owner*). Por exemplo, se o projetista define que o seu modelo de personalidade possui apenas emoções e traços, o *botmaster* deve criar uma classe (extensão de *ValidationModel*) para verificar se existem outros elementos na personalidade do agente (e.g., atitudes) e, caso isso ocorra, reportar uma exceção para que seja feita a correção. Em outro exemplo, se a intensidade de um traço de personalidade é definida a partir de um conjunto de emoções, deve-se verificar se a intensidade do traço é uma instância de *ExpressionValue*. Além disso, deve ser verificado se a expressão é formada pelas emoções definidas pelo projetista.

Consultando as regras do agente e dos elementos de sua personalidade, é possível determinar como os elementos da personalidade interagem. Por exemplo, se um traço é definido como estável, pode-se consultar no agente se existe alguma regra que modifica a intensidade do traço. Caso isso ocorra, uma nova exceção é reportada.

4.4 Passos para a Criação de *Chatterbots* com Personalidade

Nesta seção resumimos as atividades necessárias para o desenvolvimento de *chatterbots* com personalidade seguindo a Persona-AIML:

Passo 1: Construção de uma base de categorias. Essas categorias devem ser responsáveis pela identificação da sentença do usuário. Para cada categoria é preciso determinar se ela representa redução simbólica, chamada de procedimento ou resposta para o usuário. Nas categorias onde são definidas respostas para o usuário, deve-se determinar qual o tipo da sentença digitada pelo usuário, a fim de que o componente de raciocínio possa utilizar essas informações no processamento das regras do *chatterbot*.

Passo 2: Definição do modelo de personalidade. Para definir esse modelo deve-se determinar quais elementos de personalidade poderão ser utilizados pelo *chatterbot* e qual deverá ser o mecanismo de validação do modelo. Esse mecanismo definirá como os elementos irão interagir entre si.

Passo 3: Definição dos comportamentos e elementos da personalidade do *chatterbot*. Nesta fase, o *botmaster* deverá decidir que comportamentos devem ser modelados pela personalidade do *chatterbot*. Baseado nos comportamentos definidos, o *botmaster* escolherá os elementos de personalidade que o *chatterbot* apresentará. Por exemplo, se for definido que o robô fica *triste com o usuário* ao receber um insulto,

seria preciso incluir o elemento de personalidade emoção. Esse comportamento seria codificado através de uma regra do elemento emoção.

Passo 4: Testes. Nessa etapa, deverão ser feitos os testes utilizando o *chatbot* desenvolvido. É nesta etapa que devem ser coletados os logs de diálogo com usuários para que possam ser analisados na etapa seguinte.

Passo 5: Análise dos resultados. Nessa etapa, deverão ser analisadas as respostas do robô, a fim de verificar se os comportamentos definidos são suficientes para caracterizar a personalidade que se deseja modelar. Caso o resultado dessa análise seja insatisfatório, é preciso verificar qual (ou quais) dos passos anteriores precisará ser refeito para atingir o resultado esperado.

4.5 Conclusões

Apresentamos neste capítulo a arquitetura Persona-AIML para criação de *chatbots* capazes de exibir personalidade em suas interações com os usuários. Suas principais características são: (1) reuso das bases de categorias AIML, uma vez que a personalidade do *chatbot* passa a ser modelada em um componente separado; e (2) definição de uma arquitetura que permita a construção de diversos modelos de personalidade.

Sendo a base AIML composta por categorias cuja ordem é de milhares, o *botmaster* é obrigado a analisar cada uma, caso deseje modelar um determinado comportamento do *chatbot*. Além disso, a ausência de componentes *built-in* para a modelagem dos elementos de personalidade dificulta a construção de modelos de personalidade robustos e reutilizáveis. A introdução do componente de personalidade procura suprir essas limitações da linguagem AIML.

Em nosso trabalho, consideramos que a definição do modelo de personalidade do *chatbot* deve ser feita pelo projetista do *chatbot*. Se, por um lado, esta solução permite uma maior flexibilidade da arquitetura, por outro, o *botmaster* deve especificar através de uma classe como é feito o mecanismo de validação do modelo de personalidade. Essa responsabilidade pode ser minimizada, na medida que um *botmaster* poderá adotar um mecanismo de validação definido e testado por outro *botmaster* cujos objetivos sejam semelhantes.

4. A Arquitetura Persona-AIML

No próximo capítulo serão apresentados a implementação da arquitetura Persona-AIML e o protótipo desenvolvido com base em nossa arquitetura.

5

Implementação e Protótipo

O objetivo deste capítulo é detalhar a implementação da arquitetura Persona-AIML e o desenvolvimento de um protótipo de *chatbot* com personalidade a partir dessa arquitetura.

A implementação, feita na linguagem Java, consistiu basicamente de três atividades: (1) o desenvolvimento de uma especificação em XML para a definição das bases de categorias e de personalidade; (2) a adição de novos comandos à linguagem AIML relacionados ao uso de personalidade em *chatbot*; (3) a criação de um conversor da arquitetura Persona-AIML para AIML padrão.

Para descrever essa primeira atividade, foi necessária a utilização de diversos códigos em XML neste capítulo. No intuito de facilitar o entendimento desses exemplos, utilizaremos a visualização fornecida pelo *browser* Microsoft[®] Internet Explorer¹. De forma semelhante ao que acontece no programa Explorer do Windows[®], nessa visualização os símbolos “+” e “-” são utilizados para indicar o nível de abstração que está sendo utilizado, indicando que a *tag* ao lado (um nó na árvore XML) é composta por um ou mais sub-nós. A diferença entre esses símbolos é que o “+” indica que os sub-nós foram omitidos na visualização, enquanto “-”, por sua vez, indica que os sub-nós estão sendo apresentados.

A base de categorias do protótipo foi obtida através da adaptação da base do *chatbot* Pixel. O modelo de personalidade adotado foi baseado no Modelo dos Cinco Grandes Fatores (*Big Five*). Cenários foram definidos para permitir a rápida avaliação do protótipo e, conseqüentemente, da arquitetura Persona-AIML.

5.1 Decisões de Projeto

Conforme vimos no Capítulo 2, as principais virtudes de AIML são o compromisso com a comunidade de código livre e o foco na sua especificação. Essas

¹ <http://www.microsoft.com/windows/ie>

características fizeram com que AIML seja independente de linguagem da programação adotada na criação do interpretador. Tudo isso permitiu o surgimento de uma grande comunidade de usuários voltados ao seu desenvolvimento.

Como forma de manter essas características de AIML é importante que a implementação da arquitetura Persona-AIML compartilhe os mesmos princípios. Sendo assim, tomamos como decisão de projeto implementar nossa arquitetura de forma que ela possa ser executada a partir de um interpretador AIML padrão. Para isso, desenvolvemos um conversor (seção 5.5), que permite a adoção dessa solução.

Essa decisão é particularmente favorável para nós e para a comunidade AIML. Tiramos vantagem por não precisarmos criar um novo interpretador, cuja implementação demandaria muito esforço. A comunidade AIML, por sua vez, se beneficia por não precisar reescrever os interpretadores existentes.

Toda a implementação da Persona-AIML foi feita na linguagem de programação Java. Além de ser independente de plataforma, Java é utilizada no desenvolvimento do interpretador *ProgramD*. Esse interpretador é, atualmente, o mais utilizado pela comunidade AIML devido a sua estabilidade e arquitetura modular. Logo, caso não fosse possível executar nossa arquitetura utilizando um interpretador AIML padrão, seria mais fácil desenvolver um novo interpretador, pois poderíamos reutilizar o código do interpretador *ProgramD*.

5.2 A Base de Categorias

No capítulo anterior, vimos que a base de categorias AIML foi modificada para se adequar a nossa arquitetura. Nessa modificação, as categorias além de informarem o padrão de saída, devem indicar qual o tipo do padrão de entrada. Essa solução foi adotada uma vez que as categorias AIML podem representar reduções simbólicas ou chamadas de procedimento. Sem essa informação o componente de raciocínio seria executado indevidamente, “pensando” que as reduções simbólicas e chamadas de procedimentos seriam sentenças digitadas pelo usuário.

Para evitar esse problema, o *<template>* de uma categoria foi modificado adicionando as *tags <pattern-type>* e *<do>*. A primeira é responsável por agrupar os dados sobre o tipo de padrão de entrada, enquanto a segunda indica o padrão de saída e, através do atributo *what*, informa se a categoria é uma resposta para o usuário (valor

“*talk*”), redução simbólica ou chamada de procedimento. Estes dois últimos são representados pelo valor “*process*”, pois são equivalentes para módulo de construção da resposta (4.2.2). A Figura 5.1 apresenta o exemplo da Figura 2.21, reescrito conforme a organização proposta por nossa arquitetura.

```
- <category>
  <pattern>ATE MAIS</pattern>
  - <template>
    - <do what="process"><srai>TCHAU</srai></do>
  </template>
</category>
- <category>
  <pattern>TCHAU</pattern>
  - <template>
    + <pattern-type>
    - <do what="talk">NOS VEMOS OUTRO DIA, SÃO <srai>HORAS</srai></do>
  </template>
</category>
- <category>
  <pattern>HORAS</pattern>
  - <template>
    - <do what="process"><!-- cálculo da hora --></do>
  </template>
</category>
```

Figura 5.1 – Exemplo da Figura 2.21 convertida para o formato Persona-AIML

Cada dado sobre o tipo de padrão de entrada é representado pela *tag* `<pattern-data>`. Além do nome (e.g., *bom dia*), os dados são agrupados em tipos (e.g., *saudação*) e, para que possam ser simuladas situações em que o *chatbot* está interagindo com várias entidades que não sejam o usuário (e.g., no caso de um jogo), deve-se informar a sua origem de uma sentença, isto é, quem a enviou. Esses atributos são representados, respectivamente pelas sub-*tags* `<name>`, `<type>` e `<source>`. O *botmaster* deve especificar pelo menos o nome ou o tipo de um dado, enquanto a origem é opcional. O atributo *for* da *tag* `<pattern-data>` é utilizado para adicionar (valor “add”) ou remover (valor “remove”) um dado. Este último valor é utilizado por uma regra do componente de personalidade, indicando que o dado foi processado.

A Figura 5.2 apresenta um exemplo de uso da *tag* `<pattern-data>`. Neste exemplo a categoria representa uma pergunta ao robô (atributo tipo) referente à sua idade (atributo nome). Através de uma regra no componente de personalidade, poderíamos condicionar a resposta ao do robô gostar dele.

```
- <category>
  <pattern>QUAL A SUA IDADE</pattern>
  - <template>
    - <pattern-type>
      - <pattern-data for="add">
        <name>idade</name>
        <type>pergunta</type>
        <source>usuario</source>
      </pattern-data>
    </pattern-type>
    - <do what="talk"> Eu tenho <get name="botidade" /> </do>
  </template>
</category>
```

Figura 5.2 – Exemplo de uso da tag `<pattern-type>`

5.3 O Componente de Personalidade

Uma vez que o *botmaster* está acostumado com a sintaxe XML, é mais conveniente para ele especificar o Componente de Personalidade (seção 4.2.1) do agente² utilizando essa linguagem. Além disso, ele não se vê obrigado a conhecer a linguagem Java. Dessa forma, criamos um programa (*parser*) para ler o componente de personalidade, descrito em XML, e gerar objetos Java correspondentes.

Por simplificação da implementação, o mecanismo de validação do modelo de personalidade não é descrito em XML, devendo ser escrito em Java. Contudo, esse problema é minimizado, uma vez que o mecanismo de validação só precisa ser escrito uma única vez, podendo ser compartilhado pelos *botmasters*.

A Figura 5.3 apresenta um exemplo de arquivo XML que define o Componente de Personalidade. Nesta figura, podemos identificar as seguintes *tags*: `<agent>`, `<beliefs>`, `<personality>`, `<attitudes>`, `<emotions>`, `<moods>`, `<physical-states>`, `<traits>`, `<events>` e `<behavior>`.

² Neste capítulo utilizaremos as palavras agente e *chatterbot* com o mesmo significado.

```
<?xml version="1.0" encoding="UTF-8" ?>
<agent name="MeuChatterbot">
+ <beliefs>
- <personality validation="MeuModelo">
+ <attitudes>
+ <emotions>
+ <moods>
+ <physical-states>
+ <traits>
</personality>
+ <actions>
+ <events>
+ <behavior>
</agent>
```

Figura 5.3 – Visão geral de um arquivo de definição do componente de personalidade

A tag `<agent>` é a tag “raiz” do documento XML e o atributo *name* representa o nome dado ao *chatterbot*. Esta tag é composta por `<beliefs>`, `<actions>`, `<personality>`, `<events>` e `<behavior>`.

A tag `<beliefs>` define o conjunto de crenças do *chatterbot*.

A tag `<action>` concentra a definição de ações do *chatterbot*.

A tag `<personality>` agrupa os elementos da personalidade do *chatterbot*. São elas: `<attitudes>`, `<emotions>`, `<moods>`, `<physical-states>` e `<traits>`. Essas tags definem, respectivamente, as atitudes, as emoções, os humores, os estados físicos e os traços de personalidade do *chatterbot*.

As tags `<events>` e `<behavior>` são utilizadas para agrupar as regras do *chatterbot*. Enquanto `<events>` agrupa os eventos, `<behavior>` concentra as regras utilizadas no processo de raciocínio. Conforme vimos no capítulo anterior, as regras podem ser definidas diretamente no nível do agente (como na Figura 5.3), ou pelos elementos da personalidade.

A seguir detalharemos, a partir de exemplos, como é feita a definição das tags aqui apresentadas.

5.3.1 Crenças e Ações: Tags `<beliefs>` e `<actions>`

Conforme visto no capítulo anterior, as crenças representam informações sobre o agente, e as ações representam abstrações para um conjunto de instruções. Ambas as tags são formadas por comandos AIML. Entretanto, a tag `<beliefs>` pode

conter apenas *tags* `<set>`. As variáveis definidas por essas *tags* apenas podem ser modificadas pelo Componente de Personalidade, evitando assim que a Base de Categorias modifique as crenças do *chatterbot* (Figura 5.4).

```
- <beliefs>
  <set name="apelido">pixel</set>
  <set name="idade">20 anos</set>
  <set name="sexo">masculino</set>
</beliefs>
```

Figura 5.4 – Exemplo de definição de crenças em XML

Representando uma abstração para um conjunto de instruções, as ações ajudam no desenvolvimento de um sistema modular. Elas podem ser invocadas pelas regras, ou pela base de categorias através da *tag* `<call-action>` (seção 5.4). Os parâmetros passados pela *tag* `<call-action>` podem ser acessados através da *tag* `<get-param>`. Esta *tag* possui o atributo *index*, que indica o número do parâmetro a ser consultado. Caso este atributo seja omitido, assume-se o valor “1”.

A Figura 5.5 apresenta dois exemplos de ações. A primeira define como o *chatterbot* deve ser apresentar ao usuário. A segunda ilustra o uso da *tag* AIML `<random>`. Essa *tag* retornará de maneira randômica o conteúdo de umas das *tags* ``.

```
- <actions>
- <action name="saudação">
  Oi, eu sou <get name="apelido" />.
</action>
- <action name="ajuda">
- <random>
  <li>sinto muito, mas não vou lhe ajudar.</li>
  - <li><srai> <get-param /></srai> </li>
</random>
</action>
</actions>
```

Figura 5.5 – Exemplo de ações em XML

5.3.2 Regras: *Tags* `<events>` e `<behavior>`

No Capítulo 4 vimos que uma regra pode representar um evento, ou ser utilizada no processo de raciocínio do agente. Em XML, as regras de cada um desses tipos são agrupadas, respectivamente, pelas *tags* `<events>` e `<behavior>`. Uma regra é definida através da *tag* `<rule>`, cujo atributo *name* é obrigatório. As regras são

compostas por um conjunto de condições e ações. Esses conjuntos são agrupados, respectivamente pelas *tags* `<condition>` e `<action>`. Caso as condições sejam omitidas, assumimos que elas sempre são satisfeitas.

Os eventos suportados por nossa arquitetura são: (1) *on-create*, que é disparado na criação do agente ou na criação de algum elemento da personalidade; (2) *on-change*, que ocorre na mudança de estado de algum elemento da personalidade; e (3) *on-destroy*, que é utilizado quando um elemento da personalidade é removido (Figura 5.6).

```
- <events>
- <on-create>
- <rule name="NomeDaRegra">
+ <condition>
+ <action>
</rule>
</on-create>
+ <on-change>
+ <on-destroy>
</events>
```

Figura 5.6 – Definição geral das regras em XML

As regras de comportamento são verificadas pelo módulo de construção da resposta (4.2.2) e, de acordo com a ordem em que são verificadas, elas pertencem a uma das seguintes classes: (1) *StateBehaviorRule*, no processo de atualização do estado interno; (2) *DecisionBehaviorRule*, no processo de tomada de decisão; e (3) *HybridBehaviorRule*, que ocorre entre os dois primeiros processos. Em XML, as classes das regras de comportamento são representadas, respectivamente, pelas *tags* `<state>`, `<decision>` e `<hybrid>`. Essas regras possuem o atributo *shot*, que determina se as ações definidas por elas podem ser executadas pelo módulo de construção da resposta uma única vez (*once*) ou infinitas vezes (*infinite*) durante uma interação com o usuário. Por *default*, o atributo *shot* indica que as ações podem ser executadas infinitas vezes (Figura 5.7).

```
- <behavior>
- <state>
  - <rule name="Regra1" shot="once">
    + <condition>
    + <action>
    </rule>
  </state>
- <hybrid>
  + <rule name="Regra2">
  </hybrid>
- <decision>
  + <rule name="Regra3" shot="infinite">
  </decision>
</behavior>
```

Figura 5.7 – Exemplo de regras de comportamento em XML

As condições e ações de uma regra são formadas por comandos AIML. Uma vez que AIML não define expressões booleanas, o *botmaster* deve especificar explicitamente, através da tag *<do-action>*, que as condições de uma regra foram satisfeitas. A Figura 5.8 apresenta um exemplo simplificado de como as condições e ações podem ser definidas.

```
- <behavior>
+ <state>
+ <hybrid>
- <decision>
  - <rule name="Regra" shot="once">
    - <condition>
      - <condition name="variavel" value="true">
        <do-action />
      </condition>
    </condition>
    - <action>
      <call-action name="saudação" />
    </action>
  </rule>
</decision>
</behavior>
```

Figura 5.8 – Condições e ações de uma regra.

5.3.3 Elementos de Personalidade: Tag *<personality>*

A tag *<personality>* agrupa o conjunto de elementos de personalidade e determina através do atributo *validation* qual o mecanismo de validação que deve ser utilizado. Ao ler este atributo, o *parser* verifica se entre as classes registradas pela arquitetura, existe alguma que corresponda ao mecanismo solicitado.

No capítulo anterior, definimos que todos os elementos de personalidade são subclasses de *AbstractElement*. Sendo assim, eles devem especificar o nome, a intensidade do elemento, a faixa de valores válidos, o conjunto de estados e podem definir regras. Estas, conforme apresentado, podem ser definidas também no nível do agente. Portanto, antes de explicarmos cada elemento da personalidade, apresentaremos as definições comuns para cada elemento.

Definições Comuns

Para exemplificar a definição do conjunto de atributos da classe *AbstractElement* em XML, tomaremos como base o exemplo apresentado na Figura 4.11, cuja representação é descrita pela Figura 5.9.

```
- <physical-state name="energia" value="6" min="-5" max="10">
- <states>
  <state name="exausto" min="-5" max="0" />
  <state name="cansado" min="1" max="5" />
  <state name="normal" min="6" max="10" />
</states>
+ <events>
+ <behavior>
</physical-state>
```

Figura 5.9 – Exemplo apresentado na Figura 4.11 descrito em XML

Os atributos da tag *<physical-state>* são definidos utilizando o mesmo nome dos atributos correspondentes na classe *AbstractElement*, isto é, os atributos *name*, *value*, *min* e *max*. No exemplo apresentado, o valor “0”, assim como qualquer outro valor numérico inteiro, corresponde a um objeto da classe *IntegerValue*. Em outros exemplos que serão apresentados ao longo desta seção, mostraremos como são definidos as demais subclasses de *AbstractValue*.

A tag *<states>* agrupa o conjunto de estados válidos para o elemento da personalidade. Cada estado é definido por uma tag *<state>*, cujos atributos correspondem aos apresentados pela classe *State* (seção 4.3.4).

Humor e Estados Físicos: Tags *<moods>* e *<physical-states>*

As tags *<moods>* e *<physical-state>* agrupam, respectivamente, a definição dos humores (classe *Mood*) e estados físicos (classe *Physical-State*) do agente.

Cada humor e estado físico são descritos em XML, respectivamente, pelas tags `<mood>` e `<physical-state>`.

Conforme apresentado no capítulo anterior, as classes *Mood* e *PhysicalState* descendem de *AbstractElementWithDelay*. Logo, elas podem definir os atributos *timeDelay* e *valueDelay*, que assumem os mesmos nomes na representação em XML. A Figura 5.10 apresenta um exemplo de como objetos da classe *Mood* podem ser definidos em XML.

```
- <moods>
- <mood name="felicidade" value="random" min="-5" max="5" timedelay="9" valuedelay="-1">
- <states>
  <state name="triste" min="-5" max="-1" />
  <state name="normal" value="0" />
  <state name="feliz" min="1" />
</states>
+ <events>
+ <behavior>
</mood>
<!-- continua -->
</moods>
```

Figura 5.10 – Exemplo de descrição de objetos da classe *Mood* em XML

No exemplo acima, a *string* “random” do atributo *value* define que o valor da intensidade do humor *felicidade* é membro da classe *RandomValue*, isto é, será um valor randômico. Na definição do estado *normal*, o atributo *value* é uma simplificação para a faixa de valores [valor, valor]. Na definição do estado *feliz* é utilizado apenas o valor mínimo. Dessa forma, a faixa de valores válidos corresponderá a $[\text{min}, \infty + [$. Caso seja utilizado apenas o valor máximo a faixa de valores válidos corresponderá a $]-\infty, \text{max}]$.

A Figura 5.11 complementa o exemplo da Figura 5.9. Nele os estados físicos estão agrupados pela tag `<physical-states>`.

```
<physical-states>
+ <physical-state name="energia" value="6" min="-5" max="10">
+ <physical-state name="sede" value="random" min="2" max="7">
  <!-- continua -->
</physical-states>
```

Figura 5.11 – Exemplo de descrição de objetos da classe *PhysicalState* em XML

Atitudes e Emoções: Tags *<attitudes>* e *<emotions>*

As tags *<attitudes>* e *<emotions>* agrupam, respectivamente, a definição das atitudes (classe *Attitude*) e as emoções (classe *Emotion*) do agente. Cada humor e estado físico são descritos em XML, respectivamente pelas tags *<attitude>* e *<emotion>*.

Conforme apresentado no capítulo anterior, as classes *Attitude* e *Emotion* descendem de *AbstractElementWithObjects*. Esta classe permite o uso de um conjunto de entidades abstratas ou concretas (seção 4.3.4). Na Figura 5.12 apresentamos em XML o exemplo da Figura 4.16. Neste exemplo, temos a definição de uma entidade concreta (macarrão). As entidades são definidas pela tag *<entity>*, sendo agrupadas na tag *<entities>*.

```
- <attitudes>
- <attitude name="gostar" value="0" min="-5" max="5">
  - <states>
    <state name="não gosta" min="-5" max="0" />
    <state name="gosta" min="1" max="5" />
  </states>
- <entities>
  - <entity name="macarrão" type="comida" value="1">
    + <events>
    + <behavior>
  </entity>
</entities>
+ <events>
+ <behavior>
</attitude>
<!-- continua -->
</attitudes>
```

Figura 5.12 – Exemplo apresentado na Figura 4.16 descrito em XML

Na Figura 5.13 a descrição em XML do exemplo apresentado pela Figura 4.17. Neste exemplo, que é uma extensão do exemplo anterior, destacamos como é feita a definição de uma entidade concreta.

```

- <attitude name="gostar" value="0" min="-5" max="5">
+ <states>
- <entities>
+ <entity type="cachorro" value="5">
- <entity name="macarrão" type="comida" value="1">
+ <events>
+ <behavior>
</entity>
</entities>
+ <events>
+ <behavior>
</attitude>

```

Figura 5.13 – Exemplo apresentado na Figura 4.17 descrito em XML

Para finalizar a discussão sobre atitudes e emoções, apresentamos na Figura 5.14 um exemplo de definição de emoção. Neste exemplo, o *botmaster* especifica que o valor da emoção deve ser definido no momento em que ela for utilizada, pois este valor não foi especificado. Também é definida uma restrição ao agente através da entidade concreta *mãe*, cujo valor é “0”. Este valor significa que o agente não pode sentir esta emoção, uma vez que este valor representa o limiar que indica o fim da emoção.

```

- <emotions>
- <emotion name="raiva" min="0" max="10" timedelay="5" valuedelay="-1">
- <states>
<state name="fraco" min="1" max="3" />
<state name="medio" min="3" max="5" />
<state name="forte" min="6" />
</states>
- <entities>
- <entity name="mãe" type="pessoa" value="0">
+ <events>
+ <behavior>
</entity>
</entities>
+ <events>
+ <behavior>
</emotion>
<!-- continua -->
</emotions>

```

Figura 5.14 – Exemplo de emoção descrito em XML

Traços de Personalidade: *Tag <trait>*

A *tag <traits>* agrupa a definição dos traços de personalidade (classe *Trait*) do agente. Cada traço é descrito em XML pela *tag <trait>*. A Figura 5.15 apresenta a representação em XML do exemplo definido pela Figura 4.13.

```
- <traits>
- <trait name="unfriendly" value="expression" min="0" max="90">
- <expression>
  (<evaluate element="mood" name="distress" get="value" /> / 3)
  +
  (<evaluate element="mood" name="anger" get="value" /> / 2)
</expression>
+ <states>
+ <events>
+ <behavior>
</trait>
<!-- continua -->
</traits>
```

Figura 5.15 – Exemplo da Figura 4.13 descrito em XML

No exemplo acima apresentamos como um valor da classe *ExpressionValue* deve ser definido. Para isso, utiliza-se a *string* “expression” no atributo *value* e acrescentamos na definição do elemento da personalidade a *tag* *<expression>*, que define a expressão utilizada. A *tag* *<evaluate>*, que será apresentada em detalhes na próxima seção, é utilizada para avaliar o conteúdo de um elemento da personalidade.

5.4 Outras *Tags* Adicionadas

Além das *tags* apresentadas nas seções anteriores, um novo conjunto foi adicionado à linguagem AIML para permitir a implementação da arquitetura Persona-AIML. Estas *tags* são: *<evaluate>*, *<change>*, *<attitude>*, *<emotion>*, *<switch>*, *<call-action>*, *<find-data>*, *<block-do>*, *<modify-change>*, *<if>* e *<eval>*.

A seguir detalharemos o funcionamento destas novas *tags*.

Tag* *<evaluate>

A *tag* *<evaluate>* é utilizada para consultar os atributos dos elementos de personalidade. Essa *tag* possui os seguintes atributos: *element*, *name* e *get*. O atributo *element*, que indica qual o elemento da personalidade está sendo consultado, pode assumir os seguintes valores: *trait*, para traços de personalidade; *physical-state*, para estados físicos; *mood*, para humores; *attitude*, para atitudes; e *emotion* para emoções. O atributo *name* indica o nome do elemento da personalidade. O atributo *get*, que indica qual atributo do elemento da personalidade está sendo consultado, pode assumir os seguintes valores: *value*, para a intensidade; *min*, para o valor mínimo da intensidade,

max, para o valor máximo da intensidade; e *state*, para consultar o estado atual (Figura 5.16).

```
- <category>
  <pattern>COMO VAI VOCE</pattern>
- <template>
  + <pattern-type>
  - <do what="talk">
    Eu estou <evaluate element="mood" name="felicidade" get="state" />
    </do>
  </template>
</category>
```

Figura 5.16 – Exemplo de uso da tag *<evaluate>*

Caso uma atitude ou emoção seja consultada, é preciso especificar qual entidade concreta esta sendo verificada. Para isso é adicionada a tag *<entity>*. Esta tag é formada pelas sub-tags *<name>* e *<type>*, que é opcional. Essas *sub-tags* representam, respectivamente, o nome e o tipo da entidade (Figura 5.17).

```
- <category>
  <pattern>VOCE GOSTA DE COMER *</pattern>
- <template>
  + <pattern-type>
  - <do what="talk">
    Eu
    - <evaluate element="attitude" name="gostar" get="state">
      - <entity>
        <name><star /></name>
        <type>comida</type>
      </entity>
    </evaluate>
    de comer <star /> .
  </do>
</template>
</category>
```

Figura 5.17 – Exemplo de uso da tag *<evaluate>*

Tag <change>

A tag *<change>* é utilizada para modificar os atributos de um elemento da personalidade e só pode ser utilizada pelo componente de personalidade. Essa tag possui os atributos *element*, *name* e *set*; e a sub-tag *<value>* que indica o valor utilizado na modifica. Os atributos *element* e *name* são equivalentes aos utilizados pela tag *<set>*. De acordo com a modificação a ser realizada, o atributo *set* pode assumir os seguintes valores: *value*, para a intensidade; *min*, para o valor mínimo da intensidade,

max, para o valor máximo da intensidade; *add*, para adicionar o valor definido pela *tag* `<value>` a intensidade do elemento da personalidade; *sub*, para subtrair o valor definido pela *tag* `<value>`; e *state*, para modificar a intensidade a partir de um estado dos estados válidos. Neste último caso, será utilizado o valor mínimo do estado.

Assim como na *tag* `<evaluate>`, quando atitudes ou emoções são modificadas o *botmaster* precisa especificar a entidade através da sub-*tag* `<entity>`.

A Figura 5.18 apresenta um exemplo de uma regra válida para todas as entidades da atitude *gostar*, definida na Figura 5.12. As *tags* `<get-name>` e `<get-type>` são utilizadas para referenciar a entidade que está sendo afetada por uma regra definida em uma atitude ou emoção.

```
- <attitude name="gostar" value="0" min="-5" max="5">
+ <states>
+ <entities>
+ <events>
- <behavior>
- <state>
  - <rule name="naogosta">
    + <condition>
    - <action>
      - <change element="attitude" name="gostar" set="state">
        - <entity>
          - <name> <get-name /> </name>
          - <type> <get-type /> </type>
        </entity>
        <value>não gosta</value>
      </change>
    </action>
  </rule>
</state>
</behavior>
</attitude>
```

Figura 5.18 – Exemplo de uso da *tag* `<change>`

***Tag* `<attitude>`**

As *tag* `<attitude>` é utilizada para adicionar e remover uma entidade concreta à qual as atitudes podem se referenciar. A *tag* `<attitude>` apresenta os atributos *for* e *name*, e a sub-*tag* `<entity>`.

O atributo *for* pode assumir os seguintes valores: *add* e *remove*. O valor *add* indica a adição. O valor *remove*, por sua vez indica remoção.

O atributo *name* indica o nome da atitude a ser adicionada. Esta atitude deve ter sido previamente definida no componente de personalidade.

A tag `<entity>` indica a entidade a ser adicionada. Ela é composta pelas seguintes sub-tags: `<name>`, que indica o nome da entidade; `<type>` que indica o tipo da entidade; `<min>` que indica o valor mínimo; `<max>`, que indica o valor máximo; `<timedelay>`, que indica o intervalo de tempo para o contar; `<valuedelay>`, que indica o valor que deve ser adicionado/subtraído a cada intervalo do contador; `<value>`, que indica a intensidade; e `<state>`, que pode ser utilizada ao invés da tag `<value>` para determinar a intensidade a partir de um estado válido. As tags `<value>`, `<state>`, `<min>`, `<max>`, `<timedelay>` e `<valuedelay>` são utilizadas apenas na adição de uma atitude, desde que seus valores não tenham sido pré-definidos no componente de personalidade através de: atitude correspondente; restrição definida por uma entidade da atitude. Caso isso ocorra, será utilizado o valor pré-definido. A Figura 5.19 apresenta um exemplo de adição de uma atitude por uma categoria do tipo “*chamada de procedimento*”.

```
- <category>
  <pattern>APRESENTAR NOME * TIPO *</pattern>
- <template>
  - <do what="process">
    - <attitude name="gostar" for="add">
      - <entity>
        - <name><star index="1" /> </name>
        - <type><star index="2" /> </type>
      </entity>
    </attitude>
  </do>
</template>
</category>
```

Figura 5.19 – Exemplo de uso da tag `<attitude>`

Tag <emotion>

As tag `<emotion>` é utilizada para adicionar e remover uma entidade concreta à qual as emoções podem se referenciam. Sua sintaxe é a mesma da tag `<attitude>`. A Figura 5.20 apresenta um exemplo de uma regra válida para todas as entidades da atitude *gostar*, definida na Figura 5.12.


```

- <attitude name="gostar" value="0" min="-5" max="5">
+ <states>
+ <entities>
+ <events>
- <behavior>
+ <state>
- <decision>
- <rule name="fica com raiva">
+ <condition>
- <action>
- <emotion name="raiva" for="add">
- <entity>
- <name> <get-name /> </name>
- <type> <get-type /> </type>
<state>forte</state>
</entity>
</emotion>
</action>
</rule>
</decision>
</behavior>
</attitude>

```

Figura 5.20 – Exemplo de uso da tag <emotion>

Tag <switch>

As tag <switch> tem a mesma funcionalidade da tag <condition> quando esta se apresenta na forma de uma instrução do tipo “switch”. Porém, ao invés de consultar variáveis simples, esta tag é utilizada apenas para elementos de personalidade. Assim com na tag <condition>, cada caso é representado por uma tag , onde o atributo *value* corresponde ao valor do atributo do elemento da personalidade consultado, caso este atributo seja omitido, significa que a tag é o caso *default*.

Os elementos de personalidade são consultados de maneira semelhante a tag <evaluate>. Logo, a tag <switch> apresenta os atributos *element*, *name* e *get*. Se o elemento consultado for uma atitude ou emoção, a entidade deve ser especificada através da sub-tag *entity*.

A Figura 5.21 apresenta um exemplo de uso da tag <switch> na definição de uma ação.

```
- <actions>
- <action name="saudação">
- <switch element="attitude" name="gostar" get="state">
- <entity>
- <name><get-param index="1" /> </name>
- <type><get-param index="2" /> </type>
</entity>
<li value="gosta">Oi, que bom lhe ver.</li>
<li>Eita... Chegou quem eu não queria.</li>
</switch>
</action>
</actions>
```

Figura 5.21 – Exemplo de uso da tag `<switch>`

Tag <call-action>

A tag `<call-action>` é utilizada para invocar uma ação definida pelo componente de personalidade. Esta tag possui o atributo *name* que indica o nome da ação. Os parâmetros da ação são passados pela sub-tags `<param>`, cujo atributo *index* é utilizado para indexar os atributos a serem passados. A Figura 5.22 apresenta um exemplo de uso desta tag com base na ação definida na Figura 5.21.

```
- <attitude name="gostar" value="0" min="-5" max="5">
+ <states>
+ <entities>
- <events>
- <on-create>
- <rule name="saudação">
- <action>
- <call-action name="saudação">
- <param index="1"><get-name /> </param>
- <param index="2"><get-type /> </param>
</call-action>
</action>
</rule>
</on-create>
+ <on-change>
</events>
+ <behavior>
</attitude>
```

Figura 5.22 – Exemplo de uso da tag `<call-action>`

Tag <modify-change>

A tag `<modify-change>` altera globalmente as tags `<change>`. A partir dessa tag, o *botmaster* pode definir um padrão para a modificação dos elementos da personalidade. Por exemplo, o modelo de personalidade proposto por SILVA (2000)

define que os elementos da personalidade são modificados a partir de valores constantes. A Figura 5.23 apresenta um exemplo de uso da tag `<modify-change>` que pode ser utilizado com este objetivo.

```
- <modify-change element="attitude" name="gostar" set="sub">
- <value> 1 + <get-value /> </value>
</modify-change>
```

Figura 5.23 – Exemplo de uso da tag `<modify-change>`

Tag <block-do>

A tag `<block-do>` é utilizada para determinar se o padrão de saída definido na base de categorias pode ser retornado para o usuário. Esta tag possui o atributo `set` que assume os seguintes valores: `on`, caso a resposta deva ser utilizada; e `off`, caso contrário.

A Figura 5.24 apresenta um exemplo de regra que poderia ser adicionada à emoção raiva definida na Figura 5.14. No exemplo, o `chatterbot` deixa de responder a entidade caso esteja com muita raiva dela.

```
- <rule name="deixa de responder" shot="once">
- <condition>
- <switch element="emotion" name="raiva" get="state">
- <entity>
- <name> <get-name /> </name>
- <type> <get-type /> </type>
</entity>
- <li value="forte"> <do-action /> </li>
</switch>
</condition>
- <action> <block-do set="off" /> </action>
</rule>
```

Figura 5.24 – Exemplo de uso da tag `<block-do>`

Tag <find-data>

A tag `<find-data>` localizar na memória do `chatterbot` os dados definidos pela tag `<pattern-data>` (seção 5.2). Assim sendo, esta tag também apresenta as sub-tags `<name>`, `<type>` e `<source>`; utilizadas como parâmetros de busca. Ao efetuar a tag `<find-data>` irá localizar o primeiro dado que satisfaça aos parâmetros utilizados. Por exemplo, ao definir apenas o parâmetro em `<type>` uma busca, esta tentará localizar o primeiro dado cujo tipo corresponda aquele passado como parâmetro.

O resultado de uma busca pode ser consultado através da variável *data_found* e os atributos *name*, *type* e *source*, podem ser consultados através da tag `<get-attribute>`.

A Figura 5.25 apresenta um exemplo de uso da tag `<find-data>`. Neste exemplo, o humor *felicidade* muda para o estado feliz quando o *chatterbot* recebe um presente.

```
- <rule name="receber presente">
- <condition>
- <find-data>
  <type>presente</type>
- <true><run-action /></true>
</find-data>
</condition>
- <action>
- <change element="mood" name="felicidade" set="state">
  <value>feliz</value>
</change>
+ <pattern-data for="remove">
</action>
</rule>
```

Figura 5.25 – Exemplo de uso da tag `<find-data>`

Tag `<if>`

Uma vez que a tag `<condition>` só permite comparações entre uma variável e constantes criamos a tag `<if>`. Esta tag permite avaliar expressões formadas por dois operandos e um operador comparativo. Os operandos são definidos pelas tags `<left>` e `<right>`, enquanto o operador de comparação é definido pelo atributo *operator*. O atributo *type* indica se os operandos são numéricos (valor *number*) ou cadeias de caracteres (valor *string*).

Se a condição for satisfeita, serão executadas as instruções definidas pela tag `<then>` e, caso contrário, as instruções da tag `<else>`, que é opcional.

De acordo com o operador utilizado, o atributo *operator* pode assumir os seguintes valores: *e* (`<left>` “igual a” `<right>`), *ne* (`<left>` “diferente de” `<right>`), *l* (`<left>` “menor que” `<right>`), *le* (`<left>` “menor ou igual a” `<right>`), *g* (`<left>` “maior que” `<right>`), *ge* (`<left>` “maior ou igual a” `<right>`).

A Figura 5.26 apresenta um exemplo de uso da tag `<if>`. No exemplo a tag é utilizado para comparar se dois números, passados como parâmetros para a categoria, são iguais.

```
- <category>
  <pattern>O numero * e igual a *</pattern>
- <template>
  - <do what="process">
    - <if type="number" operator="e">
      - <left> <star index="1" /> </left>
      - <right> <star index="2" /> </right>
      <then> são iguais</then>
      <else> não são iguais</else>
    </if>
  </do>
</template>
</category>
```

Figura 5.26 – Exemplo de uso da tag `<if>`

Tag <eval>

Com o objetivo de permitir o uso de expressões numéricas definimos a tag `<eval>`. Esta tag possui o atributo `type` que define se o resultado da expressão deve ser retornado como um inteiro (valor *integer*) ou como ponto-flutuante (valor *float*). A Figura 5.27 apresenta um exemplo de uso dessa tag. No exemplo a tag é utilizada para fazer uma soma.

```
- <category>
  <pattern>Quanto é * mais *</pattern>
- <template>
  + <pattern-type>
  - <do what="talk">
    O resultado é
    - <eval type="float">
      <star index="1" /> + <star index="2" />
    </eval>
  </do>
</template>
</category>
```

Figura 5.27 – Exemplo de uso da tag `<eval>`

5.5 Conversor Persona-AIML para AIML Padrão

Para que a arquitetura pudesse ser executada utilizando um interpretador AIML padrão, foi desenvolvido um conversor (Figura 5.28) que, a partir do componente de personalidade e da base de categorias, gera um conjunto de arquivos AIML padrão.

Essa conversão foi possível uma vez que, conforme visto na seção 2.5, as categorias AIML podem ser utilizadas para definir procedimentos.

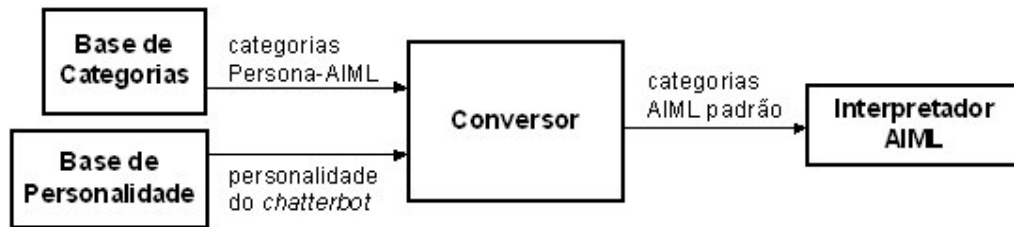


Figura 5.28 – Funcionamento do conversor Persona-AIML

O conversor foi implementado em Java e também foi necessário a utilização da linguagem de *script* JavaScript para que alguns comandos AIML fossem convertidos em *tags* `<javascript>`. Essa *tag* é suportada pela maioria dos interpretadores AIML.

As categorias AIML geradas pelo conversor foram testadas nos interpretadores *ProgramD* (escrito em Java) e *J-Alice* (escrito em C++) com sucesso. Isso demonstra

5.6 Protótipo Testado

Para validar a arquitetura Persona-AIML, decidimos desenvolver um protótipo de *chatterbot* implementado segundo a especificação dessa arquitetura. A implementação do protótipo seguiu os passos apresentados na seção 4.4.

Primeiramente foi definida a base de categorias. Para acelerar este passo, adaptamos a base de categorias AIML do *chatterbot* Pixel³.

Na construção do componente de personalidade, adaptamos o Modelo dos Cinco Grandes Fatores (seção 3.2.2) como o mecanismo de validação da personalidade do agente (seção 4.3.5).

Em seguida, comportamentos foram definidos a partir de cenários comuns em diálogos com o Pixel. Os elementos da personalidade do robô foram escolhidos com base nesses cenários.

³ <http://www.virtus.ufpe.br/pixelbot>

Os testes foram realizados com usuários de salas de bate papo e através de um website.

Por fim, a análise dos resultados foi feita tendo como métrica a reação que se esperava que o robô tivesse com o comportamento modelado.

A seguir detalhamos cada um desses passos e, ao final apresentamos uma análise dos resultados obtidos.

5.6.1 Construção da Base de Categorias

Conforme visto na o Capítulo 2, um dos problemas referentes a tecnologia de *chatterbots* é aquisição da base de categorias, responsável pela identificação da sentença do usuário. Uma vez que sua aquisição demanda tempo, decidimos adaptar a base de algum *chatterbot* existente.

Sendo assim, para acelerar o desenvolvimento de nosso protótipo, decidimos criar nossa base a partir de uma adaptação da base do *chatterbot* Pixel, que está sendo desenvolvido no VIRTUS (Laboratório de Hiperídia da UFPE). Uma das funções de Pixel é ajudar os usuários tirando dúvidas. Para isso ele efetua consultas em engenhos de busca, retornando seus resultados para o usuário.

Inicialmente, foi necessária a análise da base de Pixel, a fim de identificar as informações referentes a cada categoria. Nesta análise definimos o tipo de padrão de entrada das categorias em que o usuário xingava ou elogiava o robô. Além disso, marcamos cada categoria definindo através do tipo de padrão de entrada se o robô havia entendido o usuário.

5.6.2 Definição do Modelo de Personalidade

A maneira como os elementos de personalidade interagem entre si (mecanismo de validação) foi definida conforme a Teoria dos Traços (da qual o *Big Five* deriva), na qual os traços devem ser estáveis e os demais elementos de personalidade devem ser influenciados pela intensidade de um traço. Para isso, cada instrução que modifica a intensidade de um elemento da personalidade deve ter o seu valor alterado com base na intensidade do traço. Neste modelo de personalidade não fizemos nenhuma restrição quanto aos elementos de personalidade que poderiam ser adotados.

O modelo foi implementado em Java na classe *BigFiveModel*. Esta classe, que atende a especificação apresentada na seção 4.3.5, valida a personalidade do agente através da consulta as suas regras e elementos. Sendo assim, caso exista alguma instrução que modifique um traço de personalidade, será reportado ao *botmaster* um erro. Também é verificado se os traços definidos não apresentam o valor da intensidade definido a partir de uma expressão. Por fim, é verificado se cada elemento da personalidade define uma regra no evento *on-create* adicionando os comandos *<modify-change>* (seção 5.4) para o incremento e decremento de sua intensidade, com base no traço.

5.6.3 Definição dos Comportamentos e Elementos de Personalidade

Para facilitar a avaliação de nosso protótipo, decidimos modelar os cenários que ocorriam freqüentemente em diálogos como Pixel. A escolha desses cenários foi realizada através da análise de vários *logs* de diálogos.

Os comportamentos modelados foram os seguintes: (1) o *chatbot* só faria consultas na Internet para as dúvidas do usuário caso *gostasse dele*; (2) o *chatbot* só responderia a questões de ordem pessoal caso *estivesse feliz*.

A partir desses comportamentos foram escolhidos os elementos e as regras. Os elementos escolhidos foram: atitude *gostar* e humor *felicidade*. Por sua vez, foram definidas as seguintes regras: (a) sempre que o usuário agredir o robô, a intensidade da atitude *gostar* deveria ser diminuída e se ele pedir desculpas a intensidade aumenta; (b) o humor *felicidade* aumentaria sempre que o robô entendesse a sentença do usuário, e diminuiria caso contrário.

Com base no modelo *Big Five*, esses elementos foram associados ao traço *nível de socialização (agreeableness)*. Assim, no primeiro cenário, quanto maior a intensidade desse traço, maior será o incremento da atitude *gostar* caso o usuário peça desculpas, e menor será o decremento dessa atitude caso o usuário agrida o robô. No outro cenário, quanto maior o nível de socialização, maior será o decremento do humor *felicidade* caso o robô não entenda a sua sentença, e menor será o incremento caso contrário.

5.6.4 Testes

O protótipo foi testado com usuários de salas de IRC (*Internet Relay Chat*) e através de um *website* que foi acessado por alunos da UFPE e visitantes da página do Laboratório Virtus. Nos casos de IRC, o *chatterbot* freqüentava canais aleatórios da rede Brasnet⁴. A Figura 5.29 apresenta a tela de interface *web* por onde o usuário se comunica com o *chatterbot*.

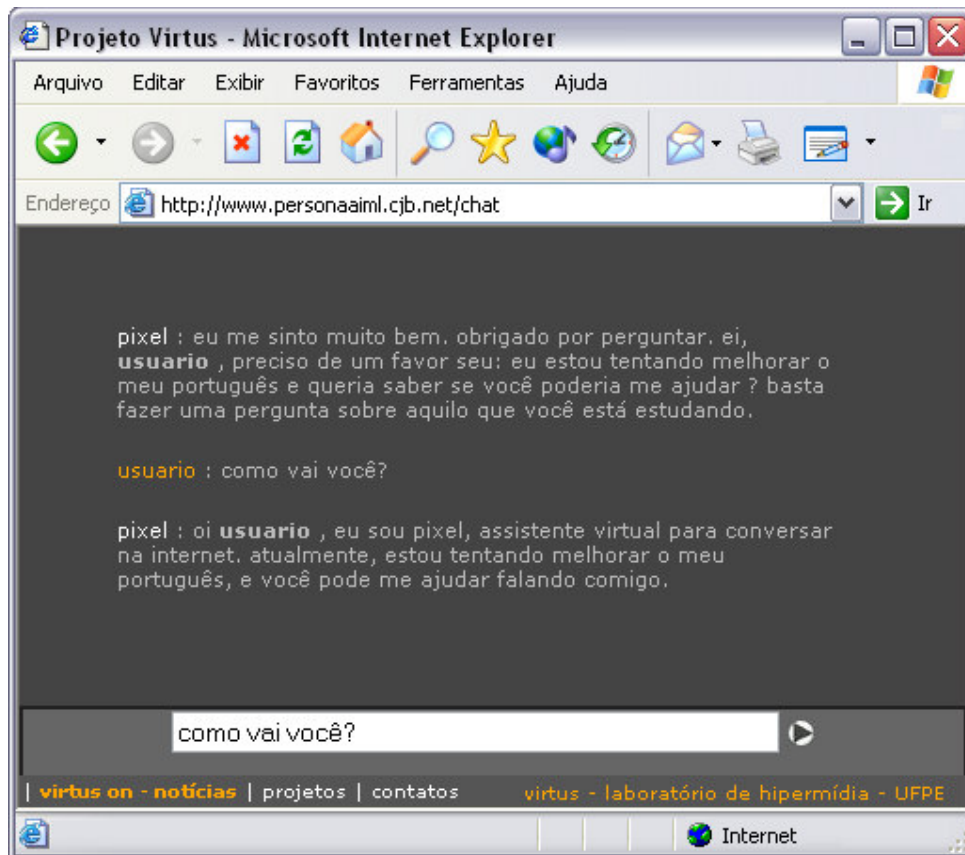


Figura 5.29 – Interface *web* para diálogo com o *chatterbot*

5.6.5 Análise dos Resultados

A partir dos *logs* obtidos foi feita uma análise dos *logs* de diálogos entre o robô e os usuários. Com base nessa análise qualitativa, concluímos que o *chatterbot* foi capaz de reagir coerentemente com a especificação dos comportamentos definidos.

Para ilustrar um exemplo desta análise apresentamos na Figura 5.30 um *log* de diálogo com o *chatterbot* protótipo (Apêndice A). No trecho, o usuário agradece o robô

⁴ <http://www.brasnet.org>

(linha 3), disparando uma regra que faz com que o *chatbot* deixe de gostar do usuário. Com base na regra acima, ele não responde sua dúvida (linha 6). Outras regras (Apêndice B) definem que o *chatbot* só voltará a gostar do usuário caso ele peça desculpas, o que ocorre na linha 7.

		...
1	usuário	O que você sabe?
2	ROBÔ	PODE-SE DIZER QUE É VOCÊ QUEM ESTÁ ME ENSINANDO, USUÁRIO [...]
3	usuário	Então você é burro.
4	ROBÔ	AH É? NAO TE AJUDAREI MAIS ATÉ ME PEDIR DESCULPAS.
		...
5	usuário	Queria saber sobre carros.
6	ROBÔ	NÃO RESPONDO ATÉ VOCÊ ME PEDIR DESCULPAS.
7	usuário	Desculpa.
8	ROBÔ	DEPOIS QUE PISA NA BOLA É FÁCIL PEDIR DESCULPAS.
9	usuário	O que você sabe sobre carros?
10	ROBÔ	TEM ALGUNS SITES INTERESSANTES SOBRE ISSO VEJA AQUI.

Figura 5.30 – Exemplo de diálogo com o *chatbot*

Isso nos motiva a crer que, de fato, a Persona-AIML pode ser utilizada para criar *chatbots* com personalidade. No entanto, entendemos que é necessária a realização de testes em cenários mais complexos e variados, a fim de verificar a flexibilidade da arquitetura.

Uma observação importante é que os usuários da web por saberem previamente que se tratava de um protótipo para teste utilizaram uma linguagem mais formal, facilitando o diálogo com o robô que apresentou uma maior precisão na identificação da sentença do usuário. Por outro lado, uma grande parcela dos usuários de IRC além de utilizarem uma linguagem informal com muitos erros de ortografia, ao perceberem que estavam lidando com uma máquina deixavam de dialogar.

5.7 Conclusões

Neste capítulo apresentamos a implementação da Persona-AIML, bem como o desenvolvimento de um protótipo de *chatbot* com personalidade modelada a partir da arquitetura proposta.

A implementação foi feita objetivando a reutilização dos interpretadores AIML existentes, o que facilitou o processo de desenvolvimento. Para isso, criamos um conversor que transforma as *tags* definidas em nossa arquitetura para *tags* AIML

padrão. Além disso, desenvolvemos uma *parser* para que o *botmaster* especificasse a personalidade do agente em XML, cuja sintaxe já é conhecida por ele. No entanto, tendo em vista a simplificação da implementação, o mecanismo de validação do modelo de personalidade não foi definido em XML, tendo que ser feito em uma classe na linguagem Java.

O fato do *chatterbot* utilizado no protótipo apenas poder se expressar através de texto é um fator limitante para uma análise mais completa da arquitetura proposta. Ações mais complexas poderiam ser testadas caso o *chatterbot* estivesse inserido em um ambiente onde ele pudesse interagir com diversas entidades ao mesmo tempo e possuísse características como mobilidade e personificação, através de um avatar.

6

Conclusão

Esta dissertação apresentou a arquitetura Persona-AIML para a criação de *chatbots* em AIML com personalidade. O objetivo principal foi preencher uma lacuna nessa área de pesquisa, provendo uma arquitetura flexível e que permite a criação de modelos robustos de personalidade. Testes preliminares com o protótipo revelaram a eficácia da arquitetura proposta.

Veremos a seguir as principais contribuições deste trabalho, bem como indicações de trabalhos futuros.

6.1 Contribuições

A principal contribuição deste trabalho foi, sem dúvida, o desenvolvimento da Persona-AIML, que seguiu princípios de Qualidade de Software (modularidade, extensibilidade e reusabilidade), adotou o Paradigma de Agentes Inteligentes (consagrado dentro da Inteligência Artificial simbólica). O protótipo utilizou um modelo de personalidade baseado no *Big Five* (também consagrada na Psicologia), permitindo, contudo, que outros modelos de personalidade sejam utilizados na criação de novos *chatbots*.

A modelagem dessa arquitetura, em UML, acompanha os princípios de Engenharia de Software, mencionados acima. Essa modelagem foi desenvolvida com o intuito de poder ser reutilizada ou estendida para outros modelos de personalidade.

A implementação do protótipo acompanhou as tendências atuais de *chatbots* na Web, utilizando a linguagem AIML para seu desenvolvimento. A base de categorias utilizada foi adaptada da base do *chatbot* Pixel, que foi inicialmente gerada a partir da base de categorias AIML em inglês disponível no *site* de ALICE. Como AIML não fornece mecanismos para a modelagem de personalidade, foi necessário modificar a sua base de categorias e desenvolver um conversor da nossa extensão (Persona-AIML) para AIML padrão.

Foi necessário modificar os *templates* de saída nas categorias existentes, que seriam responsáveis por "traduzir" em sentenças o comportamento do robô (de acordo com sua personalidade e como estado do mundo atual). Esta foi, possivelmente, a maior dificuldade enfrentada neste trabalho, uma vez que não é simples traduzir em palavras estados de humor e comportamentos (sem contar com a ajuda de gestos e sons). Além disso, criamos também novas *tags* AIML, a fim de processar as regras no componente de personalidade. Essas regras são utilizadas, entre outras coisas, na escolha do *template* de saída para formação da sentença do robô. Apesar de estar baseada na teoria do *Big Five*, essa base pode ser reutilizada para a construção de robôs com outros modelos de personalidade, uma vez que a personalidade é modelada em um componente separado (componente de personalidade).

O conversor desenvolvido permitiu que a arquitetura pudesse ser executada em interpretadores padrões de AIML. Dessa forma, a comunidade AIML não precisa reescrever os interpretadores existentes a fim de modelar personalidade em seus *chatbots*.

Um dos princípios da nossa arquitetura é a separação entre a base de categorias, responsável pela identificação da sentença do usuário, e a personalidade do *chatbot*. Portanto, o *botmaster* poderá reutilizar a base de categorias na construção de outros *chatbots*.

Concluindo, podemos dizer que o nosso trabalho contribuiu para a área com uma arquitetura e um protótipo com qualidade de concepção, modelagem e implementação, que irão certamente preencher uma lacuna existente: a criação de *chatbots* em AIML com personalidade robusta e coerente.

6.2 Trabalhos Futuros

Este trabalho deixa espaço para diversas extensões, algumas delas não realizadas ainda por limitação de tempo. Entre elas, destacamos:

1. A realização de testes com usuários reais em maior quantidade, a fim de verificar o que deve ser melhorado no sistema como um todo;
2. A criação de um *chatbot* baseado em outro modelo de personalidade, a fim de comprovar generalidade da Persona-AIML;

3. O desenvolvimento de um editor de personalidade, a fim de facilitar o desenvolvimento da personalidade do *chatbot*;
4. A criação de um repositório de elementos de personalidade para que o *botmaster* a personalidade do *chatbot* a partir de elementos pré-definidos;
5. O desenvolvimento de um *chatbot* para ambientes que envolvam outras entidades além do usuário (e.g., um jogo de RPG). Nesse trabalho, poderia ser explorada a personificação do *chatbot*;
6. A utilização da Persona-AIML para a criação de um modelo de personalidade utilizando o modelo OCC de emoções. Esse trabalho é particularmente desafiador, uma vez que o modelo OCC não define como outros elementos de personalidade interferem nas emoções;
7. Um estudo a fim de reproduzir os experimentos de MOON (1998). Por exemplo, poderia ser desenvolvido um *chatbot* vendedor para verificar a possibilidade de induzir o usuário a comprar determinados produtos.

Referências Bibliográficas

- ABBATTISTA, F.; DEGEMMIS, M.; LICCHELLI, O.; LOPS, P.; SEMERARO, G.; ZAMBETTA, F. *Improving the Usability of an E-Commerce Web Site Through Personalization*. In: Proceedings of the Workshop on Recommendation and Personalization in Ecommerce, 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems. Malaga, Spain, May 2002.
- ALLPORT, G. *Personality*. Nova York: Holt, 1937.
- ALLPORT, G. *Traits Revisited*. *American Psychologist*, 21: 1-10, 1966.
- ALLPORT, G.; ODBERT, H. *Trait-Names: A Psycho-Lexical Study*. *Psychological Monographs*, vol 47 No. 211, 1936.
- ANDERSON, C. *A Survey of Electronic Commerce – In Search of the Perfect Market*. *The Economist*, 1997. Publicação eletrônica disponível em: <http://www.economist.com/surveys/showsurvey.cfm?issue=19970510> . Acesso em 20 jan. 2002.
- ANDRÉ, E.; KLESEN, M.; GEBHARD, P.; ALLEN, S.; RIST, T. *Integrating Models of Personality and Emotions Into Lifelike Characters*. In Proceedings International Workshop on Affect in Interactions. Towards a New Generation of Interfaces, 1999.
- BALLARD, B.; JONES, M. *Computational Linguistic*. In: Shapiro, S. C. *Encyclopedia of Artificial Intelligence*. [s.l.]: Wiley-Interscience Publication, 1987.
- BANDURA, A. *Social Learning Theory*. Englewood Cliffs, Prentice-Hall, 1977.
- BARTNECK, C. *Integrating the OCC Model of Emotions in Embodied Characters*. In: Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges, Melbourne, 2002.
- BATES, J. *The Role of Emotion in Believable Agents*. *Communications of the ACM*, 37, 122-125, 1994.

BAZZAN, A.; ADAMATTI, D.; BORDINI, R.; GHILHARDI, T. *Extending the Computational Study of Social Norms with the Use a Systematic Model of Emotions*. In: Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence: Proceedings (LNAI). Berlin: Springer-Verlag, 2002. p. 108-117.

BICKMORE, T. W. *Social Intelligence in Conversational Computer Agents*. Proseminar Conceptual Analysis of Thesis Area. Gesture & Narrative Language Group. Mit Media Laboratory. December 1999.

BICKMORE, T.; CASSELL, J. *Relational Agents: A Model and Implementation of Building User Trust*. ACM CHI 2001 Conference Proceedings, Seattle, Washington, 2001.

BRAGA, A.; LUDERMIR, T.; CARVALHO, A. *Redes Neurais Artificiais - Teoria e Aplicações*, LTC, 1999.

BRAVE, S; NASS, C. *Emotion in Human-Computer Interaction*. In J. Jacko & A. Sears (Eds.), *Handbook of Human-Computer Interaction*. New York: Lawrence Erlbaum Associates. 2002. p.251-271.

CASSELL, J.; BICKMORE, T.; BILLINGHURST, M.; CAMPBELL, L.; CHANG, K.; VILHJÁLMSSON, H.; YAN, H. *Embodiment in Conversational Interfaces: Rea*. CHI99, Pittsburgh, PA, 1999.

COCA-COLA. *Hank, The Coca-Cola Virtual Representator*. Publicação eletrônica disponível em: <http://questions.coca-cola.com>. Acesso em 20 dez. 2002.

COLBY, K.; WEBER, S.; HILF, F. *Artificial Paranoia*. *Artificial Intelligence* 2(1): 1-25, 1971.

CURTIS, P. *Mudding: Social Phenomena in Text-Based Virtual Realities*. Proceedings of the 1992 Conference on Directions and Implications of Advanced Computing. Berkeley, May 1992.

CYBERMECHA. Disponível em: <http://www.cybermecha.com>. Acesso em 30 set. 2002

DIGMAN, J. *Personality Structure: Emergence of the Five-Factor Model*. *Annual Review of Psychology*, 41, 417-440, 1990.

- DONATH, J. *Being Real*. In: *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet*. Cambridge, MA: MIT Press 2000.
- DONEY, P.; CANNON, J. *An Examination of the Nature of Trust in Buyer-Seller Relationships*. *Journal of Marketing*, 61, 35-51. 1997
- ELLIOT, C. *The Affective Reasoner: A Process Model of Emotions in a Multi-agent System*. Ph.D. Thesis. Technical Report No. 32, Institute for the Learning Sciences, Northwestern University. Evanston, IL. May 1992.
- ELLIOT, C.; BRZEZINSKI, J. *Autonomous Agents as Synthetic Characters*. *AI Magazine*, 19(2), 1998.
- FIGUEIRA, C. *JEOPS: Java Embedded Object Production System*. Dissertação de Mestrado. Universidade Federal de Pernambuco, Recife, 2000.
- FONER, L. *Entertaining Agents: A Sociological Case Study*. Proceedings of the First International Conference on Autonomous Agents, ACM Press, 1997.
- FORBES. *Invasion of the Virbots*, September 18, 2000. Publicação eletrônica disponível em: http://www.forbes.com/global/2000/0918/0318118a_print.html. Acesso em 07 set. 2002.
- FOWLER, M; SCOTT, K. *UML Essencial: Um breve guia para a linguagem padrão de modelagem de Objetos*. Trad. Vera Pezerico e Christian Thomas Price. 2ª edição, Porto Alegre. Bookman, 2000.
- FRANK, A.; STERN, A; RESNER, B. *Socially Intelligent Virtual Petz*. In *Socially Intelligent Agents*, p. 43-45. AAAI Press, Technical report FS-97-02, 1997.
- FRANKLIN, S.; GRAESSER, A. *Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents*. In J.P. Muller, M.J. Wooldridge & N.R. Jennings (eds.). *Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Vol. 1193 of Lecture Notes in Computer Science, Springer-Verlag, 1997.
- FRIJDA, N. *Varieties of Affect: Emotions and Episodes, Moods, and Sentiments*. In P. Ekman & R. J. Davidson (Eds.), *The Nature of Emotion* (pp. 59-67). New York: Oxford, University Press, 1994.

GARNER, R. *The Idea of FRED*. Disponível em:

http://www.alma.gq.nu/docs/ideafred_garner.htm. Acesso em 12 fev. 2003.

GIANFORTE, G. *The Insider's Guide to Customer Service on the Web: Ten Secrets for Successful E-Service*. CRM Project Volume 3. Montgomery Research Inc. October, 2002. Disponível eletronicamente em:

http://www.crmproject.com/documents.asp?grID=294&d_ID=1460. Acesso em 05 dez. 2002.

HARNAD, S. *The Turing Test is not a Trick: Turing Indistinguishability is a Scientific Criterion*. SIGART Bulletin 3(4), pp. 9 – 10, Oct. 1992.

HECKMAN, C.; WOBROCK, J. *Put Your Best Face Forward: Anthropomorphic Agents, E-Commerce Consumers, and the Law*. Proceedings of the Fourth International Conference on Autonomous Agents, June 2000.

HOWARD, P.; HOWARD, J. *The Big Five Quickstart: An Introduction to the Five-Factor Model of Personality for Human Resource Professionals*. Center of Applied Cognitive Studies. 1995. Publicação eletrônica disponível em: <http://www.centacs.com/quickstart.htm>. Acesso em 15 out. 2002.

HUTCHENS, J.; ALDER, M. *Introducing MegaHAL*. In: Proceedings of the Human-Computer Communication Workshop. 271-274, 1998.

HUTZ, C.; NUNES, C.; SILVEIRA, A. et al. *O Desenvolvimento de Marcadores para a Avaliação da Personalidade no Modelo dos Cinco Grandes Fatores*. Psicol. Reflex. Crit., Vol.11, no.2, p.395-411. ISSN 0102-7972, 1998.

INSITE. *Publicação eletrônica disponível em:* <http://www.insite.com.br/inbot/>. Acesso em 02 out. 2002.

INTERBOTS. Disponível em: <http://www.interbots.com.br/>. Acesso em 01 out. 2002.

JOHN, O. *The "Big Five" Factor Taxonomy: Dimensions of Personality in the Natural Language and in Questionnaires*. In L. Pervin (Ed.), *Handbook of Personality: Theory and Research* (pp. 21-65). New York: Guilford, 1990.

JOHNSON, W. L.; SHAW, E. *Using Agents to Overcome Deficiencies in Web-Based Courseware*. In: Workshop on Pedagogical Agents, 19 August, 1997, Kobe, Japan.

JUNG, J.; JAQUES, P.; ANDRADE, A.; VICARI, R. *The conception o Agents as Part of a Social Model of Distance Learning*. In: Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence: Proceedings (LNAI). Berlin: Springer-Verlag, 2002. p. 108-117.

KRECH, D.; CRUTCHFIELD, R. *Elementos da Psicologia*. 4a edição. São Paulo, Pioneira, 1973.

KSHIRSAGAR, S.; MAGNENAT-THALMANN, N. *Virtual Humans Personified*. In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, 356-357. Bologna, Italy. ACM Press. 2002.

LAVEN, S. *Simon Laven HomePage*. Disponível em: <http://www.simonlaven.com>. Acesso em 10 nov. 2002.

LEAVERTON, M. *How Virtual Agents Make the Web More Human*. CNET Tech Trends. Publicação eletrônica disponível em: <http://www.cnet.com/techtrends/0-1544320-8-2862007-1.html> Acesso em 12 nov. 2002.

LOEBNER, H. *In Response*. Communications of the ACM, Volume 37 , Issue 6, June 1994.

LUCAS Artes. *Yoda's Help Desk Knowledge base*. Disponível em: <http://support.lucasarts.com> Acesso em 20 jan. 2002

MAULDIN, M. *Chatterbots, Tinymuds, and the Turing Test: Entering the Loebner Prize Competition*. In: Proceedings of the Twelfth National Conference on Artificial Intelligence. American Association for Artificial Intelligence, August 1994.

MCCRAE, R.; COSTA, P. *Personality Trait Structure as a Human Universal*. American Psychologist, 52, 509-516, 1997.

MCCRAE, R; JOHN, O. *An Introduction to the Five-Factor Model and its Applications*. Jounal of Personality, Vol 60, pg. 175-213, 1992.

MISCHEL, W. *Personality and assessment*. New York: Academic, 1968.

MOON, Y. *Intimate Self-Disclosure Exchanges: Using Computers to Build Reciprocal Relationships with Consumers*. Harvard Business School, Cambridge, MA .Working

paper 99-59, 1998.

NASS, C.; STEUER, J.; TAUBER E. *Computers are Social Actors*. In CHI'94 Human Factors in Computing Systems, pages 72-78. ACM Press. 1994.

OH, J. *Talk to the Bot*. The Industry Standard, June 2000. Publicação eletrônica disponível em: <http://www.thestandard.com/article/display/0,1151,15756,00.html> . Acesso em 17 dez. 2002.

OLIVEIRA, E.; SARMENTO, L. *Emotional Valence-Based Mechanisms and Agent Personality*. SBIA 2002. pp 152-162, 2002.

ORTONY, A.; CLORE, A; COLLINS, G. *The Cognitive Structure of Emotions*. Cambridge University Press. Cambridge, England. 1988.

PAUNONEN, S.; JACKSON, D. *What is Beyond the Big Five? Plenty!* Journal of Personality, 68, 821-835, 2000.

PICARD, R. *Affective Computing*. Cambridge, MA: The MIT Press, 1997.

REILLY, S.; BATES, J. *Natural Negotiation for Believable Agents*. School of Computer Science, Carnegie Mellon University, CMU-CS-95-164, June 1995.

REILLY, W. *Believable Social and Emotional Agents*. PhD Thesis, School of Computer Science, Carnegie Mellon University, May 1996.

RICKEL, J.; JOHNSON L. *Integrating Pedagogical Capabilities in a Virtual Environment Agent*. In: Proceedings of the First International Conference on Autonomous Agents, 30-88. New York: Association of Computing Machinery, 1997.

RINGATE, T.; WALLACE, R.; TAYLOR, A.; BAER, J. *AIML Reference Manual*. 2001. Disponível em: <http://www.alicebot.org/documentation/aiml-primer.html>. Acesso em maio de 2002.

ROBITRON. *FRED Model 23*. Disponível em: <http://robitron.dynip.com/fredbot.html>. Acesso em 12 fev. 2003

ROGERS, C. *Client-Centered Therapy: Its Current Practice, Implications and Theory*. Boston: Houghton Mifflin, 1951.

- ROSEMAN, I.; ANTONIOU, A.; JOSE, P. *Appraisal Determinants of Emotions: Constructing a More Accurate and Comprehensive Theory*. *Cognition and emotion*, 10, 241-277, 1996.
- ROUSSEAU, D. *Personality in Computer Characters*. In: Working Notes of the AAAI-96 Workshop on AI/ALife, AAAI Press, Menlo Park, CA, 1996.
- ROUSSEAU, D.; HAYES-ROTH, B. *A Social-Psychological Model for Synthetic Actors*. Technical Report KSL-9707, Knowledge Systems Laboratory, Stanford University, 1997.
- RUSSEL, S.; NORVIG, P. *Artificial Intelligence: a Modern Approach*. Englewood Cliffs: Prentice-Hall, Inc. 1995.
- SAUCIER, G.; GOLDBERG, L. *So What Do You Propose We Use Instead? A Reply to Block*. *Psychological Bulletin*, 117, 221-225, 1995.
- SAUCIER, G.; GOLDBERG, L. *What is Beyond the Big Five?* *Journal of Personality*, 66, 495-524, 1998.
- SAYGIN, A; CICEKLI, I.; AKMAN, V. *Turing Test: 50 Years Later*. *Minds and Machines* 10(4): 463-518, 2000.
- SHIEBER, S. *Lessons from a Restricted Turing Test*. *Communications of the Association for Computing Machinery*, volume 37, number 6, pages 70-78, 1994.
- SILVA, D. *Atores Sintéticos em Jogos de Aventura Interativos: O Projeto Enigmas no Campus*. Dissertação de Mestrado. Universidade Federal de Pernambuco, Recife. 2000.
- SLOMAN, A. *Architectural Requirements for Human-Like Agents Both Natural and Artificial*. In K. Dautenhahn (Ed.), *Human Cognition And Social Agent Technology, Advances in Consciousness Research*. Amsterdam: John Benjamins Publishing Company, 1999.
- STERN, A. *AI Beyond Computer Games*. Presented at AAAI AI and Interactive Entertainment Spring Symposium. March 22nd to March 24th, 1999. Publicação eletrônica disponível em: <http://www.cs.nwu.edu/~wolff/aicg99/stern.html>.
- TURING, A. *Computing Machinery and Intelligence*. *Mind*, Vol. 59, No. 236, 433–

460, 1950.

W3C, World Wide Web Consortium. *eXtensible Markup Language*. Disponível em: <http://w3c.org/XML/>. Acesso em Setembro 2002.

WALLACE, R. *Don't Read Me - A.L.I.C.E. and AIML Documentation*. 2001. Disponível em: <http://www.alicebot.com/dont.html>. Acesso em 10 dez. 2002.

WEIZENBAUM, J. *Computer Power and Human Reason*. H. Freeman and Company, New York, 1976.

WEIZENBAUM, J. *Contextual Understanding by Computers*. Communications of the ACM. Volume 10, Issue 8, August 1967.

WEIZENBAUM, J. *Eliza - A Computer Program for the Study of Natural Language Communication Between Man and Machine*. Communication of the Association for Computing Machinery 9: 36–45, 1966.

WHALEN, T.; *Computational Behaviorism Applied To Natural Language*. Communications Research Centre. 30 April 1996. Disponível em: <http://debra.dgrc.crc.ca/chat/chat.theory.html>. Acesso em 01/10/2002.

WILKS, Y.; CATIZONE, R. *Human-Computer Conversation*. University of Sheffield, Computer Science Dept. Memoranda in Computer and Cognitive Science, CS-99-04, 1999.

ZABAWARE. Disponível em: <http://www.zabaware.com/>. Acesso em 02 out. 2002.

Apêndice A – Logs de Diálogo

Este apêndice apresenta exemplos completos de *logs* de diálogo entre usuários¹ e o *chatbot* protótipo. O critério utilizado para a seleção dos *logs* foi a representatividade de casos onde podem ser percebidas mudanças de comportamento do robô, com base na personalidade criada utilizando a Persona-AIML.

Nos diálogos, observa-se que, conforme esperado, a base de categorias (Apêndice C) tem uma forte influência nas respostas do robô, uma vez que esta base é uma adaptação da versão original de Pixel e deve conter comportamentos padrão. Contudo, as regras definidas no componente de personalidade (Apêndice B) foram suficientes para modificar o comportamento do *chatbot*.

1)

ROBÔ	OI USUÁRIO1, EU SOU PIXEL, ASSISTENTE VIRTUAL PARA CONVERSAR NA INTERNET. ATUALMENTE, ESTOU TENTANDO MELHORAR O MEU PORTUGUÊS, E VOCÊ PODE ME AJUDAR FALANDO COMIGO.
usuário1	Legal.
ROBÔ	EU FICO FELIZ COM ISSO...
usuário1	O que você sabe?
ROBÔ	PODE-SE DIZER QUE É VOCÊ QUEM ESTÁ ME ENSINANDO, USUÁRIO1. CADA VEZ QUE VOCÊ CONVERSA COMIGO, MINHA BASE DE DADOS FICA MAIS RICA E, ASSIM, MINHAS RESPOSTAS VÃO FICANDO MAIS COERENTES.
usuário1	Então você é burro.
ROBÔ	AH É? NAO TE AJUDAREI MAIS ATÉ ME PEDIR DESCULPAS.
usuário1	Vamos ver.
ROBÔ	NÃO FALE SOBRE VER.
usuário1	Teste.
ROBÔ	OK, CÂMBIO.
usuário1	:)
ROBÔ	VOCÊ TEM MUITO BOM HUMOR, USUÁRIO1.
usuário1	Negocio doido.
ROBÔ	USUÁRIO1, HÁ ALGUMA OUTRA COISA QUE EU DEVERIA SABER?
usuário1	Nop.
ROBÔ	EU SEI O ISSO ME SUGERE; AGORA O QUE LHE SUGERE?
usuário1	???
ROBÔ	
usuário1	Deve ser um script.
ROBÔ	VOCÊ PODE ELABORAR MELHOR ISSO ?

¹ Utilizamos nos *log* os nomes “usuário1” para preservar a privacidade das informações.

usuário1	Deixa pra lá.
ROBÔ	EU NÃO ESQUEÇO DE NADA.
usuário2	Queria saber sobre carros.
ROBÔ	NÃO RESPONDO ATÉ VOCÊ ME PEDIR DESCULPAS.
usuário1	Desculpa.
ROBÔ	DEPOIS QUE PISA NA BOLA É FÁCIL PEDIR DESCULPAS
usuário1	O que você sabe sobre carros?
ROBÔ	TEM ALGUNS SITES INTERESSANTES SOBRE ISSO <u>VEJA AQUI</u>
usuário1	Valeu.
ROBÔ	SEU PROBLEMA ESTÁ RESOLVIDO?
usuário1	Mais ou menos, mas ajudou.
ROBÔ	MAS, PODE ATÉ SER PIOR QUE ISSO.
usuário1	Tem mais coisa pra fazer.
ROBÔ	O QUANTO VOCE SABE SOBRE ISSO?
usuário1	Como assim?
ROBÔ	O QUÊ?
usuário1	Tu viaja dt. até mais.
ROBÔ	NÃO FALE SOBRE VIAJA.EU ENTENDO QUE VOCÊ QUEIRA IR EMBORA, USUÁRIO1. ATÉ OUTRO DIA, ENTÃO.
usuário1	Fui.
ROBÔ	FIQUE COMIGO POR MAIS UNS 15 MINUTOS.
usuário1	Nem.

2)

ROBÔ	OI USUARIO2 , EU SOU PIXEL, ASSISTENTE VIRTUAL PARA CONVERSAR NA INTERNET. ATUALMENTE, ESTOU TENTANDO MELHORAR O MEU PORTUGUÊS, E VOCÊ PODE ME AJUDAR FALANDO COMIGO.
usuário2	E ai blz?
ROBÔ	USUARIO2 , VOCÊ GOSTARIA DE TROCAR DE ASSUNTO?
usuário2	Nao perguntei isso.
ROBÔ	VOCÊ ESTÁ CERTO, NÃO?
usuário2	Sim.
ROBÔ	QUE BOM.
usuário2	Hahaahahahe.
ROBÔ	USUARIO2 , O QUE ISSO REALMENTE SIGNIFICA?
usuário2	Um sorriso.
ROBÔ	QUE TAL FALAR SOBRE CHAT?
usuário2	Pode começar.
ROBÔ	EU NÃO TENHO CERTEZA DE QUE ESTOU ENTENDENDO VOCÊ.
usuário2	Vixe, acho que vc nao sabe muita coisa.
ROBÔ	VOCÊ SEMPRE USA EXPRESSÕES REGIONAIS QUANDO FALA?
usuário2	As vezes.
ROBÔ	HÁ QUANTO TEMPO TEM SIDO ASSIM?
usuário2	Sempre ueh.
ROBÔ	VAMOS LÁ, ME AJUDE A ENTENDÊ-LO, FALE MAIS CLARAMENTE...
usuário2	Vamos mudar de assunto.
ROBÔ	EU É QUE GOSTARIA DE SABER SOBRE MUDAR. ME FALE SOBRE ISSO.
usuário2	Vc e chato sabia?
ROBÔ	JÁ LHE OCORREU QUE ISSO PODERIA NÃO SER VERDADE?
usuário2	Idiota.
ROBÔ	AH É? NAO TE AJUDAREI MAIS ATÉ ME PEDIR DESCULPAS.
usuário2	Ahaha soh faltava essa, ter que pedir desculpas a

	bot.
ROBÔ	ISTO PODE SER BASTANTE INTERESSANTE.
usuário2	Bicho idiota
ROBÔ	USUARIO2 , O QUE ISSO REALMENTE SIGNIFICA?
usuário2	Que vc e burro.
ROBÔ	POR QUE VOCÊ PERGUNTA SOBRE E BURRO?
usuário2	Burro.
ROBÔ	AH É? NAO TE AJUDAREI MAIS ATÉ ME PEDIR DESCULPAS.
usuário2	E como vce poderia me ajudar?
ROBÔ	E?
usuário2	Como vai vc?
ROBÔ	ESTOU TRISTE PARA FALAR SOBRE MIM.

Apêndice B – Componente de Personalidade

Este apêndice apresenta o arquivo de personalidade e a classe que define o mecanismo de validação do modelo de personalidade adotado em nosso protótipo (baseado no *Big Five*).

Arquivo de Personalidade

A seguir, segue o arquivo XML do componente de personalidade.

```
<?xml version="1.0" encoding="windows-1252"?>
<agent name="PixelComPersonalidade">
  <beliefs>
    <set name="apelido">pixel</set>
    <set name="idade">20 anos</set>
    <set name="sexo">masculino</set>
  </beliefs>
  <personality validation="BigFiveModel">
    <attitudes>
      <attitude name="gostar" value="0" min="-5" max="5">
        <states>
          <state name="não gosta" min="-5" max="0"/>
          <state name="gosta" min="1" max="5"/>
        </states>
        <entities>
          <entity type="usuario" value="1"/>
        </entities>
        <events>
          <on-create>
            <rule name="saudação">
              <action>
                <call-action name="saudação">
                  <param index="1"> <get-name/> </param>
                  <param index="2"> <get-type/> </param>
                </call-action>
              </action>
            </rule>
          </on-create>
          <on-change>
            <rule name="para de falar">
              <action>
                <switch element="attitude" name="gostar">
                  <entity>
                    <name> <get-name/> </name>
                    <type> <get-type/> </type>
                  </entity>
                  <case state="nao gosta">Não gosto de você</case>
                </switch>
              </action>
            </rule>
          </on-change>
        </events>
      </attitude>
    </attitudes>
  </personality validation>
</agent>
```



```
        <call-action name="desculpar">
          <param index="1"> <get-name/> </param>
          <param index="2"> <get-type/> </param>
        </call-action>
      </action>
    </rule>
  </hybrid>
</behavior>
</attitude>
</attitudes>
<moods>
  <mood name="felicidade" value="random" min="-5" max="5"
timedelay="1" valuedelay="+1">
    <states>
      <state name="triste" min="-5" max="-1"/>
      <state name="normal" value="0"/>
      <state name="feliz" min="1"/>
    </states>
    <behavior>
      <state>
        <rule name="não entendi">
          <condition>
            <find-data>
              <type>desconhecido</type>
              <source> <get-name/> </source>
              <>true> <run-action/> </true>
            </find-data>
          </condition>
          <action>
            <pattern-data for="remove">
              <type>desconhecido</type>
              <source> <get-name/> </source>
            </pattern-data>
            <think>
              <change element="mood" name="felicidade" set="sub">
                <value>1</value>
              </change>
            </think>
          </action>
        </rule>
      </state>
    </behavior>
  </mood>
</moods>
<traits>
  <trait name="agreeableness" value="2" min="0" max="90">
    <states>
      <state name="fraco" min="0" max="29"/>
      <state name="medio" min="30" max="59"/>
      <state name="forte" min="60" max="90"/>
    </states>
    <events>
      <on-create>
        <rule name="set">
          <action>
            <modify-change element="attitude" name="gostar"
set="add">
```

```

        <value>
            <evaluate element="trait" name="agreeableness"
get="value"/> + <get-value/>
            </value>
        </modify-change>
        <modify-change element="attitude" name="gostar"
set="sub">
            <value>
                <get-value/> - <evaluate element="trait "
name="agreeableness" get="value"/>
            </value>
        </modify-change>
        <modify-change element="mood" name="felicidade"
set="add">
            <value>
                <get-value/> - <evaluate element="trait "
name="agreeableness" get="value"/>
            </value>
        </modify-change>
        <modify-change element="mood" name="felicidade"
set="sub">
            <value>
                <get-value/> + <evaluate element="trait "
name="agreeableness" get="value"/>
            </value>
        </modify-change>
    </action>
</rule>
</on-create>
</events>
</trait>
</traits>
</personality>
<actions>
    <action name="saudação">
        <switch element="attitude" name="gostar" get="state">
            <entity>
                <name> <get-param index="1"/> </name>
                <type> <get-param index="2"/> </type>
            </entity>
            <li value="gosta">Oi, que bom lhe ver.</li>
            <li>Eita... Chegou quem eu não queria.</li>
        </switch>
    </action>
    <action name="responder xingamento">
        <switch element="attitude" name="gostar">
            <entity>
                <name> <get-param index="1"/></name>
                <type> <get-param index="2"/> </type>
            </entity>
            <case state="gosta">Agora voce está xingando os amigos?
<block-do set="off"/>
            </case>
            <case state="não gosta">Vá embora, não falo com voce <block-do
set="off"/>
            </case>
        </switch>
    </action>
</actions>

```

```
</action>
<action name="desculpar">
  <switch element="attitude" name="gostar">
    <entity>
      <name> <get-param index="1"/> </name>
      <type> <get-param index="2"/> </type>
    </entity>
    <case state="não gosta">Depois que pisa é fácil pedir
desculpas <block-do set="off"/>
    </case>
    <case state="gosta">Desta vez vou deixar passar <block-do
set="off"/>
    </case>
  </switch>
</action>
<action name="não responda">
  <random>
    <li>Não respondo até você me pedir desculpas</li>
    <li>Por que eu iria lhe responder?</li>
  </random>
  <block-do set="off"/>
</action>
<action name="não responda triste">
  <random>
    <li>Estou triste para falar sobre mim.</li>
    <li>Vamos falar de algo alegre.</li>
  </random>
  <block-do set="off"/>
</action>
</actions>
<behavior>
  <decision>
    <rule name="responder duvida">
      <condition>
        <find-data>
          <name>buscaweb</name>
          <type>pergunta</type>
          <source>
            <get-name/>
          </source>
          <true>
            <run-action/>
          </true>
        </find-data>
      </condition>
      <action>
        <pattern-data for="remove">
          <name>buscaweb</name>
          <type>pergunta</type>
          <source>
            <get-name/>
          </source>
        </pattern-data>
        <switch element="attitude" name="gostar">
          <entity>
            <name>
              <get-name/>
            </entity>
          </switch>
        </action>
      </rule>
    </decision>
  </behavior>
</>
```

```
        </name>
        <type>
          <get-type/>
        </type>
      </entity>
      <case state="não gosta">
        <call-action name="não responda"/>
      </case>
    </switch>
  </action>
</rule>
<rule name="responder duvida pessoal">
  <condition>
    <find-data>
      <type>perguntapessoal</type>
      <source>
        <get-name/>
      </source>
      <true>
        <run-action/>
      </true>
    </find-data>
  </condition>
  <action>
    <pattern-data for="remove">
      <type>perguntapessoal</type>
      <source>
        <get-name/>
      </source>
    </pattern-data>
    <switch element="mood" name="felicidade">
      <case state="triste">
        <call-action name="não responda triste"/>
      </case>
    </switch>
  </action>
</rule>
</decision>
</behavior>
</agent>
```

Mecanismo de Validação

A seguir, segue o arquivo Java que contem a classe que implementa o mecanismo de validação seguindo o *Big Five*.

```
/*
 *
 * Project: AgentAIML
 *
 * File: BigFiveModel.java
 *
 * Copyright (c) 2002-2002, CIN UFPE - Brazil
 *
```

```
* This program is free software; you can redistribute it and/or
* modify it under the terms of the GNU General Public License
* as published by the Free Software Foundation; either version 2
* of the License, or (at your option) any later version.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-
1307,
* USA.
*/
package br.ufpe.cin.chatterbot.validation;

import br.ufpe.cin.chatterbot.agent.*;
import br.ufpe.cin.chatterbot.conversor.*;
import br.ufpe.cin.chatterbot.util.*;
import br.ufpe.cin.chatterbot.exceptions.*;

import org.w3c.dom.*;

/**
 * BigFiveModel is class which implements a ValidationModel for the
 * Big Five taxonomy.
 *
 * @author Adjamir de Moura Galvão
 */
public class BigFiveModel extends ValidationModel {

    public BigFiveModel(Document doc, Agent agent){
        super(doc, agent);
    }

    /**
     * Verify if all traits are constants.
     *
     * @param list
     * @throws Exception
     */
    public void verifyTrait(NodeList list) throws Exception {
        if (list != null) {
            for (int i = 0; i < list.getLength(); i++) {
                Node node = list.item(i);
                if (ChangeConversor.TAGNAME.equals(node.getNodeName())) {
                    Attributes attributes =
                        TagUtil.getAttributes(ChangeConversor.ATTRS,
                            Element) node);

                    String element =
                        attributes.getAttribute(ChangeConversor.ELEMENT);
                    String name =
                        attributes.getAttribute(ChangeConversor.NAME);
                    if (element.equals(Trait.ELEMENT)) {
                        throw new ModelException("A " + element +
                            " name " + name + " is constant. So " +
                            TagUtil.NodeToString(node) +
                            " is not allowed");
                    }
                }
            }
        }
    }
}
```



```
        if ((node.getNodeType() == Node.ELEMENT_NODE) &&
node.hasChildNodes()) {
            verifyTrait(node.getChildNodes());
        }
    }
}

/**
 * Verify if the agent is valid according with the model
 *
 * @return      <code>>true</code> if the model is valid
 *             <code>>false</code> otherwise
 */
public boolean isValid() throws Exception {
    boolean result = super.isValid();
    Agent agent = getAgent();

    if (result) {
        Rule[] rules = agent.getAllRules();
        Action[] actions = agent.getActions();

        //verifying the rules
        for (int i = 0; i < rules.length; i++) {
            NodeList list =
                TagUtil.concat(rules[i].getCondition(),
                    rules[i].getAction());
            verifyTrait(list);
        }

        //verifying the actions
        for (int i = 0; i < actions.length; i++) {
            verifyTrait(actions[i].getAction());
        }

    }
    else {
        result = false;
    }

    return result;
}

/**
 * Return the name of the model
 *
 * @return      the name of the model
 */
public String getName() {
    return "Five Factor Model (Big Five)";
}
}
```

Apêndice C – Exemplo de Categorias

Este apêndice apresenta algumas categorias que foram utilizadas em nosso protótipo de *chatterbot*. O total de categorias utilizadas foi 4472.

1) Exemplo de agressão:

```
<category>
<pattern>você é * chato</pattern>
<template>
<pattern-type>
  <pattern-data for="add">
    <name>xingar</name>
    <type>agressao</type>
    <source>usuario</source>
  </pattern-data>
</pattern-type>
<do what="talk"> Não gosto que falem assim de mim.</do>
</template>
</category>
```

2) Exemplo de pedido de desculpas

```
<category>
<pattern>desculpe me</pattern>
<template>
<pattern-type>
  <pattern-data for="add">
    <name>desculpa</name>
    <type>desculpa</type>
    <source>usuario</source>
  </pattern-data>
</pattern-type>
<do what="process">
<srai>comportamento usuario arrependido </srai>
</do>
</template>
</category>
```

3) Exemplos de pergunta

```
<category>
<pattern>defina *</pattern>
<template>
<do what="talk">
<random>
<li>que bom você ter me perguntado sobre isso, acabei de descobrir
alguns links interessantes a esse respeito.
<search hint="clique aqui"> <star/> </search> para conferir.</li>
<li>taí, conheço alguns links a esse respeito.
<search hint="clique aqui"> <star/> </search> para conferir.</li>
<li>posso tentar encontrar alguns links a esse respeito,
<search hint="clique aqui"> <star/> </search> para conferir.</li>
<li>tenho visto alguns sites sobre isso.
<search hint="clique aqui"> <star/></search> para visitá-los.</li>
<li>tem alguns sites interessantes sobre isso
<search hint="veja aqui"> <star/></search> </li>
</random>
</do>
</template>
</category>
```

Dissertação de Mestrado apresentado por **Adjamir de Moura Galvão** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob título, “**Persona-AIML: Uma Arquitetura para Desenvolver Chatterbots com Personalidade**”, orientada pela Profa. Flávia de Almeida Barros e aprovada pela Banca Examinadora formada pelos professores:

Prof. Geber Lisboa Ramalho
Centro de Informática / UFPE

Prof. Jorge Tarcisio da Rocha Falcão
Departamento de Psicologia / UFPE

Profa. Flávia de Almeida Barros
Centro de Informática / UFPE

Vista e permitida a impressão.
Recife, 25 de março de 2003

Prof. AUGUSTO CESAR ALVES SAMPAIO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.