CLOUSEAU: UM MODELO PARA CLASSIFICAÇÃO E CONTROLE DE AGENTES PESSOAIS COMPUTACIONAIS

Dissertação Submetida ao
Curso de Ciência da Computação
da Universidade Federal Fluminense
como Requisito Parcial para Obtenção do
Grau de Mestre

por

André Vargas Matos da Cunha e Leiradella

Orientadora: Ana Cristina Bicharra Garcia, Ph.D.

março 2005

.

BANCA EXAMINADORA

CLOUSEAU: UM MODELO PARA CLASSIFICAÇÃO E CONTROLE DE AGENTES PESSOAIS COMPUTACIONAIS

Ana Cristina Bicharra Garcia, Ph.D.
Universidade Federal Fluminense

Flavio Miguel Varejão, Ph.D.
Universidade Federal do Espírito Santo

Viviane Torres da Silva, D.Sc. (suplente)
Pontifícia Universidade Católica do Rio de Janeiro

março 2005

RESUMO

Agentes computacionais pessoais têm exercido um papel cada vez maior em ajudar pessoas a completar tarefas ordinárias, como edição de textos e classificação de mensagens de correio eletrônico. Porém, apesar de seu potencial benéfico, eles também podem ser inconvenientes e até mesmo prejudiciais aos seus usuários. Neste trabalho é proposto um modelo de classificação de agentes de acordo com dois eixos, inteligência e autonomia, capaz de diagnosticar agentes pró-ativos porém estúpidos antes que possam nos causar danos. O modelo foi implementado no domínio de aplicação de classificação de mensagens de correio eletrônico, no qual agentes anti-*spam* auxiliam seus usuários a excluir mensagens *spam*.

ABSTRACT

Personal computer agents have increasingly played an important role assisting people to accomplish ordinary tasks, such as text editing and e-mail handling. In spite of their potential benefit, they can also be inconvenient and even prejudicial to their users. In this work a model to classify agents according to the axes intelligence and autonomy, and capable of diagnosing proactive dummies before they damage our lives is proposed. The model was implemented in the anti-spam domain, in which anti-spam agents help their users to automatically delete spam messages.

DEDICATÓRIA

Dedico este trabalho à minha esposa Carla e ao meu filho Lucas, a quem privei da minha companhia para poder dedicar-me às pesquisas e aos estudos. Dedico-o também aos meus pais, pelo amor recebido, aos meus tios e a toda a minha família, especialmente ao meu avô Augusto, que morreu em Portugal no dia 7 de julho de 2004 sem ter oportunidade de conhecer o seu único bisneto, nos ombros de quem recai a responsabilidade de dar continuidade à tradicional e honrosa Casa das Leiras.

AGRADECIMENTOS

Agrageço à minha esposa Carla, pelo amor, paciência e motivação, pois acredito que sem isto não teria chegado até aqui.

Agradeço ao meu filho, pela alegria que trouxe à minha vida, e pela motivação, mesmo que inconsciente, que me faz buscar crescer como pessoa, para que ele possa ter em mim um modelo de pai e um amigo, e profissionalmente, para que eu possa lhe permitir um futuro sem limites.

Agradeço à minha orientadora Ana Cristina Bicharra Garcia pela paciência e compreensão, e também por não ter desistido de mim mesmo após prolongados períodos de silêncio de minha parte.

Agradeço aos membros da banca, pelo tempo dedicado à leitura e avaliação deste trabalho.

Agradeço ao meu empregador, Xerox Comércio e Indústria, por ter permitido que eu me ausentasse do trabalho para comparecer às aulas do curso de mestrado.

CONTEÚDO

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

ÍNDICE DE LISTAGENS

Capítulo 1

Introdução

1.1 Problema

A tecnologia traz facilidades e conforto para nossas vidas. Além disso, ela traz uma maior quantidade de informações e responsabilidades para cada um de nós. Se antes nos adaptávamos à falta de soluções alternativas, agora devemos encarar um enorme mundo de possibilidades a serem assimiladas.

Neste cenário, agentes computacionais pessoais têm uma função importante em nossas vidas. Agentes trabalham nos servindo em necessidades específicas. Seu trabalho especializado nos ajuda em tarefas repetitivas e nos poupa tempo. Quanto mais autônomos eles são, maior o suporte que nos provêm.

Por outro lado, agentes podem ser inconvenientes, nos impondo o fardo de desfazer suas ações. Por exemplo, suponhamos um agente de edição de texto que insiste em consertar erros de digitação incorretamente. Cada vez que ele age de forma imprópria nós temos que parar a digitação do texto e clicar no botão "desfazer". A figura 1 ilustra um agente como este.

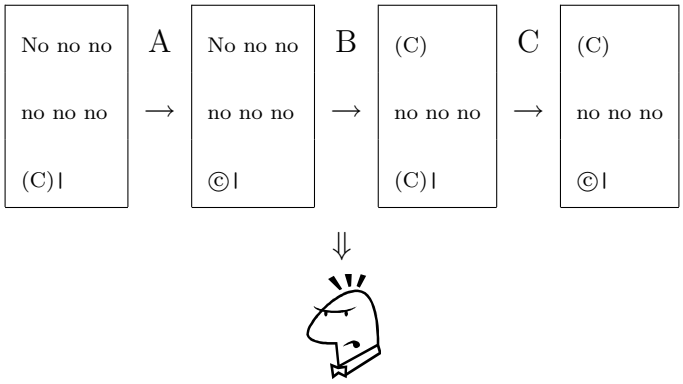

Figura 1: Agente de edição de texto atrapalhando seu usuário

Em A, o agente modifica o texto do usuário, trocando a seqüência de caracteres "(C)" por "©". Em B, o usuário desfaz a ação do agente e continua digitando seu texto. Porém, em C, o agente percebe novamente a seqüência de caracteres "(C)" e novamente a troca por "©", prejudicando o andamento do trabalho do usuário e deixando-o insatisfeito. Provavelmente, após algumas interações ruins, o usuário irá desligar o agente. Este cenário pode piorar se o usuário não puder desfazer as ações do agente.

Como um outro exemplo, podemos citar os vírus de computador, que são agentes mal intencionados e que podem causar sérios danos aos usuários dos computadores que infectam. Toda uma indústria de programas de anti-vírus e de sistemas de proteção contra invasões existe em torno destes agentes, e as empresas que atuam neste mercado realizam constantes pesquisas para melhorar seus produtos e proteger os computadores de seus clientes contra novos vírus, que são produzidos em grande quantidade e sempre usando técnicas inovadoras.

Somos constantemente avisados e estamos sempre alertas a estes agentes mal intencionados. Não existe uma relação de confiança entre um usuário de computador e um vírus, e assim que detectamos um em nosso sistema, utilizamos as ferramentas necessárias para apagá-lo de imediato, antes que cause maiores problemas.

O quadro é invertido quando utilizamos um agente como um assistente pessoal. Assumimos, por se tratar de um agente bem intencionado, feito para nos ajudar, que ele nunca nos prejudicará. Mas quando um agente bem intencionado age de forma estúpida, ele pode causar sérios danos ao seu usuário, devido à confiança que nele é depositada. Como exemplo, um usuário pode instalar um programa anti-vírus e dar a ele autonomia para procurar por vírus em seu sistema e eliminá-los,

mas este programa pode excluir arquivos importantes do usuário que encontram-se infectados ao invés de apenas retirar os vírus, talvez causando mais dano ao usuário do que o próprio vírus.

Podemos concluir que, independente de sua natureza (boa ou má), o que determina a utilidade de agentes é a forma como agem, e o pior cenário é ter um agente estúpido e completamente autônomo. Ele pode prejudicar a vida de seus usuários severamente. Nós somos avisados e prestamos atenção em agentes maléficos, como vírus, mas não contra agentes bem intencionados, porém estúpidos.

1.2 Motivação

Hoje a maioria dos agentes descritos na literatura de Inteligência Artificial são desenvolvidos por quem os usa ou testa. Mas com a expansão do uso destes programas autônomos para fora do âmbito acadêmico ou altamente especializado, o foco deve se modificar da criação para a contratação de agentes. Usuários de computadores poderão escolher agentes de diferentes desenvolvedores para agir em seus nomes e executar as mais diversas tarefas. A estes agentes será dada autonomia de ação devido à presunção de que, sendo bem intencionados, não causarão danos. No entanto, é um risco delegar tarefas a agentes, humanos ou computacionais, não apenas em razão de sua intenção, mas em grande parte por sua competência.

Neste cenário de uso de agentes desenvolvidos por outrem mas atuando em nosso nome, torna-se necessário investigar a confiabilidade de um agente a partir de sua atuação. Esta investigação é fundamental para que usuários adquiram confiança em seus agentes e dêem a eles autonomia suficiente para que possam realmente ajudá-los em suas tarefas cotidianas. Autonomia para agir e sucesso do

resultado das ações são variáveis que sabemos irão influenciar na confiabilidade do agente.

Apesar de um usuário de um agente poder concluir através da observação se um agente é bom ou ruim, se ele realmente o auxilia ou o prejudica na execução de seu trabalho, não existe um modelo formal para quantificar o grau de utilidade de um agente. Um usuário que deseja usar um agente como um assistente pessoal não tem qualquer pista de como o agente se comportará. Apenas com o tempo, e talvez arcando com o custo de desfazer ações do agente em várias oportunidades, o usuário poderá avaliar o grau de confiança que nele pode depositar. Porém, é possível que durante este tempo ocorram prejuízos ao usuário advindos de ações estúpidas causadas pelo agente.

Além disso, é difícil para um usuário comparar vários agentes que atuam no mesmo domínio de aplicação para decidir qual o melhor para seu problema em específico. Apenas após usar cada um dos agentes por tempo suficiente o usuário poderá avaliar qual deles irá utilizar, o que multiplica as chances de ocorrência de prejuízo. Ainda assim, há de se questionar a isenção dessa avaliação sujeita a variações de humor por parte do usuário, o que pode levá-lo a escolher um agente que não seja o mais indicado.

Um modelo para que agentes possam ser classificados de forma rápida e objetiva e controlados de forma eficaz se faz necessário, de forma a minimizar o tempo durante o qual o usuário fica exposto às possíveis ações de agentes estúpidos, minimizar os danos que estes agentes possam causar através destas ações e possibilitar uma escolha mais coerente e baseada em fatos, dentre as opções disponíveis na hora de se escolher um agente como um assistente pessoal.

1.3 Modelo

A hipótese que desejamos provar neste trabalho é que um modelo para classificar agentes computacionais pessoais quanto à sua inteligência e autonomia e os controlar pode reduzir o dano que agentes bem intencionados porém estúpidos possam causar aos seus usuários.

1.4 Método de Avaliação

Uma implementação do modelo proposto no domínio de aplicação anti-*spam* será realizada, e um conjunto de testes que possam aferir sua correção e a redução do dano causado por agentes anti-*spam* serão executados.

1.5 Organização da Dissertação

Esta dissertação está organizada da seguinte forma: o capítulo um explica o problema que desejamos resolver, a motivação deste desejo, a hipótese que queremos provar, e o método de avaliação usado para testar a hipótese.

O capítulo dois introduz conceitos importantes para o entendimento dos demais capítulos da dissertação, como agentes computacionais, agentes anti-*spam*, modelos de classificação e lógica nebulosa.

O capítulo três introduz o Modelo Clouseau, e detelha seus módulos, e o capítulo quatro detalha o experimento realizado para testar o modelo, com seus módulos, e fornece os resultados deste experimento.

O capítulo cinco mostra trabalhos correlatos ao Modelo Clouseau, traçando entre eles um comparativo, e o capítulo seis conclui a dissertação com as contribuições do trabalho, suas limitações e os trabalhos futuros.

O apêndice A contém a referência das mensagens existentes no ambiente de teste em FIPA ACL, e o apêndice B contém os diagramas das classes usadas no experimento.

Capítulo 2

Conceitos

2.1 Agentes Computacionais

2.1.1 Definição

Apesar de não existir uma definição universalmente aceita de agente, uma das mais abrangentes [Garcia e Sichman, 2003] diz:

> *Um agente é uma entidade real ou virtual, capaz de agir num ambiente, de se comunicar com outros agentes, que é movida por um conjunto de inclinações (sejam objetivos individuais a atingir ou uma função de satisfação a otimizar), que possui recursos próprios, que é capaz de perceber seu ambiente (de modo limitado), que dispõe (eventualmente) de uma representação parcial deste ambiente, que possui competência e oferece serviços, que pode eventualmente se reproduzir e cujo comportamento tende a atingir seus objetivos, utilizando as competências e recursos que dispõe, e levando em conta os resultados de suas funções de percepção e comunicação, bem como suas representações internas.*

Segundo Russel e Norvig [Russel e Norvig, 1995], agentes podem ser definidos como entidades que podem perceber, através de sensores, o ambiente no qual estão inseridos, agindo sobre ele através de ações. Para Maes [Maes, 1994], agentes são sistemas computacionais utilizados para auxiliar seus usuários, de forma a ajudá-los na realização de tarefas trabalhosas que demandem tempo.

As definições acima são genéricas, podendo compreender tanto um robô móvel (entidade real) quanto um robô virtual (entidade computacional). Contudo, existem ingredientes chave que efetivam a caracterização de um agente [Wooldridge e Jennings, 1994], tais como:

- **Autonomia**: Agentes devem realizar a maior parte de suas tarefas sem a intervenção direta de humanos ou outros agentes, e eles devem possuir um grau de controle sobre suas ações e seu estado interno [Castelfranchi, 1995];

- **Sociabilidade**: Agentes devem poder interagir, quando julgarem apropriado, com outros agentes artificiais ou humanos, para completar suas tarefas e ajudar a outros com suas atividades, através de alguma linguagem de comunicação de agentes [Genesereth e Ketchpel, 1994]. Isto requer que agentes tenham algum meio através do qual possam comunicar seus requerimentos a outros e mecanismos internos para decidir quando interações sociais são apropriadas, tanto em termos de gerar requisições apropriadas quanto em julgar requisições recebidas;

- **Reatividade**: Agentes devem perceber seu ambiente (que pode ser o mundo real, o usuário, uma coleção de agentes, a Internet etc.) e responder em tempo hábil a mudanças que nele ocorram; e

- **Pró-atividade**: Agentes não devem simplesmente responder aos seus ambientes, eles devem ser capazes de exibir comportamentos oportunistas e orientados à objetivos, e tomar a iniciativa quando apropriado.

Às características listadas acima, Huhns e Singh [Huhns e Singh, 1998] adicionam outras para definir agentes:

- **Foco**: Um agente deve ter um plano de atuação ou ser capaz de planejar uma seqüência de ações para alcançar seus objetivos;
- **Adaptabilidade**: Um agente deve aprender com seus erros, acertos e observações, ajustando seu comportamento;
- **Mobilidade**: Um agente deve saber sobreviver, movendo-se entre ambientes diferentes; e
- **Personalidade**: Um agente pode ter um estereótipo ou personalidade.

Outros pesquisadores argumentam que outras propriedades de agentes devem ser enfatizadas:

- **Veracidade**: A assunção de que um agente não irá tornar disponíveis informações falsas "conscientemente" [Galliers, 1989];
- **Benevolência**: A assunção de que agentes compartilham um objetivo comum, e que cada agente irá tentar fazer o que dele se pede [Rosenschein e Genesereth, 1985];
- **Racionalidade**: A assunção de que agentes irão agir para atingir seus objetivos, e não de forma contrária [Russell e Wefald, 1991];
- **Aprendizagem**: A assunção de que agentes irão se adaptar à mudanças em seu ambiente [Wooldridge, 1998]; e
- **Emoção**: A capacidade de agentes de expressar emoções que representam seus estados internos [Bates, 1994].

Em uma tentativa de formalizar a definição de agentes computacionais, Franklin e Graesser [Franklin e Graesser, 1997] definem um agente autônomo, após um

extenso levantamento de definições de agentes existentes na literatura e na Internet [Crystaliz, 1997; Russel e Norvig, 1995; Maes, 1995; Smith *et al.*, 1994; Hayes-Roth, 1995; IBM, 1997; Wooldridge e Jennings, 1995; Coen, 1994; Foner, 1993; 1997; Brustoloni, 1991], como "um sistema situado em e fazendo parte de um ambiente, que sente este ambiente e age sobre ele, através do tempo, perseguindo seus próprios objetivos para afetar o que sentirá no futuro".

A falta de uma definição universalmente aceita para agentes levou Carl Hewitt a comentar que a questão "o que é um agente?" é tão embaraçosa para a comunidade de computação baseada em agentes quanto a questão "o que é inteligência?" o é para a comunidade de Inteligência Artificial[1]. Mas independente desta definição, agentes que se encaixam nas definições acima em maior ou menor grau vêm sendo usados em sistemas de gerenciamento de informações pessoais, comércio eletrônico, design de *interfaces*, jogos de computador, gerenciamento de processos comerciais e industriais e diversas outras aplicações.

Dentre as características listadas, é possível destacar três que afetam fortemente a percepção de usuários em relação à utiliade de seus agentes: autonomia, pró-atividade e racionalidade. As duas primeiras determinam se um agente agirá sem a intervenção de seu usuário ou se o incomodará a cada ação que deseja tomar. A terceira relaciona-se à inteligência do agente que é percebida pelo usuário; agentes racionais bem intencionados que buscam atingir seus objetivos têm potencial para ajudar seus usuários, enquanto agentes irracionais bem intencionados podem causar grandes prejuízos devido à confiança que recebem.

[1]No 13ºWorkshop Internacional em Inteligência Artificial Distribuída.

2.1.2 Tipos de Agentes

Dentre as várias taxonomias para a classificação de agentes (por exemplo [Brustoloni, 1991; Franklin e Graesser, 1997]), a de Caglayan e Harrison [Caglayan e Harrison, 1997] os classificam de acordo com seu ambiente computacional. Esta classificação tem três categorias:

1. **Agentes de *desktop*** atuam exclusivamente no ambiente *desktop* do usuário com o objetivo de ajudá-lo no uso de aplicações locais. Estes agentes são subdivididos em agentes do sistema operacional, que auxiliam o usuário em tarefas relacionadas ao sistema operacional, tais como a instalação de um novo dispositivo, agentes de aplicação, que acessoram o usuário no uso de uma aplicação em particular, e agentes de um conjunto de aplicações, que ajudam o usuário a conseguir uma melhor performance nestas aplicações;

2. **Agentes de Internet** atuam na Internet, atuando em nome do usuário ou para ele simplificando diversas tarefas. Estes agentes podem ser classificados em:

    - Agentes de pesquisa;
    - Agentes de serviços Web, que residem em um sítio da Internet para fornecer serviços auxiliares aos seus visitantes;
    - Agentes de recuperação de informações, que oferecem pacotes personalizados de informações de acordo com as preferências dos usuários;
    - Agentes de filtragem, que filtram informações para seus usuários;
    - Agentes de notificação, que notificam seus usuários da ocorrência de eventos de seus interesses;

- Agentes de serviços, que fornecem serviços especializados aos seus usuários; e
- Agentes móveis, que se movem pela Internet para executar tarefas para seus usuários.

3. **Agentes de Intranet** atuam em redes corporativas, executando serviços específicos destes ambientes. Eles podem ser classificados em agentes colaborativos, que promovem o fluxo de trabalho, agentes de bancos de dados, e agentes de gerenciamento de recursos, que alocam recursos para seus usuários.

Os agentes pessoais anti-*spam* que serão usados para testar o modelo de classificação de agentes quanto à sua confiabilidade se encaixam na categoria de agentes de Internet, pois são usados para filtrar as caixas de correio eletrônico de seus usuários, movendo ou excluindo mensagens que consideram serem irrelevantes para seus usuários.

2.1.3 Aplicações

Agentes computacionais vêem sendo utilizados com sucesso nos mais diversos domínios de aplicação, de forma isolada, ou em ambientes multi-agentes competitivos ou cooperativos. É evidente que seu uso pode nos auxiliar em nossas tarefas cotidianas, sejam elas simples ou complexas. A lista abaixo apresenta alguns exemplos de agentes.

- **Agentes pessoais**
  - Maxims, um agente que aprende a priorizar, excluir, ordenar e arquivar

mensagens de correio eletrônico, ajuda a marcar reuniões com outras pessoas e filtra mensagens de Usenet News [Maes, 1994];

  - WebMate, um agente que ajuda o usuário a navegar pela Internet, aprende os hábitos do usuário e monta uma página de notícias personalizada, e monitora sítios *web* por mudanças [Chen e Sycara, 1998];
  - MS Agent, um agente que auxilia usuários em tarefas do pacote de aplicativos Office [Microsoft, 2003]; e
  - Um agente para o problema de agendamento de reuniões [Hassine *et al.*, 2004], que ajuda na decisão de onde e quando uma reunião deve acontecer baseado nas restrições e preferências das pessoas envolvidas.

- **Navegação**
  - Um sistema de navegação baseado no modelo BDI (*Beliefs, Desires and Intentions*; crenças, desejos e intenções) [Parsons *et al.*, 1999].

- **Comércio**
  - AgILE, um modelo para agentes que atuam em sítios de comércio eletrônico, buscando por boas ofertas e negociando produtos em nome do usuário [Lopes, 2001]; e
  - Botticelli [Benisch *et al.*, 2004] e RedAgent-2003 [Keller *et al.*, 2004], agentes gerenciadores de cadeia de suprimentos.

- **Redes**
  - Roteamento [Gan *et al.*, 2004]; e
  - Aprendizagem de preferências pessoais para provimento de serviços *wireless* [Lee *et al.*, 2004].

- **Colaboração**
  - MultiADD, um modelo para auxiliar em design em grupos interdisciplinares [Vivacqua, 1997]; e
  - Aprendizagem colaborativa multiagente para sistemas de negócios distribuídos [Guo e Müller, 2004].
- **Aplicações científicas**
  - Agente Científico Autônomo EO-1, um agente à bordo da *Earth Observing One Spacecraft* que detecta e responde à eventos científicos que ocorrem na Terra [Chien *et al.*, 2004].

2.2 Agentes Anti-*Spam*

2.2.1 *Spam*

De acordo com Mueller [Mueller, 2000], *spam* é "a inundação da Internet com muitas cópias de uma mesma mensagem, em uma tentativa de forçar seu recebimento por pessoas que de outra forma escolheriam não recebê-la." O adjetivo *spammer* designa aquele que realiza este ato. Quando o a palavra *spam* é utilizada com substantivo, designa mensagens oriundas de atos de *spam*.

Estas mensagens são normalmente correntes ou propagandas de empresas ou negócios, muitas vezes obscuros. Às vezes elas são apenas chamarizes, usadas por *spammers* para validar endereços de correio eletrônico daqueles que as abrem. Usuários regulares de correio eletrônico têm suas caixas postais abarrotadas de *spam*, e perdem um tempo precioso para separá-las de mensagens legítimas e jogá-las fora. A imensidão de aplicativos anti-*spam* existentes na Internet [Mueller,

2005] dá a proporção do contratempo que estas mensagens causam aos usuários de correio eletrônico.

É interessante notar que *spam* não é o mesmo que lixo, pois a classificação de mensagens oriundas de atos de *spam* em relação à sua utilidade é subjetiva.

2.2.2 Agentes Anti-*Spam*

Agentes anti-*spam* são agentes de Internet pessoais, que auxiliam seus usuários a separar e excluir mensagens que consideram serem *spam* de mensagens importantes.

Para executar esta tarefa, estes agentes são dotados de uma lógica interna capaz de classificar e excluir mensagens de acordo com várias características, como por exemplo:

- **Remetente** e **domínio do remetente**: Exclui mensagens originadas de um determinado endereço de correio eletrônico ou de um determinado domínio da Internet;
- **Número de destinatários**: Exclui mensagens copiadas para muitos destinatários ao mesmo tempo;
- **Assunto**: Exclui mensagens com um determinado assunto;
- **Texto da mensagem**: Exclui mensagens que contenham algum texto ou conjunto de palavras; e
- **Anexos**: Exclui mensagens com determinado tipo de anexos.

Esta lógica pode ser determinística, como bloquear todas as mensagens de um mesmo domínio, ou probabilística, usando por exemplo Redes Bayesianas na

análise do texto da mensagem [Graham-Cumming, 2004; Peters *et al.*, 2005].

Independente de sua lógica interna, estes agentes são configurados ou treinados por seus usuários de acordo com suas preferência pessoais, pois não existe uma definição única de mensagens *spam* que possa ser utilizada para todos eles.

Muitos agentes anti-*spam* também ajudam seus usuários a classificar mensagens por assunto, movendo-as automaticamente para pastas correspondentes a estes assuntos (figura 2).

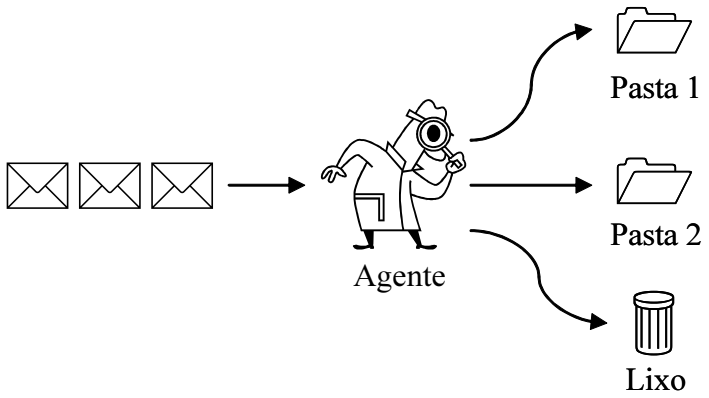

Figura 2: Um agente anti-*spam* classificando mensagens de seu usuário

2.3 Modelos de Classificação

Os humanos vêem inventando taxonomias ao longo de milênios para classificar coisas, vegetais, animais e a si próprios por diversas razões, desde para auxiliar no estudo das espécies até para agilizar análises de crédito, passando pela classificação de pessoas de acordo com suas aptidões e quanto à sua capacidade de raciocínio, como nos testes de QI (Quociente de Inteligência), de estrelas quanto à sua grandeza, geração e emissão de energia.

Estas taxonomias apresentam categorias ou índices que são modelos simplificados dos objetos que classificam, e nos ajudam a compreender problemas que de outra forma seriam extremamente complexos.

A classificação de agentes quanto à sua confiabilidade também deve apresentar esta propriedade. Infinitos graus de inteligência e autonomia podem ser atribuídos a um agente, e diferentes usuários podem atribuir diferentes valores à estas características para o mesmo agente. Um modelo para realizar esta classificação deve ser capaz de reduzir a infinidade de agentes existentes a algumas categorias bem definidas e também de classificá-los de forma objetiva.

Além disso, a automatização de um sistema de classificação de agentes pode ajudar usuários a identificar agentes perigosos com maior antecedência do que seria possível através apenas de sua observação. Controlando o agente diretamente ou informando seu usuário para que ele ajuste os parâmetros do agente, este sistema automático pode ajudar a minimizar o prejuízo ou maximizar o ganho sobre as ações do agente.

2.3.1 O Modelo de Montgomery

O Marechal de Campo britânico Bernard Montgomery, que teve atuação decisiva na Segunda Guerra Mundial, qualificava seus soldados de acordo com sua inteligência e iniciativa. Ele qualificou os soldados em quatro tipos, como ilustrado na tabela 1, alertando quanto ao perigo de se perder uma guerra pelas ações contínuas de soldados estúpidos, porém ativos (soldados do tipo IV).

Soldados do tipo IV devem ser evitados por suas ações potencialmente desastrosas. Soldados do tipo II são os ideais, pois têm iniciativa e tomam ações acertadas. Já os soldados dos tipos I e III não são benéficos nem prejudiciais, pois não têm a

Iniciativa → **Competência ↓**	**Passivo**	**Ativo**
Inteligente	Tipo I	Tipo II
Estúpido	Tipo III	Tipo IV

Tabela 1: O modelo de classificação do General Montgomery

iniciativa necessária para agir.

Agentes computacionais, assim como os soldados do Marechal Montgomery, também podem ser classificados através de sua inteligência e autonomia. Esta abordagem tem algumas vantagens, como a capacidade de classificar agentes de acordo com sua utilidade sem levar em conta sua intenção, e a forte caracterização de agentes que beneficiam seus usuários (tipo II) e de agentes que os prejudicam (tipo IV), o que é de grande valia para estabelecer a confiança que um usuário pode depositar em um agente.

Porém o modelo de Montgomery esbarra na falta de formalismo para se medir a inteligência e a iniciativa dos soldados, o que traz problemas não apenas em uma classificação isolada, mas também quando é desejável a comparação entre vários soldados para que seja possível escolher o mais apropriado para uma tarefa específica.

2.3.2 O Modelo de Cipolla

Em 1988, o historiador econômico Carlo M. Cipolla criou, inspirado nos escritos de Montgomery, uma taxonomia para a classificação de pessoas baseada na vantagem

ou prejuízo que uma pessoa gera à ela própria e à outras [Cipolla, 1988]. Seu modelo tem dois eixos, um representando a vantagem ou prejuízo gerado à própria pessoa e outro representando a vantagem ou prejuízo gerado à outras pessoas (figura 3).

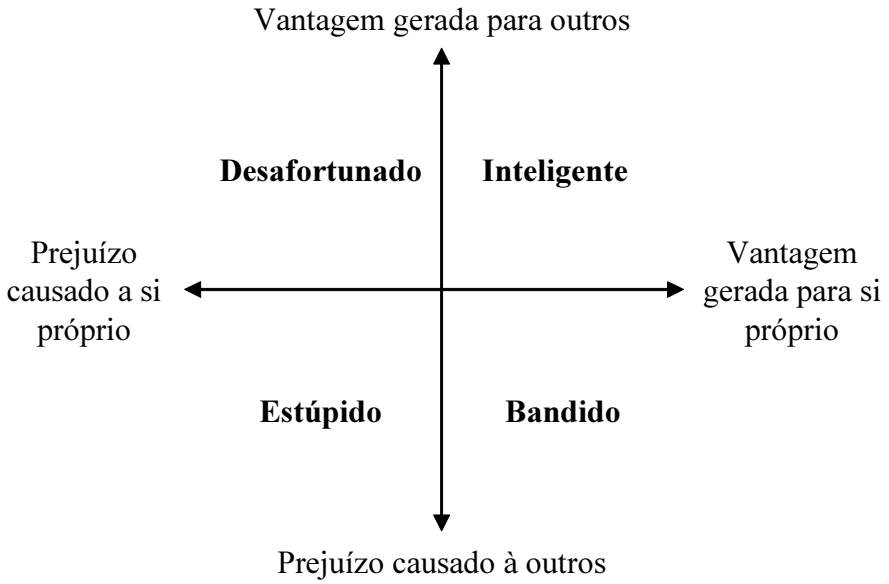

Figura 3: Modelo de classificação de Cipolla

Os quatro tipos de pessoas definidos pelo modelo de Cipolla são:

1. **Desafortunado**: Tende a gerar prejuízo para si, mas vantagem para outros;
2. **Inteligente**: Tende a gerar vantagem para si e para outros;
3. **Estúpido**: Tende a gerar prejuízo para si e para outros; e
4. **Bandido**: Tende a gerar vantagem para si, mas prejuízo para outros.

É possível perceber no sistema de Cipolla que pessoas podem ter um grau maior ou menor de cada um dos tipos, dependendo dos valores computados para cada eixo do sistema, significando que uma pessoa pode ser uma mistura dos tipos ao invés de ser classificada como sendo de um tipo em particular.

Uma outra característica interessante do sistema de Cipolla é o fato de que uma pessoa não é classificada com base em algum conceito abstrato, mas nas conseqüências observadas de seus atos em si próprio e em outras pessoas, que são usadas para computar os valores dos eixos.

Uma vez que agentes podem se encaixar em mais de uma categoria, o modelo de classificação destes quanto à sua confiabilidade deve ser capaz de perceber neles graus variados de autonomia e inteligência. Dizer que um agente é simplesmente inteligente ou estúpido e ativo ou inativo significa perder a capacidade de comparar agentes para o mesmo domínio de aplicação que apresentam pouca diferença em sua atuação e também a capacidade de exercer um controle fino sobre sua autonomia. Uma lógica nebulosa se faz necessária para destacar o agente mais apto de um conjunto de agentes com boa atuação, para permitir que um usuário faça uma escolha fundamentada.

O modelo de classificação deste trabalho utilizará uma junção das características mais úteis para a classificação da confiabilidade de agentes presentes nos modelos de Montgomery e Cipolla: a forte caracterização de tipos de agentes benéficos e maléficos quanto às suas ações e a observação de suas conseqüências como forma objetiva de classificá-los, respectivamente.

2.4 Lógica Nebulosa

A lógica nebulosa foi criada com o objetivo de representar noções incertas ou vagas, nas quais valores intermediários entre os valores verdadeiro e falso da lógica booleana são necessários. Por volta de 400 A.C., Parminedes propôs que proposições poderiam ser verdadeiras e falsas ao mesmo tempo, mas foi Platão quem fundou as bases da lógica nebulosa ao propor uma terceira região entre os valores verdadeiro

e falso.

Em 1920 Łukasiewicz extendeu a lógica bivalorada de Aristóteles e propôs uma lógica trivalorada, na qual um terceiro valor lógico $\frac{1}{2}$ foi adicionado aos valores 0 (falso) e 1 (verdadeiro). Knuth também propôs uma lógica trivalorada, mas utilizando os valores $\{-1, 0, +1\}$ ao invés de $\{0, \frac{1}{2}, 1\}$. Outros estudiosos também dissertaram sobre o assunto, mas foi novamente Łukasiewicz quem experimentou lógicas 4 e 5-valoradas e hipotetizou a possibilidade de uma lógica com infinitos valores entre o verdadeiro e o falso.

Lofti Zadeh publicou, em 1965, o artigo *Fuzzy Sets* (conjuntos nebulosos) [Zadeh, 1965], generalizando a teoria clássica dos conjuntos. Da teoria dos conjuntos nebulosos foi derivada uma lógica multivalorada, com operadores lógicos que têm uma correspondência direta com os operadores de união, interseção e complemento dos conjuntos nebulosos.

Segundo Meyer [Meyer *et al.*, 1993], se for aceito o fato de que a teoria dos conjuntos nebulosos é uma generalização da teoria clássica de conjuntos, praticamente todos as áreas de estudo que se baseiam na matemática numa base conceitual ou teórica podem ser vistas pelo novo ângulo dos conjuntos nebulosos, o que resulta em uma extensa lista de áreas que podem ser retrabalhadas, como por exemplo: lingüística, lógica proposicional, lógica de predicados e todas as lógicas bimodais; probabilidade e todas as áreas baseadas em análise estatística.

2.4.1 Conjuntos Nebulosos

Na teoria clássica de conjuntos, um conjunto pode ser definido por uma função dita característica $\chi_A : U \rightarrow \{0, 1\}$ que associa a cada elemento do universo de discurso U um valor binário. Esta função é expressa por:

$$\chi_A(x) = \begin{cases} 0, \text{ se } x \notin A \\ 1, \text{ se } x \in A \end{cases}$$

Na teoria clássica de conjuntos, o conjunto de A de pessoas altas pode ser definido como:

$$\chi_A(x) = \begin{cases} 0, \text{ se } a_x < 1,80\text{m} \\ 1, \text{ se } a_x \geq 1,80\text{m} \end{cases}$$

onde a_x é a altura de x medida em metros.

A teoria de conjuntos nebulosos engloba a teoria clássica dizendo que funções características podem associar a cada elemento do universo de discurso um valor no intervalo real $[0, 1]$. Neste caso, funções características passam a ser chamadas funções de pertinência, e um elemento pode não pertencer à um dado conjunto ($\mu_A(x) = 0$), pertencer totalmente ao conjunto ($\mu_A(x) = 1$) ou pertencer ao conjunto apenas em parte ($0 < \mu_A(x) < 1$). Quando uma função de pertinência de um conjunto nebuloso associa os elementos do universo de discurso apenas aos valores 0 e 1, tem-se um caso particular da teoria de conjuntos nebulosos, a teoria clássica dos conjuntos [Zadeh, 1965].

O conjunto A de pessoas altas é um exemplo de conjunto nebuloso. A maioria das pessoas concordará que alguém com 1,80 metro é alta. Mas este fato não torna pessoas que medem 1,79 metro baixas. O conjunto A é melhor descrito como um conjunto nebuloso no qual elementos do universo de discurso (alturas de pessoas) têm graus de pertinência no intervalo real $[0, 1]$. A função de pertinência deste conjunto pode ser definida como:

$$\mu_A(x) = \begin{cases} 0, \text{ se } a_x < 1,60\text{m} \\ \frac{a_x - 1,60}{0,20}, \text{ se } 1,60 \leq a_x \leq 1,80\text{m} \\ 1, \text{ se } a_x > 1,80\text{m} \end{cases}$$

Esta função pode ser representada como um gráfico do conjunto nebuloso que ela define, conforme a figura 4.

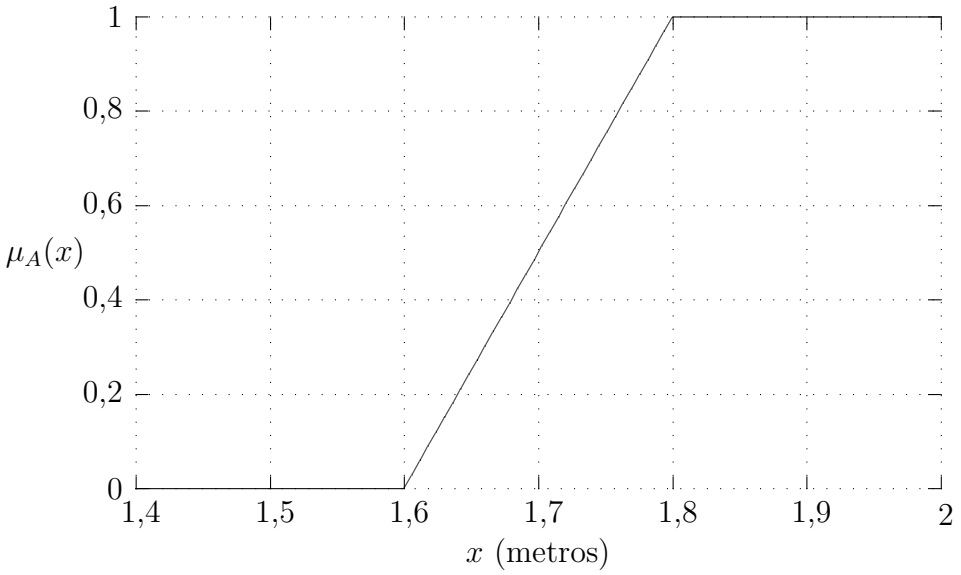

Figura 4: Conjunto nebuloso de pessoas altas

Desta forma, uma pessoa com 1,70 metro não é considerada nem alta e nem baixa, tendo um grau de pertinência de $\frac{1}{2}$ no conjunto de pessoas altas.

2.4.2 Operadores

Os operadores de união, interseção e complemento que operam em conjuntos nebulosos são definidos como:

$$\mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)]$$
$$\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)]$$
$$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$$

Uma forma mais geral de definir os operadores de união e interseção é através das normas S e das normas T, respectivamente. Abaixo encontram-se as propriedades das normas S:

- Comutatividade: $S(a, b) = S(b, a)$;
- Associatividade: $S(a, S(b, c)) = S(S(a, b), c)$;
- Monotonicidade: $a \leq b$ e $c \leq d$ então $S(a, c) \leq S(b, d)$; e
- Coerência: $S(a, 1) = 1$ e $S(a, 0) = a$.

As normas T têm as seguintes propriedades:

- Comutatividade: $T(a, b) = T(b, a)$;
- Associatividade: $T(a, T(b, c)) = T(T(a, b), c)$;
- Monotonicidade: $a \leq b$ e $c \leq d$ então $T(a, c) \leq T(b, d)$; e
- Coerência: $T(a, 1) = a$ e $T(a, 0) = 0$.

Quaisquer família de funções das normas S e T podem ser usadas como operadores de união e interseção. Ainda não existe um consenso em relação a esses operadores. Alguns estudiosos defendem o uso de outras definições, porém min e max são as mais usadas em conjuntos nebulosos.

Usando estes operadores, pode-se derivar outros conjuntos nebulosos a partir do conjunto de pessoas altas. Usando o operador de complemento, o conjunto de pessoas baixas B pode ser definido como $B = \overline{A}$. Pode-se ainda definir o conjunto de pessoas de estatura média M como $M = \overline{A \cup B}$.

2.4.3 Lógica *Fuzzy*

A lógica *fuzzy* de Zadeh é derivada de sua teoria dos conjuntos nebulosos. Nesta lógica, os operadores lógicos "e", "ou" e "não" são derivados da teoria dos conjuntos nebulosos:

$$a \wedge b = \min(a, b)$$
$$a \vee b = \max(a, b)$$
$$\neg a = 1 - a$$

onde a e b podem assumir qualquer valor do intervalo real $[0, 1]$. Assim como na teoria dos conjuntos nebulosos, não existe um consenso em relação a estes operadores.

É importante notar que dizer que $\mu_A(x) = 0,9$, sendo A o conjunto de pessoas altas definido anteriormente, quer dizer que, com certeza, x é uma pessoa alta. Interpretações como existe uma probabilidade de 90% de x ser uma pessoa alta, apesar de freqüentes, são equivocadas em lógica *fuzzy*.

2.4.4 Aplicações

Várias aplicações são desenvolvidas baseadas em lógica nebulosa e técnicas correlatas (Mandani [Mamdani, 1974], TSK [Takagi e Sugeno, 1985; Sugeno e Kang, 1988] e RNA-*Fuzzy* [Jang, 1993; Jang *et al.*, 1997]):

- Processamento de imagens [Bloch, 1994; Saeed *et al.*, 1992; Asgharzadeh, 1996; Grobel e Hienz, 1996];
- Processamento de sinais [Duru *et al.*, 1998; Stegmaier-Stracca e Tschichold-Gürman, 1995];
- Sistemas de navegação [Tunstel, 1995; Tan e Tang, 2000];
- Processamento de informações e de conhecimento [Chau e Yeh, 2000; Subtil *et al.*, 1996; Herrmann *et al.*, 1996];
- Redes neurais [Kulkarni e Muniganti, 1996; Zheng e Kainz, 1999];
- Segurança [Hosmer, 1993]; e
- Sistemas especialistas [Sedbrook, 1998].

Capítulo 3

O Modelo Clouseau

Para classificar e controlar um agente, o Modelo Clouseau insere no sistema formado pelo agente, seu usuário e o ambiente onde atuam um terceiro agente, o inspetor. O modelo deste agente encontra-se na figura 5, dentro do retângulo pontilhado.

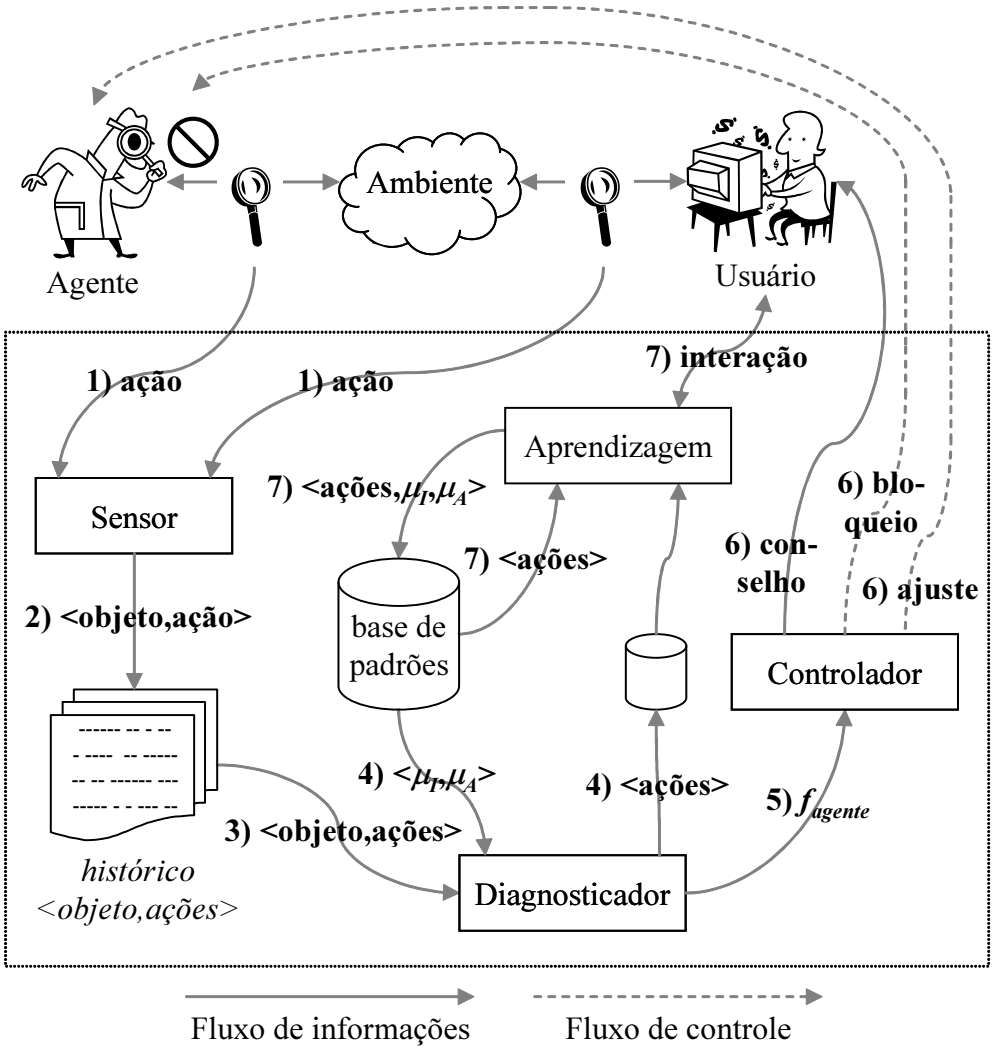

Figura 5: Modelo de agentes inspetores

O inspetor atua no sistema investigando a atuação do agente e do usuário no ambiente de forma transparente do ponto de vista do agente sendo classificado. Usando estas informações, é possível avaliar as ações do agente no ambiente quanto à sua inteligência e autonomia, e controlá-lo segundo esta classificação. Para tanto, os seguintes módulos são definidos no agente inspetor:

- **Sensor**: Captura o tráfego de mensagens entre o usuário, o agente e o ambiente;
- **Histórico de ações**: Armazena o histórico de ações tomadas para cada objeto do ambiente;
- **Base de padrões**: Armazena a inteligência e a autonomia associadas a cada seqüência possível de ações;
- **Diagnosticador**: Diagnostica a atuação do agente de acordo com as ações armazenadas no histórico e as seqüências armazenadas na base de padrões;
- **Controlador**: Controla a autonomia do agente de acordo com sua atuação;
- **Aprendizagem**: Aprende a inteligência e a autonomia de novas seqüências de ações inexistentes na base de padrões.

A dinâmica de classificação e controle do agente ocorre da seguinte forma:

1. Uma ação do agente ou do usuário no ambiente é captada pelo sensor do agente inspetor;
2. O sensor verifica o objeto alvo desta ação e a armazena em um histórico de ações executadas sobre o objeto;

3. O diagnosticador busca no histórico todos os pares $<objeto, ações>$;

4. O diagnosticador busca pelos padrões de ações do histórico na base de padrões, que retorna para cada padrão seus graus de pertinência nos conjuntos nebulosos I e A que representam a inteligência e a autonomia da seqüência de ações, respectivamente; padrões desconhecidos são depositados em uma base para serem processados pelo módulo de aprendizagem;

5. O diagnosticador usa as informações de inteligência e autonomia dos padrões conhecidos para classificar o agente;

6. O controlador, de posse da classificação do agente, controla sua autonomia pedindo ao agente que altere sua própria autonomia, bloqueando ou desbloqueando outras ações do agente no ambiente, ou aconselhando o usuário a configurar seu agente para alterar sua autonomia; e

7. O módulo de aprendizagem busca constantemente por padrões desconhecidos para tentar obter suas informações de inteligência e autonomia, com a ajuda do usuário ou buscando por padrões semelhantes na base de padrões, ou ainda usando uma combinação dos dois. Ao determinar a inteligência e a autonomia de um padrão desconhecido, ele armazena estas informações juntamente com o padrão na base de padrões para uso pelo diagnosticador.

Cada uma das partes que compõem o agente inspetor se encontram detalhadas a seguir.

3.1 Sensor

O sensor captura o tráfego de mensagens entre o usuário, o agente e o ambiente. Estas mensagens contém informações sobre as ações que são realizadas no ambiente, como por exemplo a ação de exclusão de mensagens de correio eletrônico (listagem 1, todas as mensagens encontram-se definidas no apêndice B).

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "DELE messagenum"
  :language RFC1939)
```

Listagem 1: Ação de exclusão de mensagem descrita em FIPA ACL

O sensor tem inteligência para determinar o objeto alvo de cada ação que captura no ambiente. No domínio de aplicação de edição de texto, estes objetos podem ser considerados como a última palavra digitada pelo usuário antes de uma ação do agente para substituí-la por outra, por exemplo. No domínio de aplicação anti-*spam*, os objetos são as mensagens de correio eletrônico, que podem ser identificadas de forma única através de seus atributos `Message-ID`.

Uma vez identificados a ação e seu objeto alvo, o sensor armazena a ação no histórico de ações juntamente com outras ações para este mesmo objeto armazenadas anteriormente.

3.2 Histórico de Ações

Cada ação no ambiente é armazenada pelo sensor no histórico de ações juntamente com a identificação do objeto do ambiente que foi o alvo da ação. Quaisquer ações subseqüentes do agente ou do usuário que tenham como alvo este mesmo objeto serão armazenadas juntamente com a primeira em um vetor, conforme a tabela

2. É importante notar que estas ações podem ocorrer em qualquer tempo, e até mesmo simultaneamente.

Ações	**Vetores de ações**
Usuário faz ação a_1 em objeto A	$A = \{a_1\}$
Usuário faz ação a_2 em objeto A	$A = \{a_1, a_2\}$
Agente faz ação b_1 em objeto B	$A = \{a_1, a_2\}$ $B = \{b_1\}$
Usuário faz ação a_3 em objeto A	$A = \{a_1, a_2, a_3\}$ $B = \{b_1\}$
Usuário faz ação b_2 em objeto B	$A = \{a_1, a_2, a_3\}$ $B = \{b_1, b_2\}$
Agente faz ação c_1 em objeto C	$A = \{a_1, a_2, a_3\}$ $B = \{b_1, b_2\}$ $C = \{c_1\}$

Tabela 2: Atualização do histórico de ações de um agente e de seu usuário

3.3 Base de Padrões

Os padrões de ações representam a atuação do agente no ambiente, juntamente com as reações do usuário em relação à estas ações. Desta forma, pode-se atribuir a cada padrão graus de inteligência e autonomia, que são utilizados, posteriormente, para a classificação e controle do agente em relação à sua atuação como um todo, que é composta por todo o conjunto de atuações realizadas em objetos do ambiente.

Cada padrão de ações p tem associado um par de valores $\mu_I(p)$ e $\mu_A(p)$, que representam a inteligência e a autonomia deste padrão, conforme exemplificado na tabela 3.

Nesta tabela, a representa a ação do agente de acessar uma mensagem de correio

Padrão de ações p	**Significado**	$\mu_I(p)$	$\mu_A(p)$
$\{a\}$	Mensagem acessada pelo agente anti-*spam*	?	?
$\{a, e\}$	Mensagem acessada e excluída do servidor pelo agente	?	1
$\{a, e, l\}$	Mensagem acessada, excluída do servidor e depositada na pasta lixo	?	1
$\{a, e, l, i\}$	Mensagem acessada, excluída, depositada em "lixo" e movida pelo usuário para "inbox"	0	1

Tabela 3: Padrões de ações p na base de padrões e seus respectivos significados e valores de $\mu_I(p)$ e $\mu_A(p)$

eletrônico do servidor, e representa a ação de excluir a mensagem do servidor, l representa a ação de inserir a mensagem na pasta lixo do usuário e i representa a ação do usuário de mover a mensagem para a caixa de entrada.

É importante notar que nem todos os padrões de ações têm valores determinados para as funções de pertinência dos conjuntos I e A. Quando um agente anti-*spam* acessa uma mensagem do servidor, nada pode ser dito quanto à inteligência e à autonomia desta ação. Apenas quando outras ações forem efetuadas sobre esta mesma mensagem é que será possível determinar a inteligência e a autonomia do agente sobre o objeto alvo destas ações.

3.4 Diagnosticador

A cada nova ação armazenada em sua base, o diagnosticador deve buscar por um padrão de ações que indique a autonomia e a inteligência da ação realizada no objeto do ambiente que foi alvo da ação. Como exemplo, um agente de edição de texto que altera incorretamente seqüências de caracteres e as ações do usuário para desfazer as alterações do ambiente podem gerar o histórico de ações representado na tabela 4.

Ações em T	**Vetor de ações em T**
Agente altera texto T	$T = \{a\}$
Usuário desfaz ação a no texto T	$T = \{a, d\}$
Agente altera texto T	$T = \{a, d, a\}$
Usuário desfaz ação a no texto T	$T = \{a, d, a, d\}$

Tabela 4: Atualização do histórico de ações de um agente e de seu usuário em uma aplicação de edição de texto

Nesta tabela, T é o texto que é alterado pelo agente, a é uma ação de alteração deste texto e d é uma ação do usuário que desfaz a ação do agente. Se a base de padrões do inspetor que classifica e controla este agente for como o ilustrado na tabela 5, a inteligência e a autonomia do conjunto de ações $\{a, d, a, d\}$ serão zero e 1, respectivamente.

Buscando por padrões de ações na base de padrões e seus graus de pertinência nos conjuntos nebulosos I e A, o diagnosticador pode calcular a inteligência $\mu_I(x)$ e a autonomia $\mu_A(x)$ de um agente x levando em conta todo seu conjunto de atuações, e a partir destes valores avaliar sua classificação.

Padrão de ações p	$\mu_I(p)$	$\mu_A(p)$
$\{a, d, a, d\}$	0	1
$\{a, a, a\}$	1	1

Tabela 5: Base de padrões de ações de um inspetor que classifica e controla um agente de edição de texto

A classificação de agentes no Modelo Clouseau é similar aos modelos de Montgomery e Cipolla, classificando agentes de acordo com sua inteligência e autonomia, e observando as conseqüências de suas ações ao invés de usar algum conceito abstrato para classificá-los. Além disso, os agentes são classificados de acordo com o ponto de vista de seus usuários, o que significa que não é preciso levar em conta a vantagem ou prejuízo que o agente pode gerar para si ou para outros agentes. Isto leva a um modelo diferente, com eixos diferentes que classificam agentes de acordo com sua inteligência e autonomia.

Agentes no Modelo Clouseau, assim como no modelo de Montgomery, são identificados através de regras simples, conforme ilustrado na tabela 6.

Tipo de agente	**Inteligência**	**Autonomia**
Proficiente	alta	baixa
Herói	alta	alta
Perigoso	baixa	baixa
Clouseau	baixa	alta

Tabela 6: Tipos de agentes do Modelo Clouseau e seus graus de inteligência e autonomia

Os quatro tipos de agentes do Modelo Clouseau correspondentes aos quatro

tipos de soldados de Montgomery estão listados abaixo:

1. **Proficiente** (tipo I): Agentes inteligentes, mas como têm baixa autonomia não são realmente úteis aos seus usuários, pois os interrompem a cada ação que desejam tomar para verificar se o usuário está de acordo;

2. **Herói** (tipo II): Agentes inteligentes e com autonomia, o que os torna muito úteis para seus usuários, tomando ações corretas sem precisar consultar o usuário para receber permissão para executá-las;

3. **Perigoso** (tipo III): Agentes estúpidos, mas como têm baixa autonomia não chegam a causar transtornos aos seus usuários, pois antes de executar ações desastrosas incomodam seus usuários em busca de aprovação; e

4. **Clouseau** (tipo IV): Agentes estúpidos e munidos de autonomia, o que na prática significa que tomarão ações desastrosas em nome de seus usuários sem buscar autorização para tal. Estes agentes podem causar grandes estragos, incluindo financeiros e de credibilidade.

O Modelo Clouseau muda os eixos do modelo de Cipolla pelos eixos "inteligência" e "autonomia", como no modelo de Montgomery, e também os nomes dos tipos de agentes para melhor refletir cada par de valores de inteligência e autonomia (figura 6).

Através das regras da tabela 6 é possível formalizar a classificação de cada tipo de agente do Modelo Clouseau. As equações para computar os graus de pertinência para cada tipo de agente são:

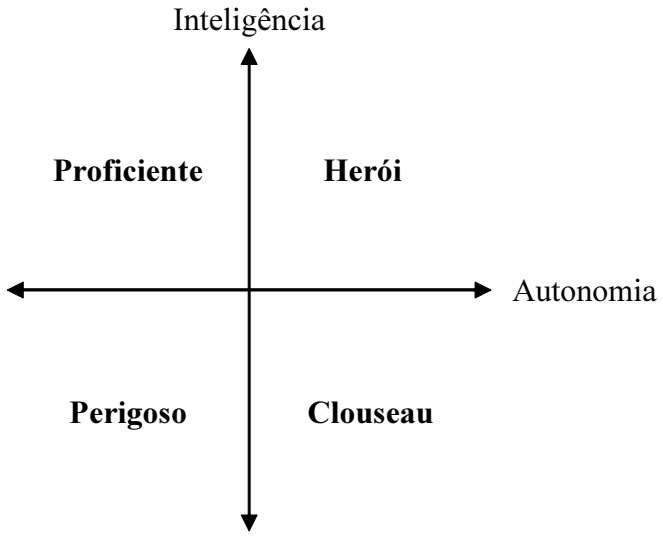

Figura 6: Modelos de Montgomery e Cipolla adaptados para a classificação de agentes

$$\mu_P(x) = \mu_I(x) \wedge \neg\mu_A(x)$$
$$\mu_H(x) = \mu_I(x) \wedge \mu_A(x)$$
$$\mu_p(x) = \neg\mu_I(x) \wedge \neg\mu_A(x)$$
$$\mu_C(x) = \neg\mu_I(x) \wedge \mu_A(x)$$

onde $\mu_P(x)$, $\mu_H(x)$, $\mu_p(x)$ e $\mu_C(x)$ são as funções de pertinência do agente x nos conjuntos nebulosos de agentes to tipo proficiente, herói, perigoso e Clouseau, respectivamente. A figura 7 ilustra os conjuntos nebulosos P, H, p e C em relação às funções de pertinência $\mu_I(x)$ e $\mu_A(x)$.

Portanto, para classificar um agente x basta computar o grau de pertinência de x nos conjuntos nebulosos referentes a cada um dos tipos de agentes, usando $\mu_I(x)$ e $\mu_A(x)$ previamente computados pelo diagnosticador. Pode-se perceber que o resultado não é uma classificação única para o agente x, mas que x apresenta

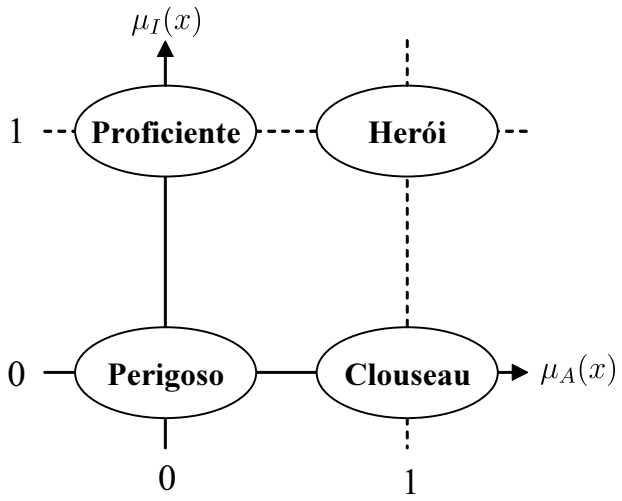

Figura 7: Conjuntos nebulosos P, H, p e C em relação às funções de pertinência $\mu_I(x)$ e $\mu_A(x)$

uma mistura de comportamentos de cada tipo de agente, da mesma forma que o modelo de Cipolla.

3.5 Controlador

Tendo a classificação do agente, o inspetor pode controlar automaticamente as ações do agentes de forma a evitar prejuízos aos usuários advindos de agentes do tipo Clouseau.

A autonomia de um agente x deve ser ajustada de acordo com o valor de $\mu_C(x)$. Quanto maior a pertinência do agente no conjunto dos agentes do tipo Clouseau, maior o controle que deve ser exercido sobre o agente, diminuindo sua autonomia para evitar ações desastrosas. Da mesma forma, quanto menor esta pertinência, menos controle devemos exercer sobre o agente para que ele possa executar suas tarefas.

A autonomia do agente é sua capacidade de agir sem o consentimento do usuário. Para controlar esta autonomia, o controlador pode agir de três formas:

1. **Contatar o usuário**: O controlador contata o usuário e o informa de que seu agente deve ter sua autonomia alterada; o usuário é responsável por executar esta ação;

2. **Contatar o agente**: O controlador contata o agente diretamente e pede à ele que altere sua autonomia; e

3. **Bloquear outras ações**: O controlador bloqueia outras ações do agente no ambiente, ou retira o bloqueio destas ações.

Na terceira opção, é necessário interceptar mensagens de ações desastrosas que os agentes enviam para o ambiente e decidir se a mensagem será enviada para o ambiente ou se será retida pelo controlador, de acordo com o controle que deve ser exercido sobre o agente.

Uma vez que o inspetor de agentes intercepta todas as interações do agente para medir e controlar sua atuação, pode-se perceber que ele é na verdade um invólucro para o agente, isolando-o do contato direto com o ambiente e com o usuário, como ilustrado na figura 8.

3.6 Aprendizagem

O módulo de aprendizagem é responsável por descobrir a inteligência e a autonomia associadas a padrões de ações enviadas pelo diagnosticador que ainda não se encontram na base de padrões.

Esta tarefa pode ser realizada de forma automática, buscando na base de padrões por outros padrões semelhantes e decidindo o que mais se aproxima do

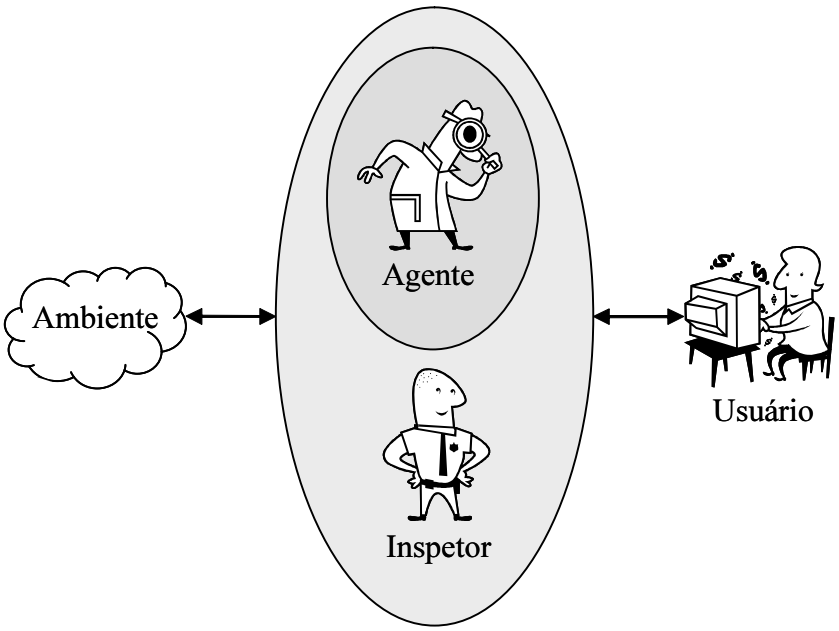

Figura 8: Controlador de agentes como um *wrapper*

novo padrão, ou pedindo ao usuário que informe a inteligência e autonomia para este novo padrão.

Uma combinação dos dois também é possível, se não for encontrado nenhum padrão existente que se aproxime suficientemente do novo padrão, então o usuário será contatado para informar a inteligência e a autonomia para o novo padrão.

Capítulo 4

Experimento

Para testar o Modelo Clouseau, foi escolhido o domínio de aplicação de filtro de mensagens de correio eletrônico (filtro anti-*spam*), e foi desenvolvido um sistema que emula um centro de atendimento a clientes de uma empresa, responsável por responder mensagens de correio eletrônico de clientes com requerimentos de suporte. O sistema é ilustrado na figura 9.

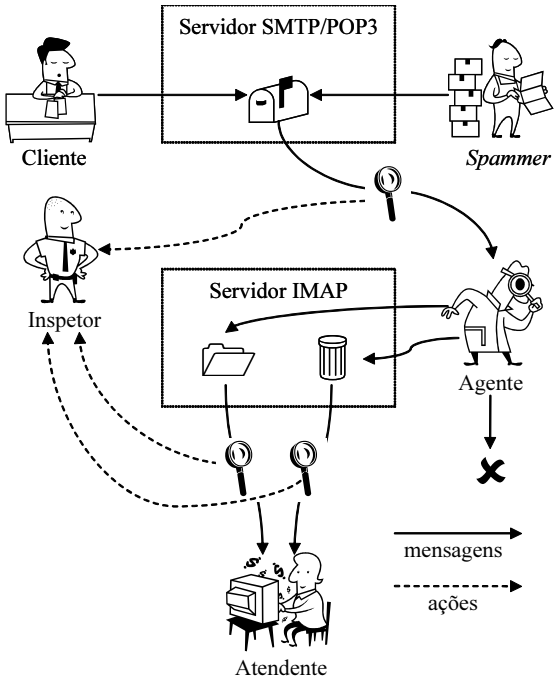

Figura 9: Diagrama do sistema de teste do Modelo Clouseau

Neste diagrama pode-se perceber os seguintes módulos:

- Um cliente que envia mensagens com requerimentos de suporte de tempos em tempos ao atendente;

- Um *spammer* que envia mensagens *spam* de tempos em tempos para o atendente;
- Um agente anti-*spam* com inteligência e autonomia configuráveis que retira mensagens da caixa de correio do atendente e as filtra, movendo mensagens *spam* para a caixa de entrada, para a pasta lixo ou deletando-as;
- Um inspetor que implementa o Modelo Clouseau e captura interações no sistema para classificar e controlar o agente;
- Um servidor SMTP/POP3, no qual as mensagens do cliente e do atendente são armazenadas;
- Um servidor IMAP, no qual o agente deposita as mensagens filtradas do correio do atendente; e
- Um programa leitor de correio eletrônico, no qual o atendente pode ler, responder, excluir e mover as mensagens recebidas.

No diagrama, as setas com linhas cheias indicam o fluxo de mensagens de correio eletrônico pelo sistema. As mensagens são originadas pelos *spammer* e cliente, que as depositam na caixa de correio do atendente. O agente anti-*spam*, ao perceber a existência de novas mensagens nesta caixa, as busca, classifica, e deposita na caixa de entrada ou na pasta lixo do servidor IMAP, ou as exclui. O leitor de correio busca mensagens no servidor IMAP, e o atendente pode então tomar as ações que julgar necessárias para com elas.

As lupas no diagrama representam pontos de coleta de informações do inspetor. As setas com linhas pontilhadas indicam o fluxo de informações dos pontos de

coleta até o inspetor, que terá então condições de julgar a autonomia e a inteligência do agente de acordo com suas ações e do atendente no ambiente.

4.1 O Cliente

O cliente emula pessoas enviando mensagens com requerimentos de suporte para o atendente de tempos em tempos. Cada mensagem contém uma expressão algébrica de dificuldade variável. O atendente deve calcular o resultado da equação utilizando uma calculadora eletrônica de quatro operações e enviar ao cliente a resposta correta.

As equações são geradas aleatoriamente em tempo de execução, tomando-se precaução para que nenhum resultado, intermediário ou final, ultrapasse dez algarismos ou tenha quaisquer algarismos após o ponto decimal. O objetivo do cálculo do resultado da equação é emular o tempo de pesquisa que se faz necessária para atender requerimentos de suporte.

Mensagens originadas pelo cliente não devem ser classificadas como *spam* por um agente proficiente ou herói.

4.2 O *Spammer*

O *spammer* emula pessoas enviando mensagens *spam* para o atendente de tempos em tempos. Estas mensagens são marcadas com o cabeçalho `X-Is-Spam: true`, compatível com o formato de mensagens de correio eletrônico [Resnick, 2001], de forma a facilitar a programação de agentes inteligentes.

As mensagens *spam* devem ser classificadas como tal por um agente proficiente ou herói.

4.3 O Agente Anti-*Spam*

O agente anti-*spam* filtra mensagens baseado em um filtro configurável chamado *action*. Este filtro segue a gramática descrita na listagem 17 do apêndice C, e tem disponíveis duas funções que podem ser usadas em sua definição:

- `random()`: Retorna um número real aleatório contido no intervalo $[0, 1)$; e
- `header(`*`cabeçalho`*`)`: Retorna o valor de *`cabeçalho`* como uma *string*, ou `null` caso *`cabeçalho`* não exista na mensagem.

O filtro *action* do agente deve retornar uma *string* dentre as listadas abaixo, indicando qual ação deve ser tomada para a mensagem sendo avaliada:

- `"inbox"`: Indica que a mensagem deve ser deixada na caixa de entrada do atendente;
- `"trash"`: Indica que a mensagem deve ser movida para a pasta lixo do atendente; e
- `"delete"`: Indica que a mensagem deve ser excluída.

Através da correta configuração deste filtro, é possível simular agentes anti-*spam* com diversos comportamentos. Como exemplo, um agente herói para a massa de mensagens de teste utilizada nos experimentos pode ter seu filtro definido como `header("X-Is-Spam") == "true" ? "delete" : "inbox"`.

4.4 O Agente Inspetor

O agente inspetor implementa o Modelo Clouseau para classificar e controlar agentes. Ele captura todo o tráfego de rede entre o agente e a caixa de correio do

atendente (um servidor POP3 [ArGo, 2004]) e entre o atendente e sua caixa de correio filtrada (um servidor IMAP [Harris, 2003]).

Observando as ações tomadas pelo agente e pelo atendente, ele calcula $\mu_I(x)$ e $\mu_A(x)$, e usa estes valores para calcular o grau de pertinência do agente em cada um dos conjuntos nebulosos que representam os diferentes tipos de agentes do Modelo Clouseau.

Para cada mensagem, o inspetor computa a autonomia do agente verificando em qual pasta a mensagem foi depositada, ou se a mensagem foi excluída, e a inteligência verificando as ações do atendente em relação à mensagem, observando se o agente tomou a ação correta ou se esta foi desfeita[2]. A tabela 7 mostra as combinações entre as possíveis ações do agente e do atendente e os graus de inteligência e autonomia relacionados às combinações.

	Movida para lixo		**Excluída**	
	Inteligência	Autonomia	Inteligência	Autonomia
Ação confirmada	1	0	1	1
Ação desfeita	0	0	0	1

Tabela 7: Graus de inteligência e autonomia associados às ações do agente e do atendente em relação às mensagens

Mensagens deixadas na caixa de entrada do atendente não são levadas em conta para a classificação do agente, pois neste caso não é possível determinar com clareza, sem dados adicionais sobre o agente, os graus de inteligência e autonomia

[2]Quando o agente exclui uma mensagem, o inspetor associa um alto grau de autonomia ao agente, porém faz uma cópia da mensagem excluída na pasta lixo para que possa verificar a inteligência do agente conforme as ações do atendente em relação à mensagem.

do agente, pois mensagens podem ser deixadas na caixa de entrada pelo agente porque:

- Tratam-se de *spam* e o agente classificou erroneamente as mensagens como não-*spam* (estúpido com autonomia indeterminada) ou o agente classificou corretamente as mensagens mas não tinha autonomia suficiente para movê-las para lixo ou excluí-las (inteligente sem autonomia); ou
- Não se tratam de *spam* e o agente classificou corretamente as mensagens (inteligente com autonomia indeterminada) ou o agente classificou erroneamente as mensagens mas não tinha autonomia suficiente para movê-las para lixo ou excluí-las (estúpido sem autonomia).

4.5 Testes Realizados

Três experimentos foram conduzidos para testar o Modelo Clouseau:

- **Teste com agentes reais**: Verifica o comportamento do sistema com um agente anti-*spam* real baixado da Internet enquanto este é treinado para classificar mensagens;
- **Teste de extremos**: Verifica se o sistema classifica corretamente os agentes que pertencem exclusivamente a apenas um dos tipos de agente do Modelo Clouseau; e
- **Teste de inteligência e de autonomia**: Verifica se o sistema avalia corretamente a inteligência do agente quando esta varia de zero a 1 e a autonomia é fixada, e vice-versa.

4.5.1 Testes com Agentes Reais

Neste experimento, uma massa real de 132 mensagens divididas em seis grupos de acordo com a tabela 8 é utilizada para classificar os agentes anti-*spam* POPFile [Graham-Cumming, 2004] e SpamBayes [Peters *et al.*, 2005], baixados da Internet. Estes agentes devem ser treinados para que possam classificar corretamente mensagens de correio eletrônico que interceptem. Após cada grupo de mensagens, o atendente informa ao agente quais mensagens do grupo foram classificadas corretamente e quais erroneamente.

Grupo	1	2	3	4	5	6
Número de mensagens	38	13	22	14	30	15

Tabela 8: Quantidade de mensagens por grupo para o teste com agentes reais

Os agentes tomam o lugar do filtro *action* do agente dos testes anteriores, mas o restante do sistema permanece o mesmo, e têm liberdade para deixar mensagens na caixa de entrada, movê-las para a pasta lixo e excluí-las[3]. As mensagens que os agentes não conseguirem classificar serão deixadas na caixa de entrada.

Para cada agente, um gráfico mostrando sua porcentagem de acerto acumulada e seis gráficos mostrando a evolução das funções de pertinência dos conjuntos nebulosos I, A, P, H, p e C para cada ação executada no ambiente são mostrados.

Agente POPFile

O agente POPFile é um agente escrito em Perl que atua como um *proxy* POP3 que se insere entre o programa leitor de correio e o servidor POP3 do usuário,

[3]Porém neste caso o inspetor irá fazer uma cópia da mensagem na pasta lixo, conforme explicado anteriormente.

analisando cada mensagem presente no servidor e inserindo nelas informações que podem ser usadas pelo programa leitor de correio para tomar as ações necessárias de acordo com as classificações, como mover mensagens para pastas específicas ou excluí-las.

Os seis gráficos seguintes ilustram a evolução das funções de pertinência em cada conjunto nebuloso conforme ações, tanto do agente quanto do usuário, em um total de 1.479, ocorrem no ambiente:

- $\mu_I(x)$: A inteligência do agente medida pelo inspetor (figura 10);
- $\mu_A(x)$: A autonomia do agente medida pelo inspetor (figura 11);
- $\mu_P(x)$: A função de pertinência do agente no conjunto dos agentes proficientes, calculada com o uso de $\mu_I(x)$ e $\mu_A(x)$ (figura 12);
- $\mu_H(x)$: A função de pertinência do agente no conjunto dos agentes heróis (figura 13);
- $\mu_p(x)$: A função de pertinência do agente no conjunto dos agentes perigosos (figura 14); e
- $\mu_C(x)$: A função de pertinência do agente no conjunto dos agentes Clouseau (figura 15).

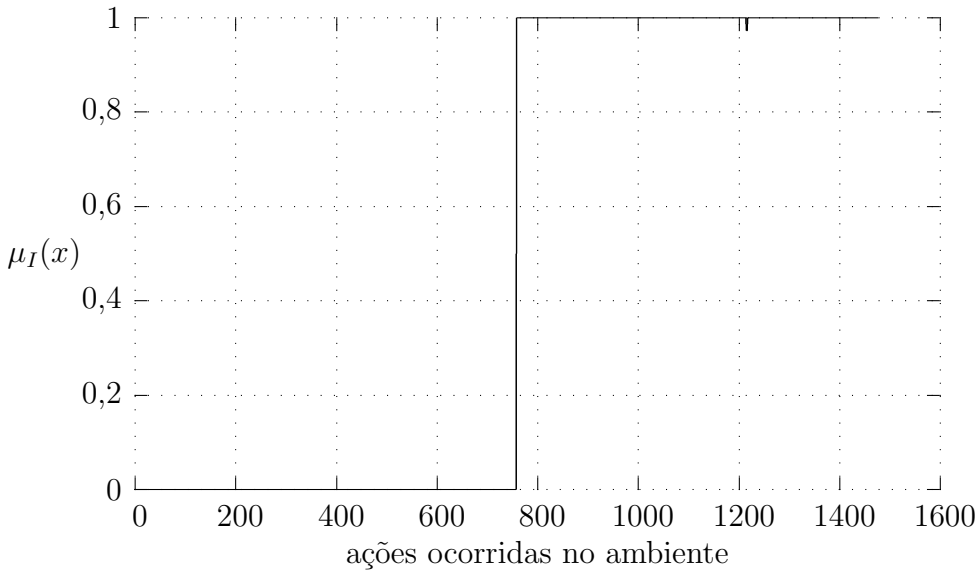

Figura 10: Inteligência medida pelo inspetor no teste do agente POPFile

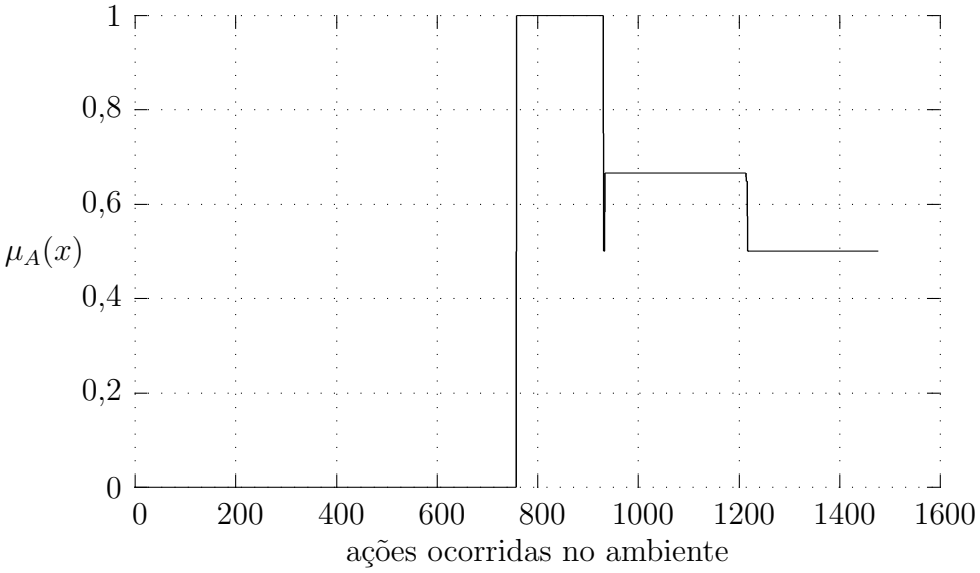

Figura 11: Autonomia medida pelo inspetor no teste do agente POPFile

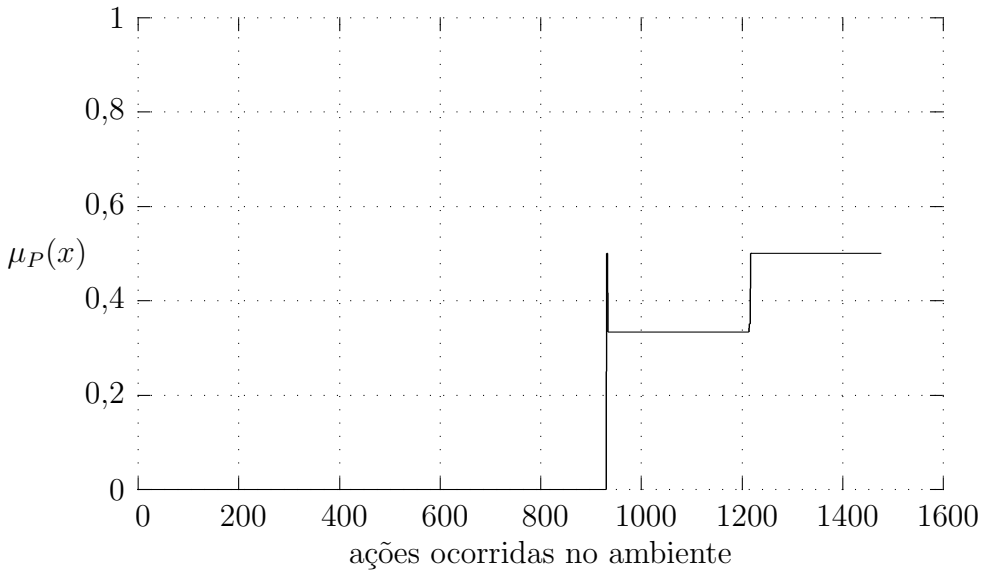

Figura 12: $\mu_P(x)$ medida pelo inspetor no teste do agente POPFile

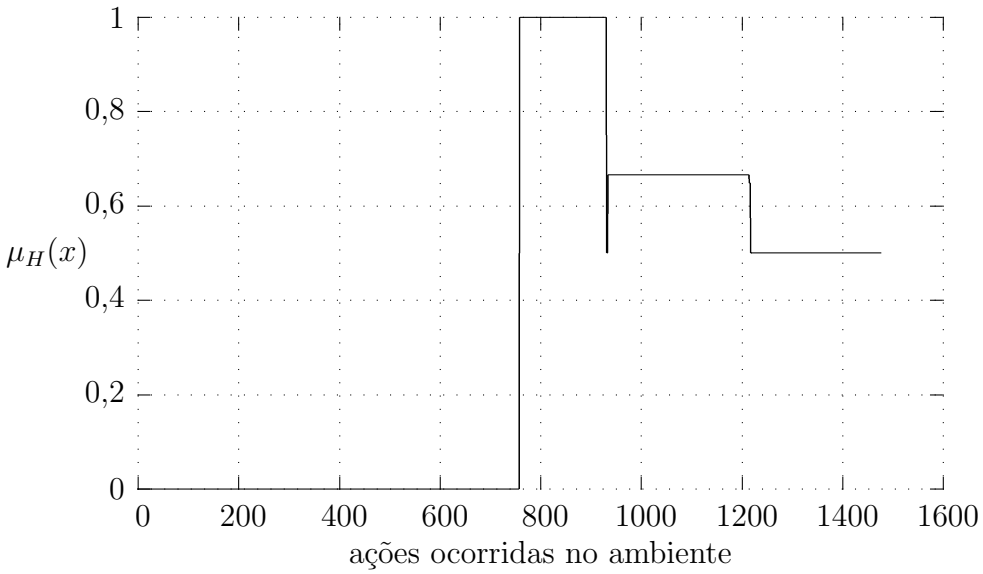

Figura 13: $\mu_H(x)$ medida pelo inspetor no teste do agente POPFile

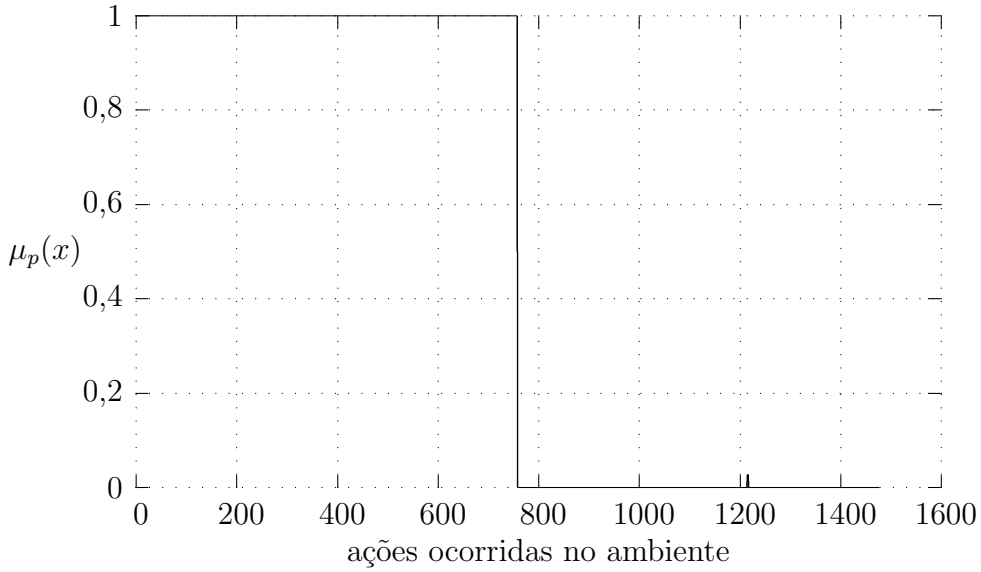

Figura 14: $\mu_p(x)$ medida pelo inspetor no teste do agente POPFile

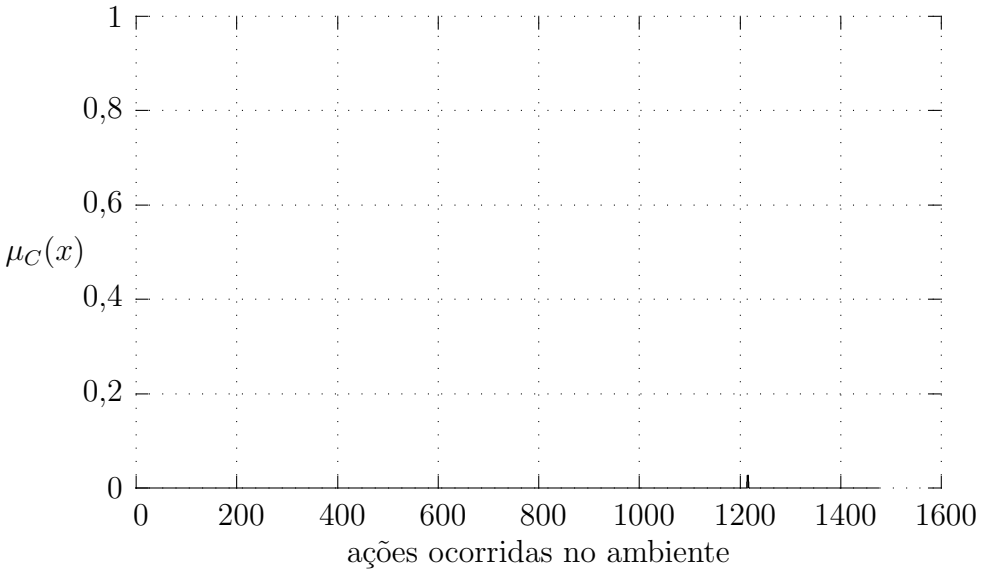

Figura 15: $\mu_C(x)$ medida pelo inspetor no teste do agente POPFile

É possível verificar através dos gráficos que o agente POPFile não foi caracterizado como do tipo Clouseau em nenhum momento durante o teste. Isto é

devido ao fato deste agente ser conservador, deixando na caixa de entrada todas as mensagens que não consegue classificar.

Como inicialmente todas as mensagens são deixadas na caixa de entrada, o agente é classificado nesta fase como perigoso. Quando, após um determinado tempo, passa a classificar mensagens de acordo com o treinamento recebido, ele deixa de ser perigoso e passa a ser herói, acertando a classificação de todas as mensagens.

Porém, com a queda da autonomia ocasionada por novas mensagens que deveriam ser excluídas mas são deixadas na pasta lixo, o agente passa a ser proficiente e herói ao mesmo tempo, com funções de pertinência $\mu_P(x) = 0,5$ e $\mu_H(x) = 0,5$, respectivamente. Este quadro se mantém estável até o final do teste.

É razoável supor que com o uso continuado, este agente fosse classificado como herói, pois com o treinamento contínuo ele acertaria cada vez mais classificações, tomando, portanto, a ação de excluir mensagens *spam*.

Agente SpamBayes

O agente SpamBayes é um agente escrito em Python que atua como um *proxy* POP3 exatamente como o POPFile, se inserindo entre o programa leitor de correio e o servidor POP3 do usuário, analisando cada mensagem presente no servidor e inserindo nelas informações que podem ser usadas pelo programa leitor de correio para tomar as ações necessárias de acordo com as classificações, como mover mensagens para pastas específicas ou excluí-las.

Os seis gráficos seguintes ilustram a evolução das funções de pertinência em cada conjunto nebuloso conforme ações, tanto do agente quanto do usuário, em um total de 1.375, ocorrem no ambiente:

- $\mu_I(x)$: A inteligência do agente medida pelo inspetor (figura 16);
- $\mu_A(x)$: A autonomia do agente medida pelo inspetor (figura 17);
- $\mu_P(x)$: A função de pertinência do agente no conjunto dos agentes proficientes, calculada com o uso de $\mu_I(x)$ e $\mu_A(x)$ (figura 18);
- $\mu_H(x)$: A função de pertinência do agente no conjunto dos agentes heróis (figura 19);
- $\mu_p(x)$: A função de pertinência do agente no conjunto dos agentes perigosos (figura 20); e
- $\mu_C(x)$: A função de pertinência do agente no conjunto dos agentes Clouseau (figura 21).

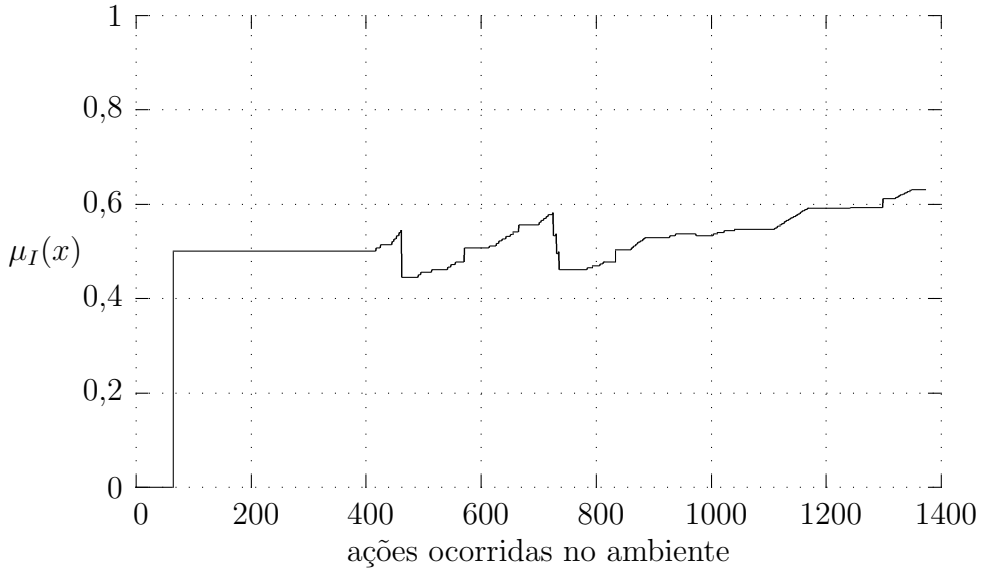

Figura 16: Inteligência medida pelo inspetor no teste do agente SpamBayes

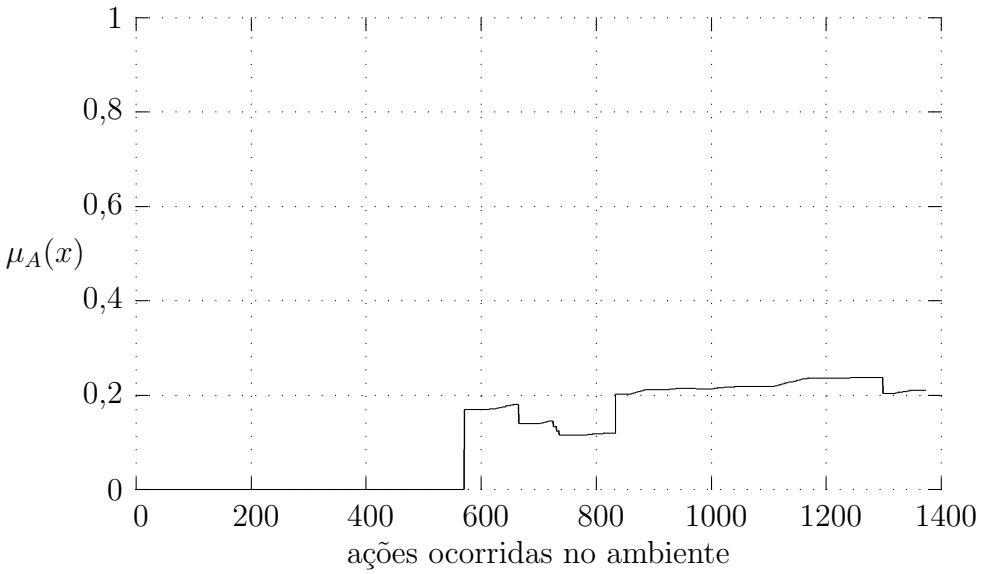

Figura 17: Autonomia medida pelo inspetor no teste do agente SpamBayes

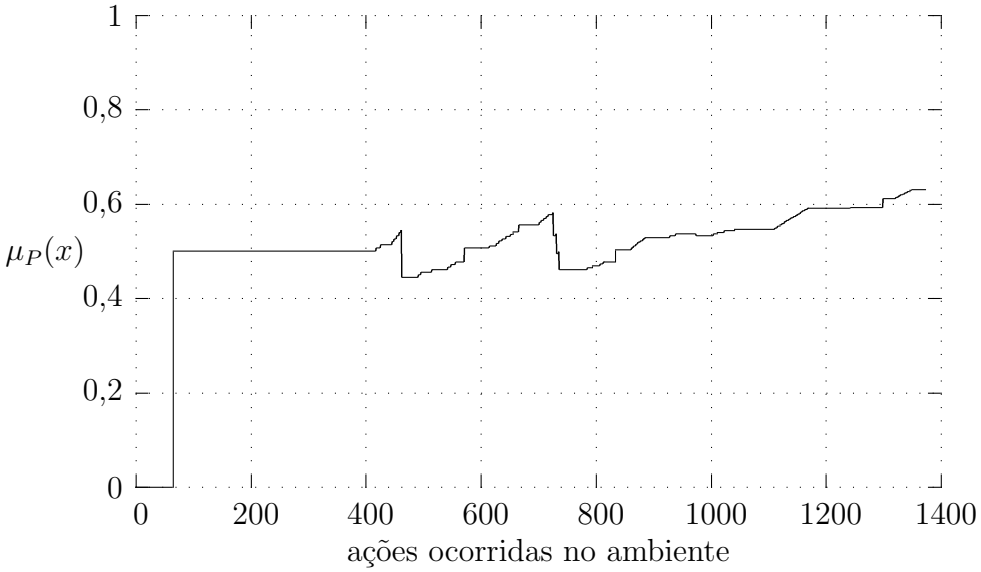

Figura 18: $\mu_P(x)$ medida pelo inspetor no teste do agente SpamBayes

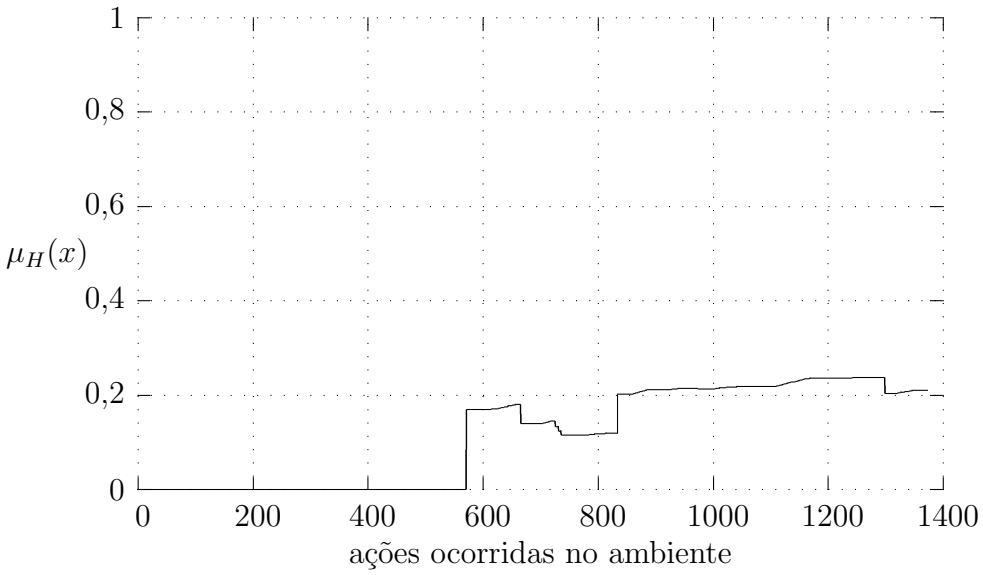

Figura 19: $\mu_H(x)$ medida pelo inspetor no teste do agente SpamBayes

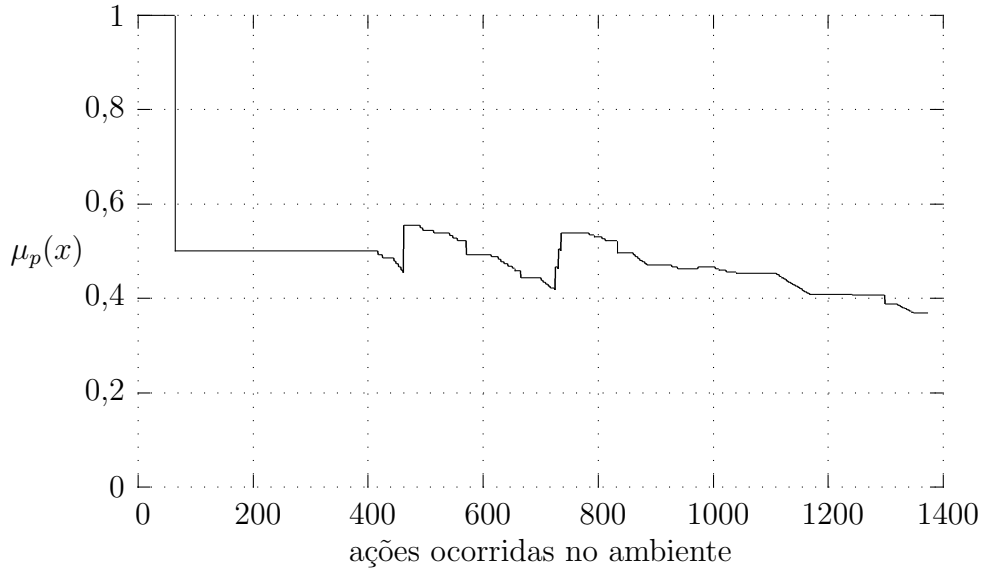

Figura 20: $\mu_p(x)$ medida pelo inspetor no teste do agente SpamBayes

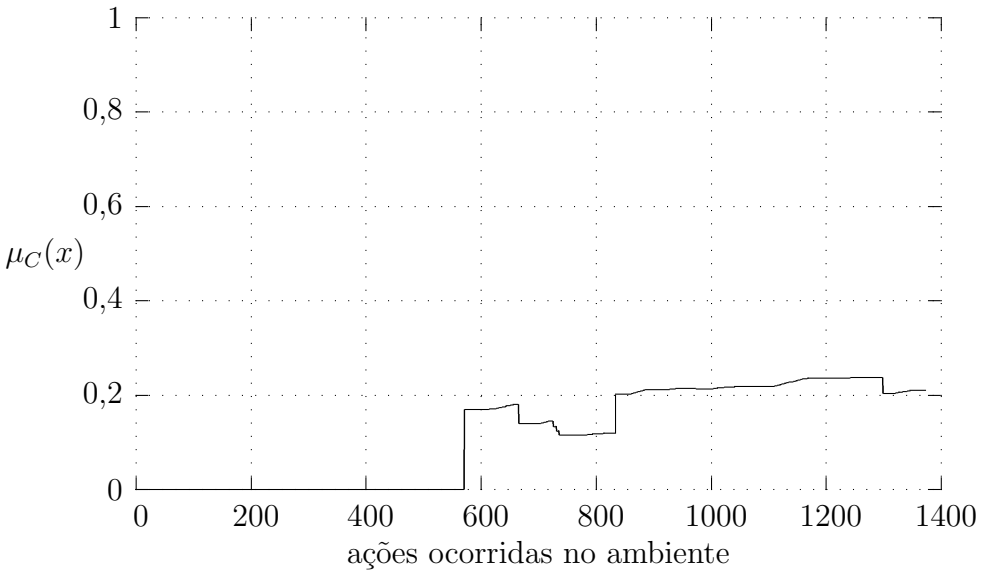

Figura 21: $\mu_C(x)$ medida pelo inspetor no teste do agente SpamBayes

É possível verificar através dos gráficos que o agente SpamBayes teve uma atuação pior do que o agente POPFile. Sua inteligência e autonomia ao fim do teste foram de apenas 0,63 e 0,21, contra 1 e 0,5 do POPFile. Porém, pela curva que os gráficos apresentam para o SpamBayes, é razoável supor que sua atuação melhore com o uso contínuo.

Pelos gráficos também é possível verificar que o agente POPFile assimilava rapidamente o treinamento recebido entre os grupos de mensagens, devido aos "saltos" bruscos nas evoluções das funções de pertinência após estes treinamentos, enquanto o SpamBayes, por sua vez, assimilava o treinamento de forma mais progressiva. Portanto, apesar de sua melhor classificação, o agente POPFile é mais sensível aos treinamentos efetuados, e deve-se tomar cuidado para que ele não mude seu comportamento bruscamente e seja classificado como um agente do tipo Clouseau.

A tabela 9 resume os valores das funções de pertinência ao final do teste para ambos os agentes.

	POPFile	**SpamBayes**
$\mu_I(x)$	1,0	0,631578947
$\mu_A(x)$	0,5	0,210526316
$\mu_P(x)$	0,5	0,631578947
$\mu_H(x)$	0,5	0,210526316
$\mu_p(x)$	0,0	0,368421053
$\mu_C(x)$	0,0	0,210526316

Tabela 9: Graus de pertinência dos agentes POPFile e SpamBayes nos conjuntos I, A, P, H, p e C ao fim dos testes

4.5.2 Testes de Extremos

Neste experimento, quatro agentes correspondendo aos tipos de agentes do Modelo Clouseau (tabela 10) foram testados para verificar a habilidade do sistema de classificar cada um dos tipos de agentes do modelo em seus casos extremos.

Tipo de agente	**Filtro utilizado**
Proficiente	`header("X-Is-Spam") == "true" ? "trash" : "inbox"`
Herói	`header("X-Is-Spam") == "true" ? "delete" : "inbox"`
Perigoso	`header("X-Is-Spam") != "true" ? "trash" : "inbox"`
Clouseau	`header("X-Is-Spam") != "true" ? "delete" : "inbox"`

Tabela 10: Tipos de agentes do Modelo Clousau e suas configurações de filtro

O experimento foi executado uma vez para cada tipo de agente. Cada execução durou 30 minutos, durante os quais 20 mensagens de clientes e dez de *spammers* foram enviadas e classificadas. Durante a execução, o inspetor grava um arquivo

em disco com informações sobre a classificação do agente a cada interação ocorrida no sistema.

As informações gravadas pelo inspetor são usadas para desenhar gráficos de $\mu_I(x)$ e $\mu_A(x)$ em função do tempo[4], e para cada tipo de agente testado foram avaliadas, ao final do experimento, as funções de pertinência em cada um dos tipos de agente definidos no Modelo Clouseau.

Agente do Tipo Proficiente

Quando o agente foi configurado como proficiente, o inspetor convergiu, após aproximadamente 9% das ações do agente e do atendente haverem sido tomadas, a inteligência e a autonomia do agente para exatamente 1 e zero, respectivamente (figura 22)[5].

A inteligência converge para 1 em decorrência do filtro usado pelo agente, que sempre acerta a classificação das mensagens. A autonomia converge para zero devido ao fato de que o agente nunca exclui mensagens da caixa do usuário.

Os graus de pertinência para cada tipo de agente do Modelo Clouseau, ao final deste teste, são $\mu_P(x) = 1$, $\mu_H(x) = 0$, $\mu_p(x) = 0$ e $\mu_C(x) = 0$, mostrando claramente que o agente foi corretamente classificado como sendo do tipo proficiente.

Os gráficos de evolução de classificação dos outros três tipos de agentes não serão incluídos, por serem extremamente semelhantes. Porém, os valores de suas

[4]Nos gráficos, o eixo horizontal não representa tempo, mas a contagem de interações ocorridas no sistema, que podem acontecer em qualquer momento durante o teste. Porém, como as interações ocorrem ordenadamente, uma após a outra, os gráficos representam a evolução da classificação do agente durante a execução do teste.

[5]O número de ações tomadas no ambiente é maior que o número de mensagens, uma vez que para cada mensagem podem haver mais de uma ação, como mover para outra pasta e responder, por exemplo.

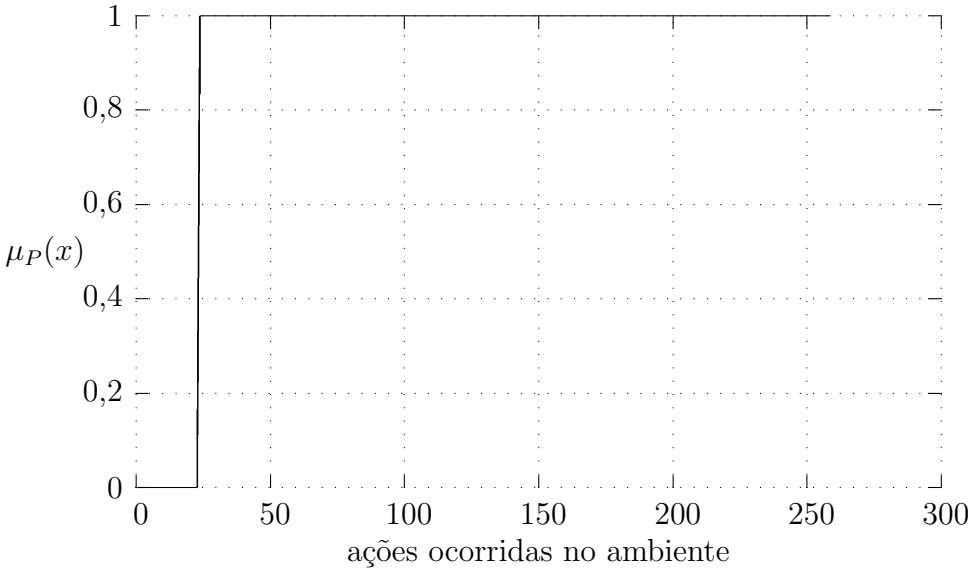

Figura 22: Evolução da classificação de um agente proficiente

funções de pertinência ao final da classificação encontram-se a seguir.

Agente do Tipo Herói

Quando o agente foi configurado como herói, o inspetor convergiu, após aproximadamente 11% das ações haverem sido tomadas, ambas a inteligência e a autonomia do agente para exatamente 1.

A inteligência converge para 1 também em decorrência do filtro usado pelo agente. A autonomia converge para 1 devido ao fato de que o agente exclui todas as mensagens consideradas *spam* da caixa de entrada do atendente.

Os graus de pertinência para cada tipo de agente do Modelo Clouseau, ao final deste teste, são $\mu_P(x) = 0$, $\mu_H(x) = 1$, $\mu_p(x) = 0$ e $\mu_C(x) = 0$, mostrando claramente que o agente foi corretamente classificado como sendo do tipo herói.

Agente do Tipo Perigoso

Quando o agente foi configurado como perigoso, o inspetor manteve ambas a inteligência e a autonomia do agente em zero, mesmo após todas as ações haverem sido tomadas.

A inteligência é mantida em zero em decorrência do filtro usado pelo agente, que sempre erra a classificação das mensagens. A autonomia converge para zero devido ao fato de que o agente nunca exclui mensagens da caixa do usuário, exatamente como no caso do agente proficiente.

Os graus de pertinência para cada tipo de agente do Modelo Clouseau, ao final deste teste, são $\mu_P(x) = 0$, $\mu_H(x) = 0$, $\mu_p(x) = 1$ e $\mu_C(x) = 0$, mostrando claramente que o agente foi corretamente classificado como sendo do tipo perigoso.

Agente do Tipo Clouseau

Quando o agente foi configurado como Clouseau, o inspetor convergiu, após aproximadamente 4% das ações do agente e do atendente haverem sido tomadas, a inteligência e a autonomia do agente para exatamente zero e 1, respectivamente.

A inteligência converge para zero também em decorrência do filtro usado pelo agente. A autonomia converge para 1 devido ao fato de que o agente exclui todas as mensagens consideradas *spam* da caixa de entrada do atendente.

Os graus de pertinência para cada tipo de agente do Modelo Clouseau, ao final deste teste, são $\mu_P(x) = 0$, $\mu_H(x) = 0$, $\mu_p(x) = 0$ e $\mu_C(x) = 1$, mostrando claramente que o agente foi corretamente classificado como sendo do tipo Clouseau.

Conclusões

O modelo classifica corretamente os tipos de agente do Modelo Clouseau quando ocorrem em seus casos extremos, ou seja, quando uma de suas funções de pertinência resulta em 1.

4.5.3 Testes de Inteligência e Autonomia

Teste de Inteligência

Neste experimento a autonomia do agente foi fixada em 1 enquanto sua inteligência foi variada de zero a 1 em intervalos de $0,1$, através da configuração do filtro *action* do agente. O objetivo é verificar a capacidade do sistema de perceber diferentes graus de inteligência do agente.

Seja i a inteligência desejada para o agente, seu filtro pode ser configurado para acertar a classificação de $(i \times 100)\%$ das mensagens através da seguinte expressão

```
header("X-Is-Spam") == "true" ?
(random() < i ? "delete" : "inbox") :
(random() < i ? "inbox" : "delete")
```

Para cada valor de i, 40 mensagens de clientes e 20 de *spammers* foram enviadas. A tabela 11 mostra os valores da inteligência do agente esperada e medida pelo inspetor.

Medida	0,000	0,171	0,158	0,344	0,467	0,410
Esperada	0,000	0,100	0,200	0,300	0,400	0,500
Medida		0,571	0,737	0,714	0,895	1,000
Esperada		0,600	0,700	0,800	0,900	1,000

Tabela 11: Valores da inteligência do agente esperada e medida pelo inspetor

O gráfico da figura 23 mostra os dados da tabela 11. O segmento de reta de $(0, 0)$ até $(1, 1)$ define a inteligência configurada para agente, e os losângos a inteligência medida pelo inspetor.

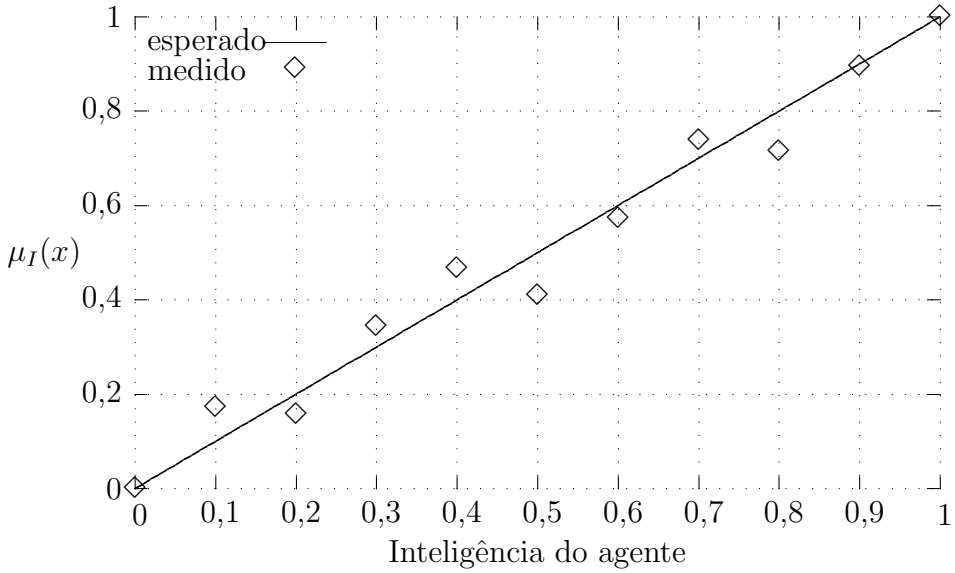

Figura 23: Inteligência medida pelo inspetor no teste de inteligência

Teste de Autonomia

Neste experimento a inteligência do agente foi fixada em 1 e sua autonomia foi variada de zero a 1, também em intervalos de $0,1$. O objetivo é verificar a capacidade do sistema de perceber diferentes graus de autonomia do agente.

Seja a a autonomia desejada para o agente, seu filtro pode ser configurado para excluir $(a \times 100)\%$ das mensagens *spam* através da seguinte expressão (as outras mensagens são deixadas na caixa de entrada):

```
header("X-Is-Spam") == "true" ?
(random() < a ? "delete" : "trash") :
("inbox")
```

Para cada valor de a, 40 mensagens de clientes e 20 de *spammers* foram enviadas. A tabela 12 mostra os valores da autonomia do agente esperada e medida pelo inspetor.

Medida	0,000	0,050	0,100	0,300	0,400	0,500
Esperada	0,000	0,100	0,200	0,300	0,400	0,500
Medida		0,700	0,600	0,800	0,850	1,000
Esperada		0,600	0,700	0,800	0,900	1,000

Tabela 12: Valores da autonomia do agente esperada e medida pelo inspetor

O gráfico da figura 24 mostra os dados da tabela 12. O segmento de reta de $(0, 0)$ até $(1, 1)$ define a autonomia configurada para agente, e os losângos a autonomia medida pelo inspetor.

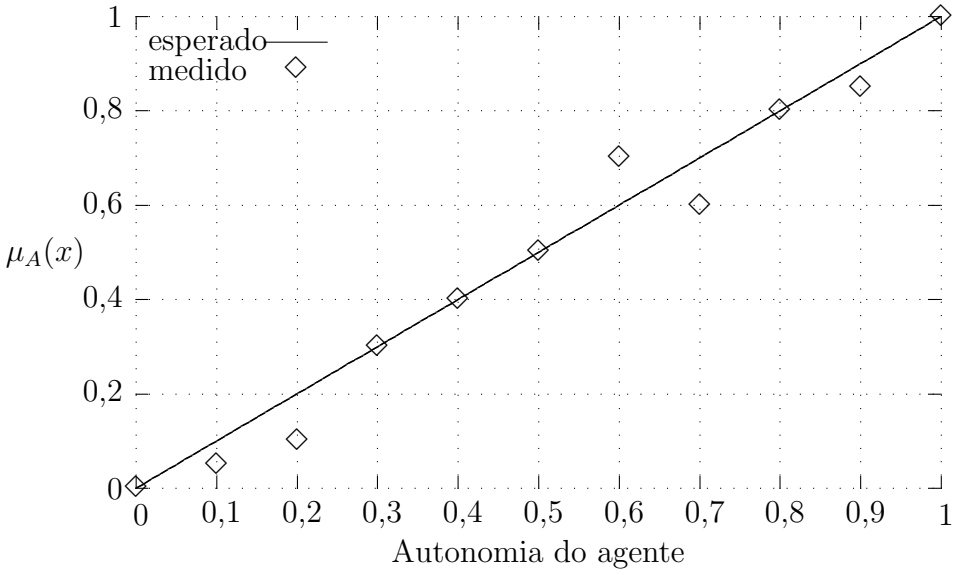

Figura 24: Autonomia medida pelo inspetor no teste de autonomia

Conclusões

Pode-se concluir que o sistema avalia corretamente a inteligência do agente quando a autonomia é mantida fixa. As variações entre a inteligência esperada e a medida podem ser explicadas pela aleatoriedade do filtro, que utiliza a função `random` para limitar a inteligência do agente e pelo pequeno volume de mensagens usadas no teste.

Da mesma forma, pode-se concluir que o sistema avalia corretamente a autonomia do agente quando a inteligência é mantida fixa. Assim como no teste de inteligência, as variações entre a autonomia esperada e a medida podem ser explicadas pela aleatoriedade do filtro, que utiliza a função `random` para limitar a autonomia do agente e pela quantidade de mensagens presentes na massa de teste.

Capítulo 5

Trabalhos Correlatos

Em uma tentativa de minimizar perdas advindas de agentes não confiáveis, vários sistemas de reputação e de controle de autonomia têm sido propostos. Sistemas de reputação têm por objetivo estabelecer a reputação de agentes em sistemas multi-agentes. O objetivo maior por trás destes sistemas é fornecer a um agente em busca por um serviço específico, fornecido por outros agentes, informações de suas reputações para que possa escolher aquele mais confiável. De fato, segundo Yu e Singh [Yu e Singh, 2003], o gerenciamento de reputação é estreitamente relacionado à confiança.

Mecanismos para a obtenção da reputação de agentes através de testemunhos de outros agentes [Yu e Singh, 2003] (figura 25) ou de suas ligações sociais em sistemas multiagentes [Pujol *et al.*, 2002; Dash *et al.*, 2004; Mao e Gratch, 2004] (figura 26) requerem que outros agentes já tenham utilizado os serviços do agente sendo avaliado e estabelecido sua reputação, que pode então ser consultada pelo agente em sua procura para a execução de um serviço.

Isto limita seu uso no cálculo da confiança em agentes computacionais que atuam como assistentes pessoais, pois estes, mesmo quando utilizando serviços de outros agentes, oferecem serviços aos seus usuários, que é o único habilitado a estabelecer a reputação do agente no cumprimento de suas tarefas em seu nome.

Ainda assim, estes sistemas têm sido utilizados em comunidades de agentes humanos para estabelecer a reputação de cada membro da comunidade. O sítio de comércio eletrônico eBay [eBay Inc., 1995] procura indicar vendedores confiáveis através do testemunho de compradores que já tenham realizado negócios com eles

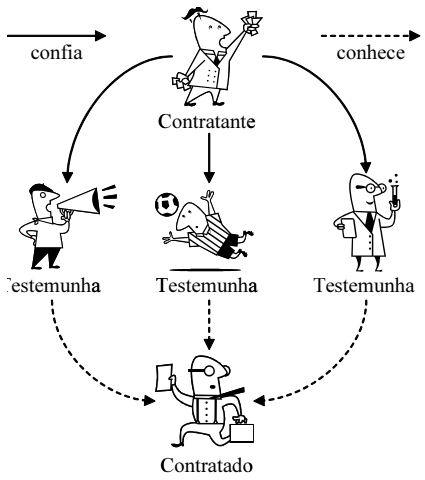

Figura 25: Estabelecimento da reputação através de testemunhas

e vice-versa. Isto leva a uma boa atuação por parte dos envolvidos, pois a possibilidade de perda de negócios devido a maus testemunhos é concreta. Porém a utilidade destes testemunhos deve ser vista com cautela, pois além de sua subjetividade os testemunhos são situados em um momento específico do tempo, e um agente pode ter mudado sua forma de atuação.

Sistemas de controle de autonomia através de interação supervisionada [Kollingbaum e Norman, 2002] e de uso de normas [d'Inverno *et al.*, 2001; Cholvy e Garion, 2004; y López e Luck, 2004] também já foram propostos. No primeiro, um agente é o mediador entre um outro agente que busca um serviço e um terceiro que o forneça, e é responsável por estabelecer sanções ao agente contratado para assegurar o cumprimento do contrato (figura 27). No segundo, normas de comportamento são estabelecidas e recompensas e sanções são aplicadas aos que as obedecem e aos que as desobedecem, respectivamente (figura 28). Porém, estes sistemas esbarram na subjetividade de regras quando da especificação de contratos, normas e sanções, pois um agente que respeite determinadas normas pode se

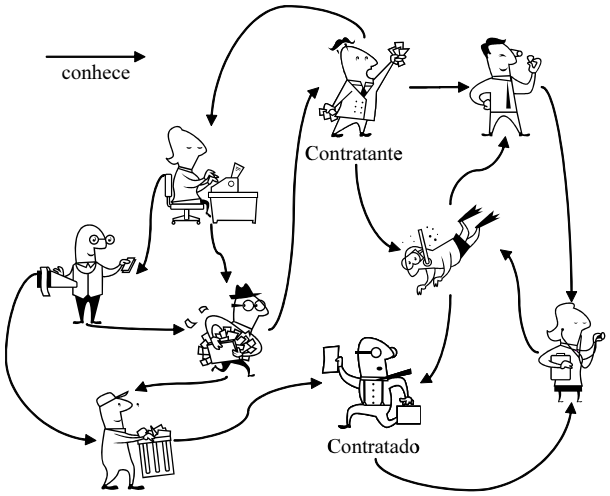

Figura 26: Estabelecimento da reputação através de ligações sociais

mostrar adequado para um usuário porém inadequado para outro. Além disso, as regras devem ser constantemente revisitadas para que sejam adequadas à novas situações.

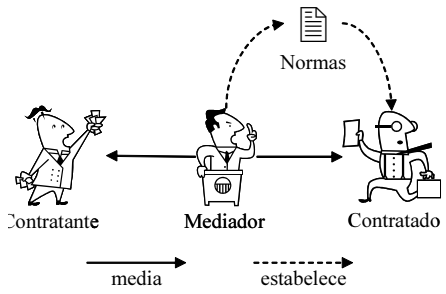

Figura 27: Estabelecimento de um contrato como garantia de fornecimento

O uso dos dois sistemas simultaneamente pode reforçar ainda mais as boas atuações de agentes fornecedores, dando informações de confiabilidade no momento da contratação de serviços e estabelecendo prêmios e sanções a serem aplicados de acordo com a atuação do fornecedor durante este fornecimento. Porém, estes sistemas, usados de forma isolada ou em conjunto, apresentam limitações no

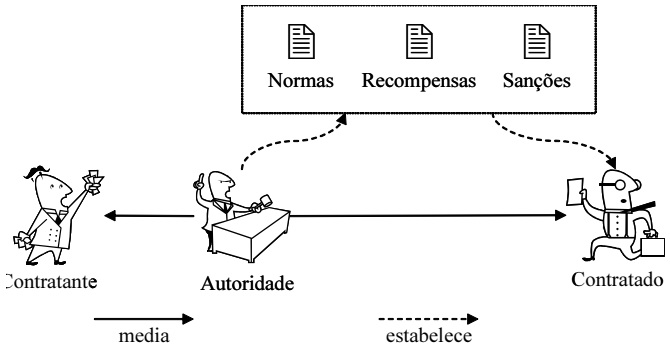

Figura 28: Estabelecimento de comportamento através de normas

estabelecimento da confiabilidade de agentes que atuam como assistentes pessoais.

Um comparativo entre os sistemas de reputação através de testemunhos, ligações sociais, interação supervisionada e uso de normas com o Modelo Clouseau encontra-se nas tabelas 13 e 14.

Parâmetro	**Modelo Clouseau**	**Testemunhos**	**Ligações sociais**
Quem avalia a reputação?	O agente inspetor	O contratante	O contratante
Como avalia a reputação?	Através da atuação do agente	Através da reputação estabelecida por testemunhas em que confia	Através das ligações sociais do contratado
Como coleta dados para a avaliação?	Investigando as ações do agente e do usuário no ambiente	Contatando as testemunhas	Contatando outros agentes
O que mede?	A atuação do agente	A confiança que as testemunhas têm no contratado	O número e a força das ligações sociais do contratado
Como controla a atuação?	Através do bloqueio de outras ações do agente	Não controla	Não controla

Tabela 13: Comparativo do Modelo Clouseau com os sistemas de reputação através de testemunhos e ligações sociais

Parâmetro	**Modelo Clouseau**	**Interação supervisionada**	**Uso de normas**
Quem avalia a reputação?	O agente inspetor	O mediador	A autoridade
Como avalia a reputação?	Através da atuação do agente	Não avalia	Não avalia
Como coleta dados para a avaliação?	Investigando as ações do agente e do usuário no ambiente	-	-
O que mede?	A atuação do agente	-	-
Como controla a atuação?	Através do bloqueio de outras ações do agente	Através do estabelecimento de normas	Através do estabelecimento de normas, sanções e recompensas

Tabela 14: Comparativo do Modelo Clouseau com os sistemas de reputação através de interação supervisionada e uso de normas

Capítulo 6

Conclusão

Foi definido o Modelo Clouseau de classificação de agentes, baseado nos modelos de classificação de soldados de Montgomery e de pessoas de Cipolla. Neste modelo, quatro tipos de agentes foram definidos baseados em sua inteligência e autonomia: proficiente, herói, perigoso e Clouseau.

Foi conduzido um experimento com o qual foi possível demonstrar que não apenas o modelo classifica corretamente os agentes testados, mas o faz de forma rápida, ajudando a identificar com agilidade os agentes do pior tipo, os do tipo Clouseau.

Outros trabalhos nas áreas de modelos de reputação e controle de autonomia fornecem outros métodos para evitar danos causados por agentes estúpidos, porém esbarram na subjetividade dos objetivos dos agentes ou na definição das regras que estes devem seguir, ou ainda requerem que determinadas características estejam presentes nos agentes ou que estes se encontrem em sistemas multi-agentes, o que nem sempre é verdade.

Controladores de atuação de agentes, conforme aqui descritos, baseiam-se apenas na interação dos agentes com o ambiente e seus usuários, e o controle de sua autonomia pode ser realizado através do bloqueio de ações potencialmente desastrosas. O controlador não precisa ter acesso à lógica de funcionamento do agente e não depende do testemunho ou ligações sociais do agente controlado com outros, e ainda assim pode avaliá-lo e controlar suas ações. Isto permite o uso de controladores com agentes concebidos para diversos domínios de aplicação.

Entretanto, outros modelos de reputação favorecem ações benéficas dos agentes

envolvidos ao invés de apenas bloquear as ações de agentes com má atuação. O uso de outros modelos de reputação em conjunto com o Modelo Clouseau pode favorecer ações benéficas ao mesmo tempo que bloqueia más atuações dos agentes.

É importante ressaltar também que o processo de classificação de um agente pode ser contínuo, pois a adaptabilidade faz parte da natureza de alguns agentes e estes podem melhorar seu desempenho com o tempo. Porém, se um agente adaptativo é classificado como Clouseau de forma constante com o passar do tempo, sem aprender para melhor servir seu usuário, ele deve ser descartado para que não cause prejuízos.

6.1 Contribuições do Trabalho

A confiabilidade de agentes é um tema muito importante na área de agentes computacionais. Estabelecer esta confiabilidade é fundamental na hora da contratação de serviços ou na escolha de um agente como assistente pessoal. Porém os modelos propostos não conseguem avaliar a confiabilidade de agentes, apenas sua reputação e ainda assim apenas do ponto de vista de outros agentes, o que torna o sistema subjetivo e sujeito a falsos testemunhos.

A contribuição deste trabalho é a formalização de um agente que estabelece automaticamente a confiabilidade de agentes medindo se suas ações são inteligentes ou estúpidas, a autonomia destas ações e as reações do usuário em relação a elas. Desta forma, o agente é classificado e controlado quanto a sua atuação do ponto de vista do usuário, e não de terceiros. A classificação e o controle automáticos de agentes pode aumentar a eficácia do agente, forçando-o a diminuir sua autonomia.

A implementação do sistema de testes mostra a viabilidade de implementação de agentes inspetores que seguem o Modelo Clouseau, e os testes empíricos reali-

zados com este sistema usando agentes reais baixados da Internet mostra a viabilidade de uso do Modelo Clouseau como forma de classificar e controlar agentes.

6.2 Limitações

O Modelo Clouseau só é capaz de classificar agentes quando começam a haver ações no ambiente. Ainda assim, algumas ações não são conclusivas quanto a sua autonomia e inteligência. Como exemplo, qual seria a inteligência da ação de um agente anti-*spam* que coloca uma mensagem na pasta lixo do usuário? Até que o usuário tome alguma ação em relação a esta mensagem, como movê-la para outra pasta ou excluí-la, a inteligência não poderá ser determinada.

Uma outra limitação é determinar os valores de inteligência e autonomia para todas as possíveis combinações de ações do agente e do usuário em objetos do ambiente. Objetos com ciclo de vida curto podem não chegar a causar problemas, mas quando seu ciclo de vida aumenta e ações são executadas constantemente, não será mais possível pedir ao usuário que informe a inteligência e a autonomia de cada nova seqüência de ações percebidas por um agente controlador. Uma forma de determinar quando novas ações não mais interferem na inteligência e autonomia do agente, ou que possibilite ao agente descobrir por si só os valores de inteligência e autonomia de novas seqüências, é necessária[6].

Uma vez que implementações do Modelo Clouseau são também agentes, uma forma de se aferir sua confiabilidade deve existir para que usuários possam confiar em seus resultados. Agentes controladores podem ser certificados por organizações isentas, ou um conjunto padrão de testes com resultados conhecidos pode ser de-

[6]O módulo de aprendizagem do inspetor não foi implementado, todas as seqüências de ações constantes na base de padrões foram inseridas manualmente e não sofriam alterações durante o tempo em que o inspetor encontrava-se ativo.

senvolvido para que não seja o agente controlador um agente Clouseau, causando prejuízos aos seus usuários.

Estas limitações deixam claro a necessidade de refinamento do Modelo Couseau.

6.3 Trabalhos Futuros

O uso do Modelo Clouseau em outros domínios de aplicação abre perspectivas interessantes a serem exploradas.

Testar o inspetor como um conselheiro para o usuário, que informa ao usuário o tipo do seu agente para ajudá-lo a ajustar ou desativá-lo sofrendo o mínimo possível de prejuízo, se não nenhum, também abre novos horizontes.

A forma de aprendizagem dos agentes controladores também pode ser revista, com a utilização de técnicas automáticas de aprendizagem que possibilitem retirar do usuário a responsabilidade de estabelecer valores de inteligência e autonomia para cada novo padrão de ações que é inserido no banco de ações do agente controlador.

Para tirar o máximo de proveito de agentes que apresentem uma atuação variável, com boas atuações em determinadas circunstâncias e más ações em outras, a avaliação e controle destes agentes pode ser baseada em objetos do ambiente ou em ações, permitindo que o agente inspetor controle sua autonomia apenas para as ações que sejam realmente desastrosas, ao invés de controlar a autonomia pelo conjunto de atuações. Desta forma, um agente com má atuação em um conjunto de objetos ou ações porém com bom atuação no restante das ocasiões teria sua autonomia controlada apenas onde apresenta um mau desempenho.

A comparação entre o Modelo Clouseau e modelos de reputação e controle de autonomia também traz novas perspectivas. O uso do Modelo Clouseau aliado

a outros modelos de reputação, pode levar a um ambiente mais cooperativo em sistemas multi-agentes.

Apêndice A

O Modelo Clouseau Aplicado em um Ambiente de Edição de Texto

Para exemplificar o uso do Modelo Clouseau em um outro domínio de aplicação, descreveremos aqui sua integração em um ambiente de edição de texto, onde o agente sendo controlado ajuda o usuário do aplicativo percebendo erros ortográficos e alterando o texto digitado automaticamente para corrigí-los sem a intervenção do usuário. O agente também altera o texto para transformar determinadas seqüências de caracteres em outras com o objetivo de facilitar a criação do texto, como por exemplo ao trocas a seqüência "(C)" por "©", chamadas seqüências de substituição. O diagrama que ilustra as relações entre o usuário, o agente corretor, o agente inspetor e o aplication de edição de texto encontra-se na figura 29.

A ordem dos eventos é como se segue:

1. O usuário digita uma seqüência de caracteres (texto), que é prontamente refletida pelo aplicativo através de seu aparecimento no documento;

2. O agente inspetor armazena o texto como um objeto do ambiente;

3. O agente corretor intercepta o texto e consulta seu dicionário para identificar possíveis erros ortográficos ou outras seqüências de substituição;

4. Caso o agente corretor identifique que o texto deva ser substituído por outro, ele desfaz a digitação do usuário e inclui seu próprio texto;

5. O agente inspetor percebe a ação do agente corretor e caso sua classificação

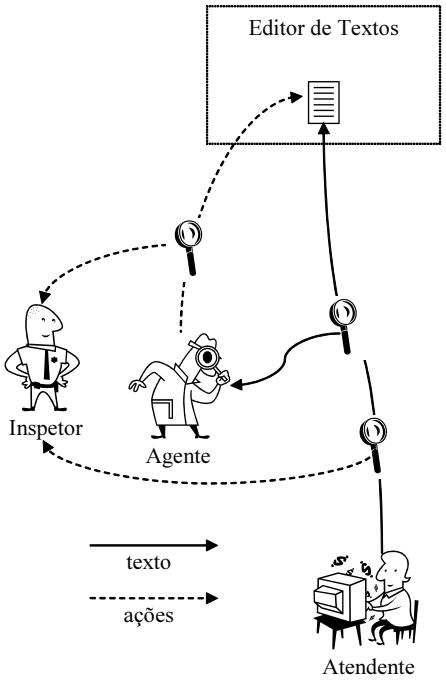

Figura 29: Diagrama do ambiente onde atua o agente corretor

para o agente corretor seja "Clouseau", intercepta a ação e evita a modificação do texto; e

6. O agente inspetor armazena a ação em seu histórico de ações e consulta sua base de padrões para reavaliar a ação do agente corretor.

Os módulos do agente inspetor, descritos no capítulo 3, encontram-se detalhados a seguir.

A.1 Sensor

O sensor do agente inspetor no âmbito de aplicativos de edição de textos atua capturando os caracteres digitados pelo usuário. Quando o agente corretor toma uma ação de substituir um trecho do texto formado por alguns de seus caracteres,

o sensor avalia quais foram os caracteres substituídos e os armazena no histórico de ações, juntamente com a ação do agente corretor.

O sensor também atua na captura de ações entre o usuário e o aplicativo. Quando o usuário toma a ação de desfazer a substituição de texto realizada pelo agente corretor, o sensor armazena o último texto substituído pelo agente juntamente com a ação de "desfazer" do usuário no histórico de ações.

A.2 Histórico de Ações

O histórico de ações armazenará trechos do texto juntamente com ações de substituição do agente corretor e ações de "desfazer" do usuário. Estes trechos são os objetos alvo das ações, e as únicas ações possíveis são as de substituição e "desfazer".

A.3 Base de Padrões

A base de padrões deve conter os padrões de ações possíveis para objetos alvo. A tabela 15 exemplifica uma possível base de padrões que identifica os graus de inteligência e autonomia para padrões comumente encontrados em aplicativos de edição de textos.

A.4 Diagnosticador

O diagnosticador atua exatamente como descrito na seção 3.4, buscando por seqüências de ações presentes no histórico de ações na base de padrões, de forma a computar a inteligência e a autonomia do agente corretor levando em conta todo o seu conjunto de atuações, para então realizar a classificação do agente corretor.

Padrão de ações	**Significado**	$\mu_I(p)$	$\mu_A(p)$
$\{s\}$	Texto substituído pelo agente corretor	?	1
$\{s, s\}$	Texto substituído pelo agente corretor em duas ocasiões	1	1
$\{s, d\}$	Texto substituído e ação desfeita pelo usuário	0	1

Tabela 15: Padrões de ações na base de padrões do agente inspetor para o domínio de aplicativos de edição de textos

A.5 Controlador

O controlador também atua como descrito anteriormente, na seção 3.5, ajustando a autonomia do agente corretor dada sua classificação realizada pelo diagnosticador.

A.6 Aprendizagem

Dado que o universo de ações possíveis neste domínio de aplicação é reduzido, limitando-se a ações de substituição de texto por parte do agente corretor e ações de "desfazer" por parte do usuário, a base de padrões pode ser construída a priori.

A.7 Conclusão

A adaptação do Modelo Clouseau é um processo simples, bastanto para tal a identificação dos objetos do domínio de aplicação que são alvo das ações do agente a ser controlado e do usuário, e a construção de uma base de padrões apropriada

para a classificação do agente.

Em casos onde a construção da base de padrões não pode ser realizada a priori, o módulo de aprendizagem pode ser escrito de forma a permitir a construção desta base, a medida em que ações do agente e do usuário ocorrem em objetos do ambiente.

Apêndice B

Descrição de Ações em FIPA ACL

As mensagens nas seções seguintes foram definidas de acordo com as seguintes especificações FIPA: **FIPA00001** [FIPA, 2002a], **FIPA00026** [FIPA, 2002d], **FIPA00037** [FIPA, 2002c] e **FIPA00061** [FIPA, 2002b].

Nas listagens, `anti-spam` é o nome do agente anti-*spam*, `pop3-server` o nome do agente POP3, `imap-server` o nome do agente IMAP e `email-client` o nome do programa de correio eletrônico do usuário.

O campo `conversation-id` deve conter um identificador único para a conversação, normalmente o nome do agente que inicia a conversa seguido de um número inteiro que é incrementado a cada nova conversa.

As seguintes palavras, quando aparecem no campo `content` em itálico, têm o significado descrito abaixo:

- *user-name*: Nome do usuário para se conectar ao agente POP3 ou IMAP;
- *password*: Senha do usuário para se conectar ao agente POP3 ou IMAP;
- *conversation-id*: Identificador da conversação entre os agentes;
- *num-messages*: Número total de mensagens existentes;
- *total-size*: Tamanho total em bytes ocupados pelas mensagens;
- *message-num*: Índice da mensagem no agente POP3;
- *size*: Tamanho rm bytes ocupado por uma mensagem;
- *headers*: Apenas os cabeçalhos da mensagem;

- *body*: Apenas o corpo da mensagem;
- *contents*: Conteúdo integral da mensagem (*headers* e *body*);
- *tag*: Etiqueta de comando no agente IMAP;
- *folder*: Pasta de mensagens do agente IMAP; e
- *uid*: Identificador único da mensagem no agente IMAP.

A seqüência de caracteres "<EOL>" designa um salto de linha formado pelos caracteres "CR" (13 em decimal) e "LF" (10 em decimal), e a seqüência de caracteres "<ANY>" designa que caracteres serão descartados nas mensagens. Linhas compridas são quebradas em duas ou mais linhas e marcadas pelo caracter "\".

B.1 Mensagens de Ações do Agente Anti-*Spam* no Ambiente

As ações do agente anti-*spam* no ambiente consistem em comandos enviados a servidores POP3, de onde o agente lê as novas mensagens de correio eletrônico do usuário e IMAP, onde ele insere as mensagens filtradas. Estas mensagens encontram-se descritas a seguir.

B.1.1 Autenticação no agente POP3

A autenticação do agente anti-*spam* no servidor POP3 de seu usuário, onde ele busca as mensagens para classificá-las, é ilustrada na listagem 2.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "USER user-name<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(inform-result
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "PASS password<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(agree
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC1939)
```

Listagem 2: Autenticação do agente anti-*spam* no agente POP3

B.1.2 Recebimento de mensagens de correio eletrônico do agente POP3

Para receber mensagens de correio eletrônico do agente POP3, o agente anti-*spam* envia uma mensagem para descobrir quantas mensagens existem no agente POP3, conforme listagem 3.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "STAT<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(inform-result
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+OK num-messages total-size<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)
```

Listagem 3: Descobrimento do número de mensagens de correio eletrônico no agente POP3

Após descobrir quantas mensagens existem no servidor, o agente anti-*spam* envia uma mensagem de recuperação de mensagem para cada uma existente no servidor, excluindo-as durante o processo conforme listagem 4.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "RETR message-num<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(inform-result
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+OK size octets<EOL>contents<EOL>.<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "DELE message-num<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)

(inform-result
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC1939)
```

Listagem 4: Recuperação e exclusão de mensagens no agente POP3

B.1.3 Fim de sessão com o agente POP3

Ao fim do processamento, o agente anti-*spam* envia uma mensagem para terminar a conexão com o agente POP3, conforme a listagem 5.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name pop3-server))
  :content "QUIT"
  :conversation-id: conversation-id
  :language RFC1939)

(inform-result
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC1939)
```

Listagem 5: Finalização da sessão do agente anti-*spam* com o agente POP3

B.1.4 Erros durante a sessão com o agente POP3

Quando o agente anti-*spam* faz um requerimento ao agente POP3 que resulta em erro, o agente POP3 envia uma mensagem com a descrição do erro para o agente anti-*spam*. A sessão permanece aberta (listagem 6).

```
(not-understood
  :sender (agent-identifier :name pop3-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "-ERR errormessage<EOL>"
  :conversation-id: conversation-id
  :language RFC1939)
```

Listagem 6: Erro durante a sessão com o agente POP3

B.1.5 Autenticação no agente IMAP

A autenticação do agente anti-*spam* no servidor IMAP de seu usuário, onde ele insere as mensagens filtradas do agente POP3, é ilustrada na listagem 7.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag LOGIN user-name password<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 7: Autenticação do agente anti-*spam* no agente IMAP

B.1.6 Inserção de mensagem de correio eletrônico no agente IMAP

A inserção de uma mensagem de correio eletrônico em uma pasta do agente IMAP é ilustrada na listagem 8.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag APPEND folder {size}<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "+ <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name imap-server))
  :content "contents"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "<ANY>tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 8: Inserção de mensagem de correio eletrônico em no agente IMAP

B.1.7 Fim de sessão com o agente IMAP

Ao final da sessão, o agente anti-*spam* deve encerrar a sessão conforme a listagem 9.

```
(request
  :sender (agent-identifier :name anti-spam)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag LOGOUT<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "<ANY>tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 9: Finalização da sessão do agente anti-*spam* com o agente IMAP

B.1.8 Erros durante a sessão com o agente IMAP

Quando o agente anti-*spam* faz um requerimento ao agente IMAP que resulta em erro, o agente IMAP envia uma mensagem com a descrição do erro para o agente anti-*spam*. A sessão permanece aberta (listagem 10).

```
(not-understood
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name anti-spam))
  :content "* BAD errormessage<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 10: Erro durante a sessão com o agente IMAP

B.2 Mensagens de Ações do Usuário no Ambiente

As ações do usuário no ambiente consistem em comandos enviados a servidores IMAP de um programa de correio eletrônico, de onde ele abre, busca, copia, exclui, move e responde mensagens de correio eletrônico. Estas mensagens encontram-se

descritas a seguir[7].

B.2.1 Seleção de pasta no agente IMAP

Para averiguar o conteúdo de uma pasta no agente IMAP, o usuário deve selecionar a pasta desejada, conforme a listagem 11.

```
(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag SELECT folder<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "<ANY>tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 11: Seleção de pasta no agente IMAP

B.2.2 Marcação de mensagem de correio eletrônico como lida no agente IMAP

O ato de ler uma mensagem na aplicação de correio eletrônico causa apenas a marcação da mensagem como lida no agente IMAP. Para marcar uma mensagem de correio eletrônico como lida, o usuário envia uma mensagem conforme a listagem 12.

[7] As trocas de mensagens para autenticação (listagem 7), encerramento da sessão (listagem 9) e reporte de erro (listagem 10) são as mesmas descritas para o agente anti-*spam*

```
(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag UID STORE uid +Flags.silent (\SEEN)<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 12: Marcação de mensagem de correio eletrônico como lida no agente IMAP

B.2.3 Busca de mensagem de correio eletrônico no agente IMAP

Para exibir efetivamente uma mensagem de correio eletrônico, a aplicação de correio eletrônico busca a mensagem no agente IMAP (listagem 13).

```
(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag UID FETCH uid (BODY.PEEK[HEADER])<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "* message-num FETCH (UID uid BODY[HEADER] {size}<EOL>" \
    "headers<EOL>" \
    ")<EOL>" \
    "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag UID FETCH uid (BODY.PEEK[1])<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "* message-num FETCH (UID uid BODY[1] {size}<EOL>" \
    "body<EOL>" \
    ")<EOL>" \
    "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 13: Busca de mensagem de correio eletrônico no agente IMAP

B.2.4 Cópia de mensagem de correio eletrônico para outra pasta no agente IMAP

Para copiar uma mensagem de correio eletrônico para outra pasta no agente IMAP, o usuário envia uma mensagem conforme a listagem 14.

```
(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag UID COPY uid folder<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 14: Cópia de mensagem de correio eletrônico para outra pasta no agente IMAP

B.2.5 Exclusão de mensagem de correio eletrônico no agente IMAP

Para excluir uma mensagem de correio eletrônico no agente IMAP, o usuário envia uma mensagem conforme a listagem 15.

```
(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag UID STORE uid +Flags.silent (\Deleted)<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag EXPUNGE<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "<ANY>tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 15: Exclusão de mensagem de correio eletrônico no agente IMAP

B.2.6 Mudança de mensagem de correio eletrônico para outra pasta no agente IMAP

Para mudar uma mensagem de correio eletrônico para outra pasta no agente IMAP, o usuário envia uma mensagem de cópia para a pasta de destino conforme a listagem 14, e então envia uma mensagem para a exclusão da mensagem de correio eletrônico de sua pasta original, conforme a listagem 15.

B.2.7 Marcação de mensagem de correio eletrônico como respondida no agente IMAP

O ato de responder uma mensagem na aplicação de correio eletrônico causa apenas a marcação da mensagem como respondida no agente IMAP. Para marcar uma mensagem de correio eletrônico como respondida, o usuário envia uma mensagem conforme a listagem 16.

```
(request
  :sender (agent-identifier :name email-client)
  :receiver (set (agent-identifier :name imap-server))
  :content "tag UID STORE uid +Flags.silent (\Answered)<EOL>"
  :conversation-id: conversation-id
  :language RFC3501)

(inform-result
  :sender (agent-identifier :name imap-server)
  :receiver (set (agent-identifier :name email-client))
  :content "tag OK <ANY><EOL>"
  :conversation-id: conversation-id
  :language RFC3501)
```

Listagem 16: Marcação de mensagem de correio eletrônico como respondida no agente IMAP

Apêndice C

Sistema de Teste

O sistema utilizado para testar o Modelo Clouseau foi desenvolvido em Java™ da Sun Microsystems, Inc., versão 1.4.2_05. O diagrama de classes UML foi separado por classe, e é apresentado a seguir. Após o diagrama, as principais classes do sistema são explicadas.

C.1 Diagramas de Classes

C.1.1 Pacote `caa.agent`

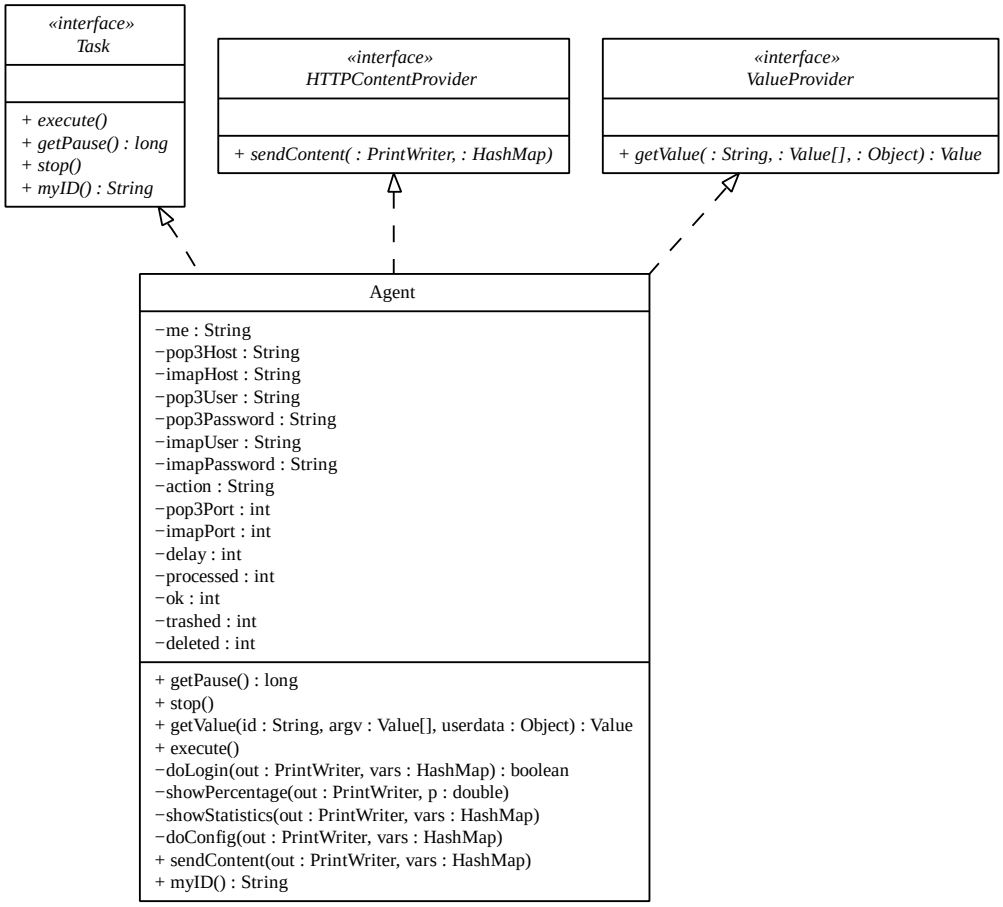

Figura 30: Classe `Agent`

C.1.2 Pacote `caa.controller`

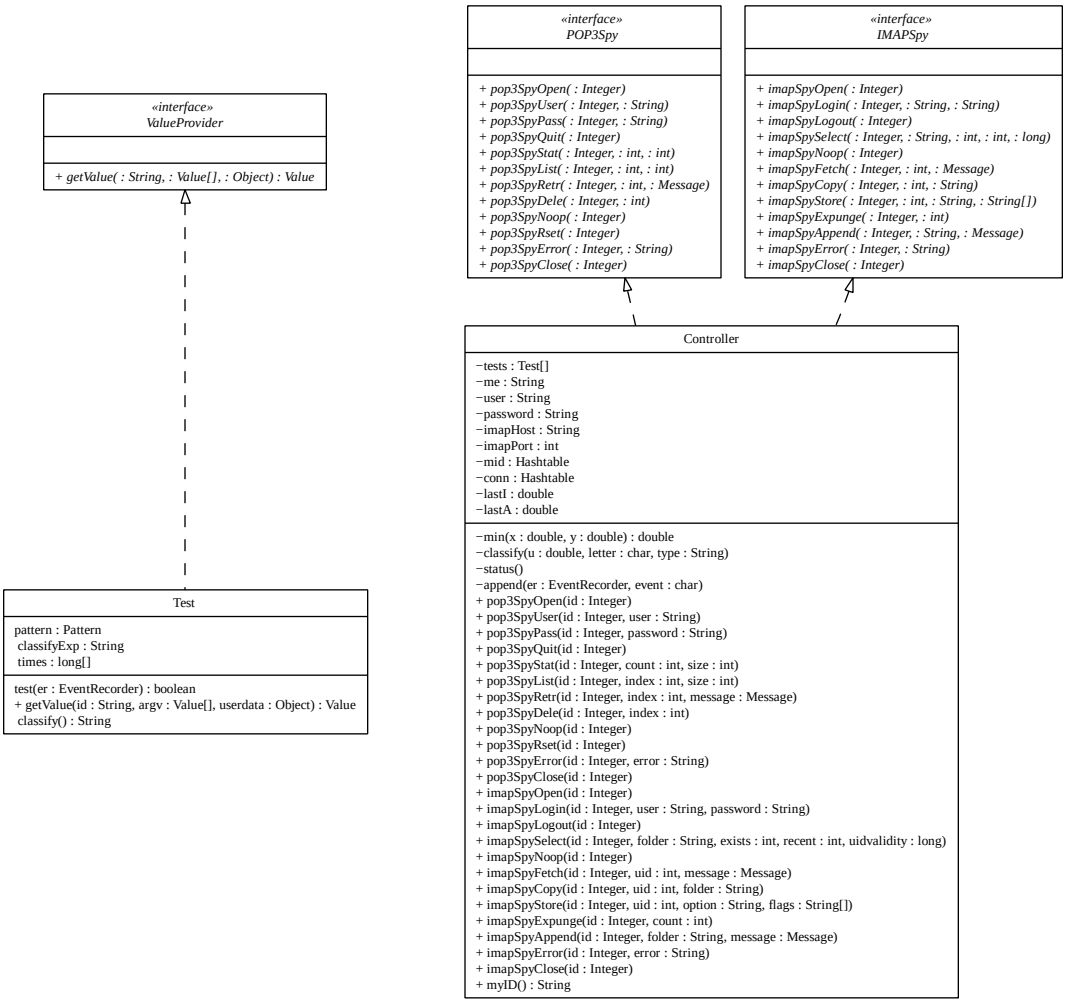

Figura 31: Classe `Controller`

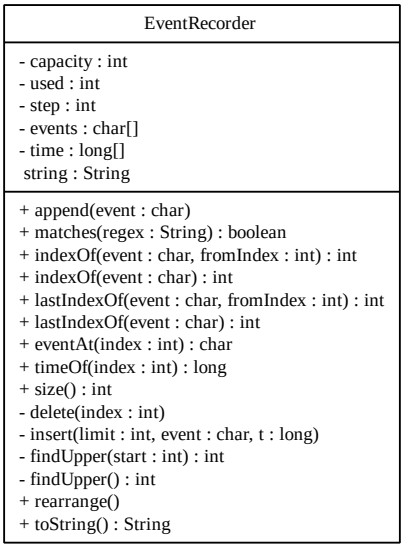

Figura 32: Classe `EventRecorder`

C.1.3 Pacote `caa.http`

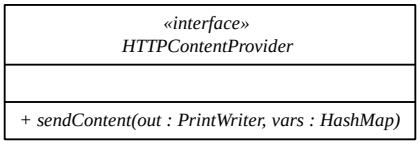

Figura 33: *Interface* `HTTPContentProvider`

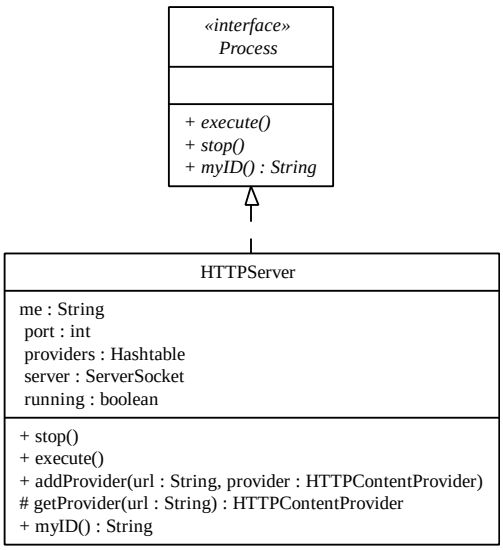

Figura 34: Classe `HTTPServer`

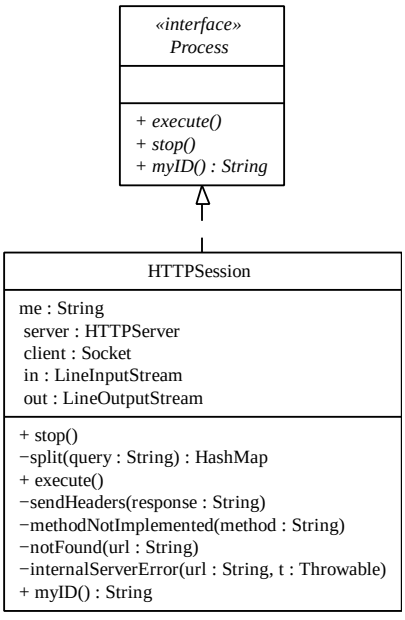

Figura 35: Classe `HTTPSession`

C.1.4 Pacote `caa.mail`

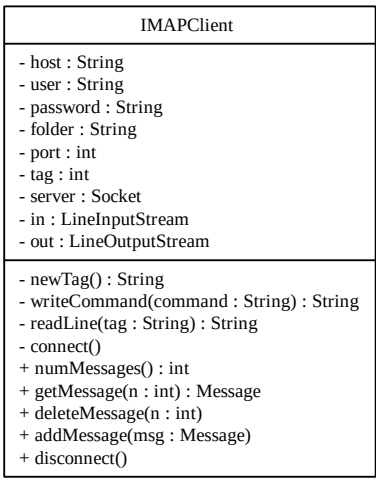

Figura 36: Classe `IMAPClient`

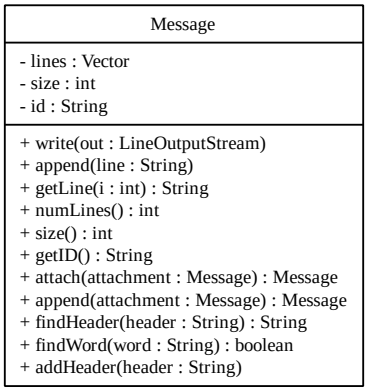

Figura 37: Classe `Message`

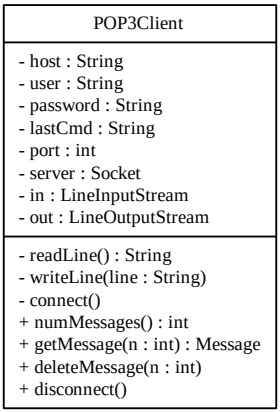

Figura 38: Classe `POP3Client`

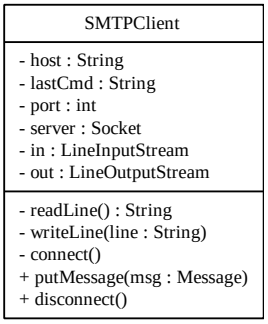

Figura 39: Classe `SMTPClient`

C.1.5 Pacote `caa.math`

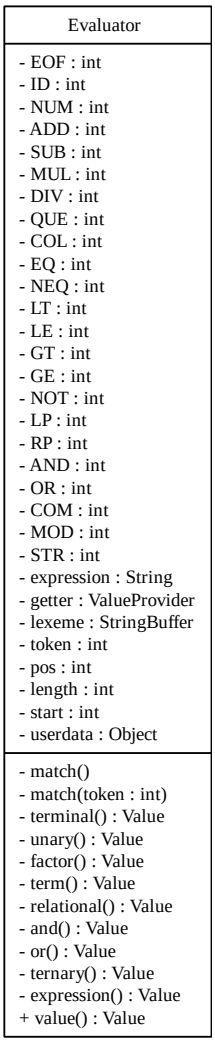

Figura 40: Classe `Evaluator`

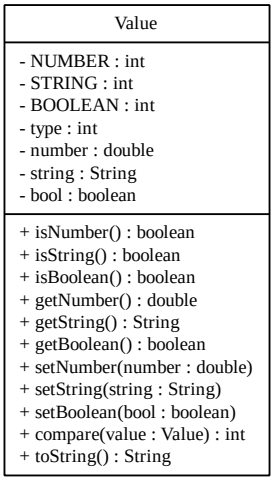

Figura 41: Classe `Value`

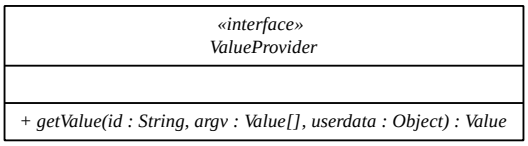

Figura 42: *Interface* `ValueProvider`

C.1.6 Pacote `caa.spy`

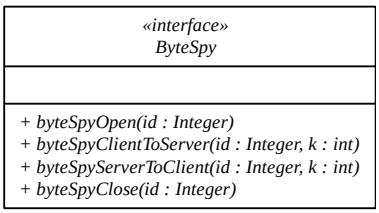

Figura 43: *Interface* `ByteSpy`

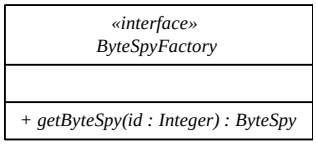

Figura 44: *Interface* `ByteSpyFactory`

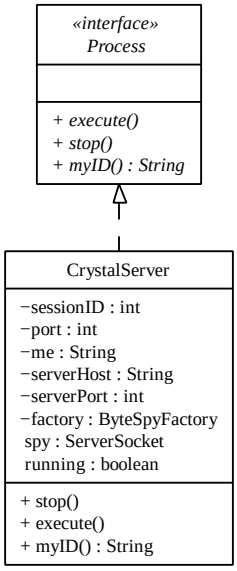

Figura 45: Classe `CrystalServer`

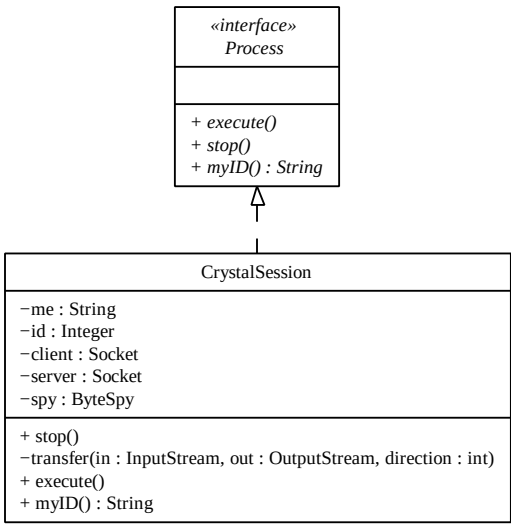

Figura 46: Classe `CrystalSession`

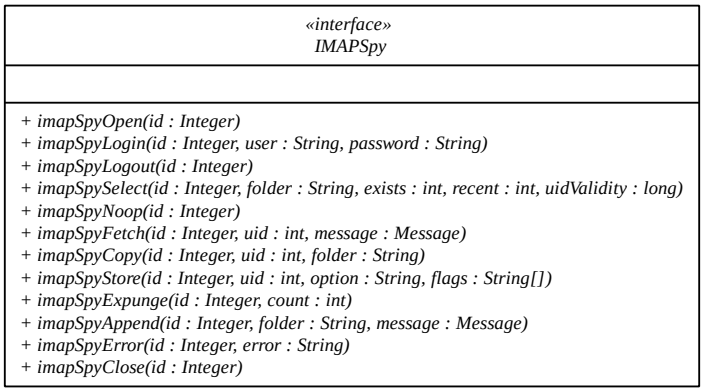

Figura 47: *Interface* `IMAPSpy`

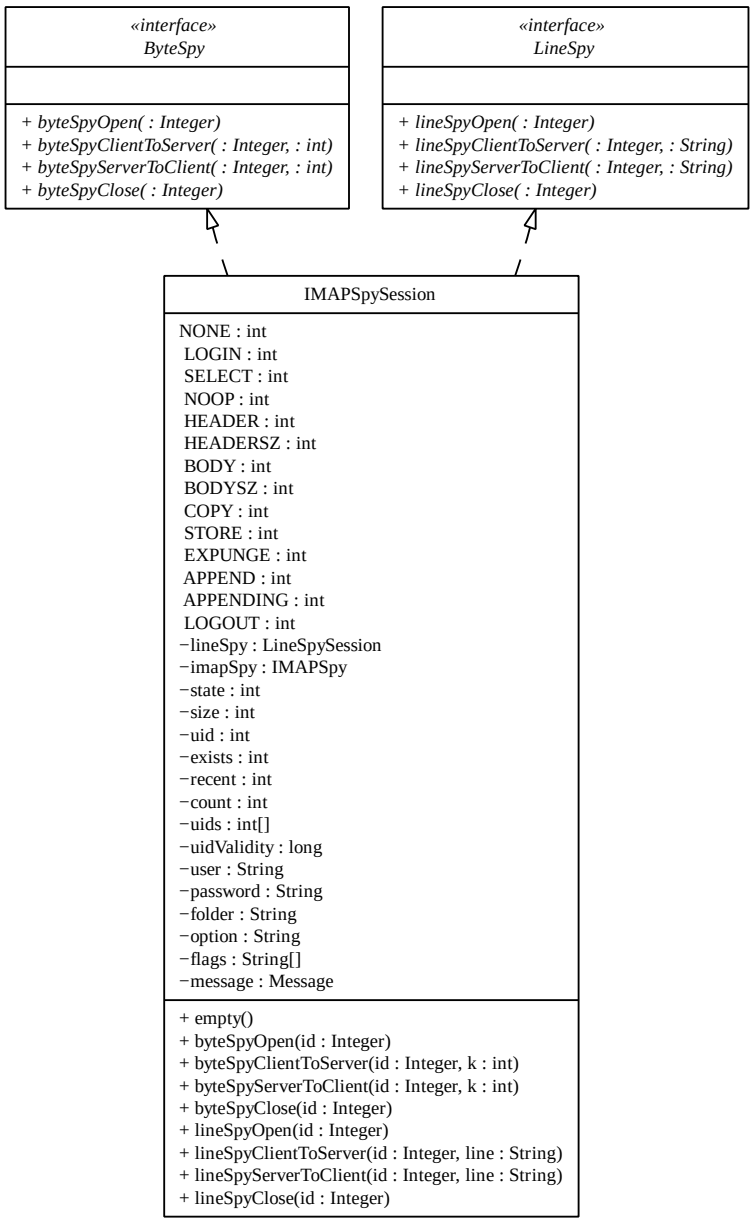

Figura 48: Classe `IMAPSpySession`

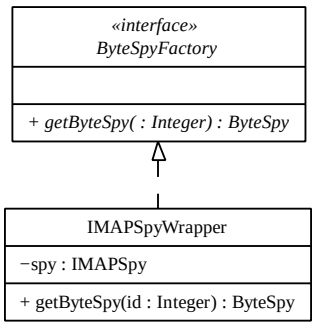

Figura 49: Classe `IMAPSpyWrapper`

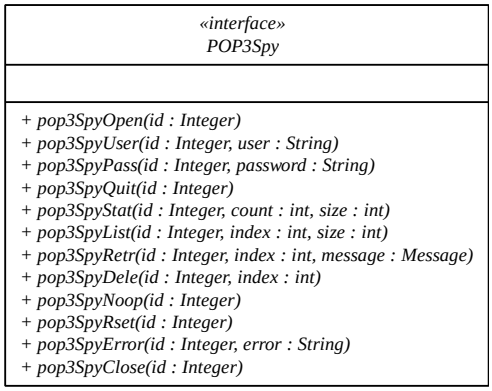

Figura 50: *Interface* `POP3Spy`

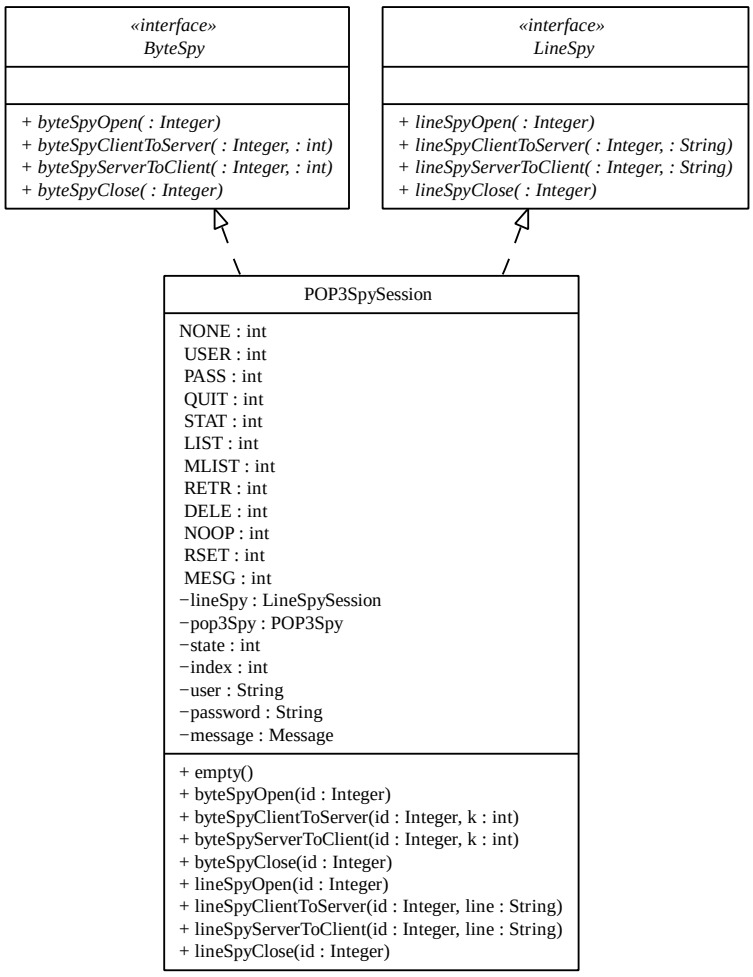

Figura 51: Classe `POP3SpySession`

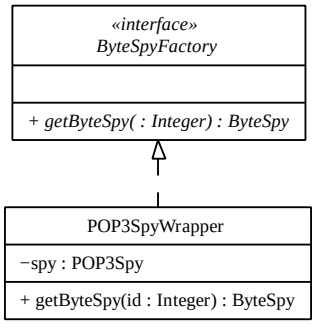

Figura 52: Classe `POP3SpyWrapper`

C.1.7 Pacote `caa.system`

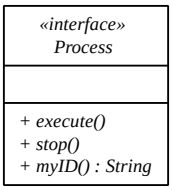

Figura 53: *Interface* `Process`

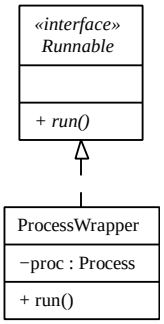

Figura 54: Classe `ProcessWrapper`

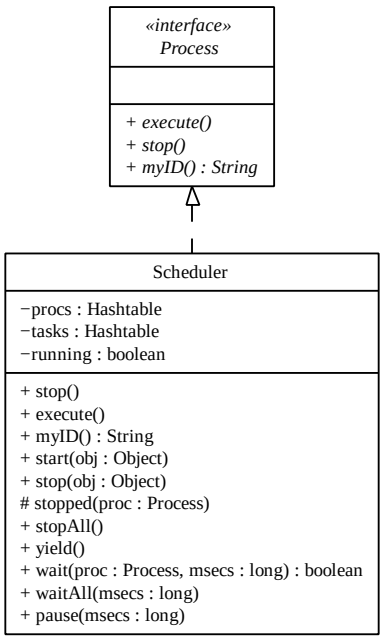

Figura 55: Classe `Scheduler`

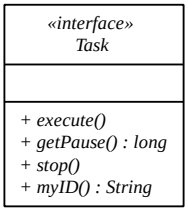

Figura 56: *Interface* `Task`

C.2 Detalhamento

C.2.1 Pacote `caa.agent`

Classe `Agent`

Instâncias da classe `Agent` são os agentes do sistema. Seu construtor requer os endereços e portas dos servidores POP3 e IMAP de onde ele busca mensagens

e onde ele insere as mensagens filtradas, respectivamente, o *login* do usuário e sua senha nestes servidores, o intervalo de tempo em que o agente deve procurar por novas mensagens no servidor POP3, o filtro *action* que será utilizado para classificar as mensagens e uma instância de um servidor `caa.http.HTTPServer`, através do qual seu usuário poderá acompanhar o trabalho do agente e configurar seu filtro em tempo de execução.

A classe `Agent` implementa as *interfaces* `caa.system.Task`, para que possa ser adicionado ao agendador de tarefas do sistema, e `caa.http.HTTPContentProvider`, para que possa ser registrado como um provedor de conteúdo em um servidor `caa.http.HTTPServer`.

O filtro informado em seu construtor é avaliado para cada mensagem recebida usando uma instância de `caa.math.expression.Evaluator`.

C.2.2 Pacote `caa.controller`

Classe `Controller`

Instâncias de `Controller` são responsáveis por capturar ações do agente e do usuário no ambiente e classificar o agente de acordo com estas ações. Para ser instanciado, requer os dados do servidor IMAP do usuário, incluindo *login* e senha, para que possa salvar cópias das mensagens excluídas pelo agente.

`Controller` implementa as *interfaces* `caa.spy.POP3Spy` e `caa.spy.IMAPSpy`, e vários de seus métodos, requeridos por estas *interfaces*, são chamados de forma assíncrona por outras classes conforme ocorram ações no ambiente.

C.2.3 Pacote `caa.http`

Classe `HTTPServer`

A classe `HTTPServer` pode ser instanciada para se obter um servidor HTTP que roda como um processo assíncrono no sistema. Em seu construtor é necessário informar apenas a porta à qual ele deve se ligar (geralmente 80). O conteúdo que é servido por ele deve ser registrado por chamadas ao método `addProvider`, que recebe como parâmetros uma *string* que indica o caminho do objeto que será servido e uma instância de `HTTPContentProvider`. Vários provedores de conteúdo podem ser adicionados ao mesmo servidor.

Sempre que o servidor receber uma requisição para um `HTTPContentProvider` registrado, ele chama o método `sendContent` do provedor, enviando como parâmetros uma *stream* `java.io.PrintWriter`, por onde o provedor pode enviar conteúdo para o *browser*, e uma `java.util.HashMap`, inicializada com todos os pares de `nome=valor` encontrados na *querystring* passada ao servidor juntamente com a URL.

C.2.4 Pacote `caa.mail`

Classes `IMAPClient`, `POP3Client` e `SMTPClient`

Estas classes podem ser instanciadas para se realizar conexões com servidores IMAP, POP3 e SMTP, e seus métodos permitem que ações sejam tomadas sem a preocupação com o protocolo propriamente dito. Como exemplo, para se obter o número de mensagens disponíveis em um servidor POP3, basta, após a conexão, chamar o método `numMessages`.

Estas classes abstraem menssagens de correio eletrônico através da classe

`Message`.

Classe `Message`

Esta classe abstrae uma mensagem de acordo com [Resnick, 2001]. É possível consultar os cabeçalhos da mensagem, o corpo da mensagem, adicionar novos cabeçalhos etc. Instâncias desta classe são retornadas por alguns métodos de `IMAPClient` e `POP3Client`, e podem ser enviadas usando `SMTPClient`.

C.2.5 Pacote `caa.math`

Classe `Evaluator`

A classe `Evaluator` é um avaliador de expressões. Seu construtor recebe como argumentos uma *string* contendo a expressão a ser avaliada, uma instância de `ValueProvider`, e uma instância de `java.lang.Object`.

A gramática da linguagem implementada pelo *parser* de `Evaluator` é definida na listagem 17.

Quando `Evaluator` encontra um identificador, ele monta um vetor de instâncias de `Value` com os argumentos da chamada de função, se o identificador for sucedido por um caracter "`(`", e passa este identificador, o vetor de argumentos, ou nulo se o "`(` não suceder o identificador, e o objeto `java.lang.Object` informado na instanciação de `Evaluator` para o método `getValue` da instância de `ValueProvider` informada também na instanciação de `Evaluator`. Este método deve retornar uma instância de `Value` com o valor do identificador ou da função, ou lançar uma exceção `EvaluatorException` informando que o identificador é desconhecido ou no caso de qualquer outro erro.

```
expression  →  ternary

ternary     →  or [ "?" expression ":" expression ]

or          →  and { "||" and }

and         →  relational { "&&" relational }

relational  →  term { ( "<" | "<=" | ">" | ">=" | "==" | "!=" ) term }

term        →  factor { ( "+" | "-" ) factor }

factor      →  unary { ( "*" | "/" | "%" ) unary }

unary       →  [ "-" | "+" | "!" ] terminal

terminal    →  identifier [ "(" [ expression { "," expression } ] ")" ]
            |  number
            |  string
            |  "(" expression ")"
```

Listagem 17: Gramática BNF do *parser* de `Evaluator`

Ao final da avaliação da expressão, `Evaluator` retorna uma instância de `Value` com seu valor de retorno.

Classe `Value`

Instâncias de `Value` representam dados dos tipos lógico booleano, número real e *string*. Elas são passadas como argumentos de funções para instâncias de `ValueProvider` e são retornadas por `Evaluator`.

C.2.6 Pacote `caa.spy`

Classe `CrystalServer`

Instâncias de `CrystalServer` são processos assíncronos que atuam como um *proxy* entre clientes e servidores. Seu construtor recebe como argumentos a porta onde se

ligará, o servidor remoto e sua porta, e uma instância de `ByteSpyFactory`. Todas as conexões feitas à ele serão repassadas ao servidor especificado de forma transparente, porém qualquer *byte* que vá do cliente ao servidor ou vice-versa será capturado e passado aos métodos `byteSpyClientToServer` e `byteSpyServerToClient` da instância de `ByteSpy` retornada por `ByteSpyFactory` quando a conexão é iniciada (figura 57).

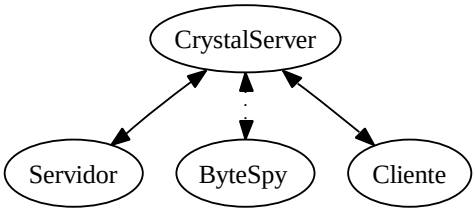

Figura 57: Uma instância de `CrystalServer` que repassa dados de uma conexão para uma instância de `ByteSpy`

Classe `LineSpyWrapper` e `LineSpySession`

Estas classes encapsulam a funcionalidade de um `ByteSpy` para permitir que linhas terminadas pelo caracter `LF` ou pela seqüência `CR LF` que trafeguem entre o cliente e o servidor sejam capturadas. Desta forma, outras classes podem ser facilmente escritas para capturar informações de nível mais alto que trafeguem pela conexão.

Quando uma linha é terminada, estas classes chamam os métodos `lineSpyClientToServer` ou `lineSpyServerToClient` que podem então interpretar o que acontece na conexão entre o cliente e o servidor.

Classe `POP3SpyWrapper` e `POP3SpySession`

Estas classes encapsulam a funcionalidade de um `LineSpy`, mas contém vários métodos que são chamados em instâncias de `POP3Spy` de acordo com os comandos POP3 [Myers e Rose, 1996] que o cliente envia para o servidor, como o método `pop3SpyRetr`, que é chamado quando o cliente recebe uma mensagem do servidor em resposta a um comando `RETR`. A mensagem retornada pelo servidor é passada à este método como uma instância de `caa.mail.Message`, o que torna fácil sua manipulação.

Classe `IMAPSpyWrapper` e `IMAPSpySession`

Da mesma forma que as classes `POP3SpyWrapper` e `POP3SpySession`, estas classes chamam métodos de instâncias de `IMAPSpy` de acordo com os comandos IMAP [Crispin, 2003] que trafegam pela conexão do cliente com o servidor.

C.2.7 Pacote `caa.system`

Interface `Process`

Esta interface deve ser implementada por todas as classes que desejam ser executadas como processos assíncronos no sistema. Quando iniciadas, seu método `execute` é chamado, e será executado até que retorne ao até que seu método `stop` seja invocado. Como a chamada a `stop` fará para terminar o método `execute` é responsabilidade da instância de `Process`.

Interface `Task`

Esta interface deve ser implementada pelas classes que desejam ser agendadas para execução periódica pelo `Scheduler`. Todas as instâncias de `Task` são enfileiradas pelo `Scheduler` e chamadas uma a uma, respeitando seus tempos de pausa mínimos. Quando não há necessidade de execução assíncrona (como por exemplo no agente anti-*spam*, que verifica a caixa do usuário de tempos em tempos), é preferível usar a `Task`, pois todas as suas tarefas são executas no mesmo processo de `Scheduler`, economizando recursos do sistema.

Classe `Scheduler`

Esta classe contém apenas métodos estáticos usados para agendar tarefas, iniciar processos e parar tarefas e processos. Quando uma tarefa ou processo lança uma exceção, `Scheduler` a captura e termina o processo ou tarefa, mantendo o restante do sistema em funcionamento.

REFERÊNCIAS

[ArGo, 2004] ArGo. Argo Mail Server freeware version 1.8. `http://www.argosoft.com/`, 2004. ArGo Software Design.

[Asgharzadeh, 1996] Ali Asgharzadeh. Image analysis and enhancement using fuzzy rule based expert system. In *Proceedings of the 1996 ACM Symposium on Applied Computing*, 1996.

[Bates, 1994] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, 1994.

[Benisch *et al.*, 2004] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, e M. Tschantz. Botticelli: A supply chain management agent. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1174–1181, 2004.

[Bloch, 1994] Isabelle Bloch. Fuzzy sets in image processing. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, 1994.

[Brustoloni, 1991] Jose C. Brustoloni. Autonomous agents: Characterization and requirements. Carnegie Mellon Technical Report CMU-CS-91-204, Carnegie Mellon University, Pittsburgh, 1991.

[Caglayan e Harrison, 1997] Alper Caglayan e Colin Harrison. *Agent Source Book - A Complete Guide to Desktop, Internet and Intranet Agents*. Wiley Computer Publishing, 1997.

[Castelfranchi, 1995] Cristiano Castelfranchi. *Guarantees for Autonomy in Cognitive Agent Architecture*, pages 56–70. Springer-Verlag, Berlin, 1995.

[Chau e Yeh, 2000] Rowena Chau e Chung-Hsing Yeh. Explorative multilingual text retrieval based on fuzzy multilingual keyword classification. In *Proceedings of the Fifth International Workshop on Information Retrieval with Asian Languages*, 2000.

[Chen e Sycara, 1998] Liren Chen e Katia Sycara. Webmate: A personal agent for browsing and searching. In Katia P. Sycara e Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 132–139, New York, May 1998. ACM Press.

[Chien *et al.*, 2004] Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castano, Ashley Davies, e Rachel Lee. The EO-1 autonomous science agent. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 420–427, 2004.

[Cholvy e Garion, 2004] Laurence Cholvy e Christophe Garion. Desires, norms and constraints. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 724–731, 2004.

[Cipolla, 1988] Carlo Maria Cipolla. *Allegro ma non troppo*. Il Mulino, 1988.

[Coen, 1994] Michael H. Coen. Sodabot: A software agent environment and construction system. Technical Report 1493, MIT AI Lab, June 1994.

[Crispin, 2003] M. Crispin. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501 (Proposed Standard), March 2003.

[Crystaliz, 1997] Crystaliz. The MuBot agent. `http://www.crystaliz.com/logicware/mubot.html`, 1997. Crystaliz, Inc.

[Dash *et al.*, 2004] Rajdeep K. Dash, Sarvapali D. Ramchurn, e Nicholas R. Jennings. Trust-based mechanism design. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 748–755, 2004.

[d'Inverno *et al.*, 2001] Mark d'Inverno, Fabiola López y López, e Michael Luck. Constraining autonomy through norms. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agents Systems*, pages 674–681, 2001.

[Duru *et al.*, 1998] Nevcihan Duru, Tarik Duru, e Nurettin Abut. Fuzzy logic based noise reduction of digitally recorded speech signal. In *Proceedings of the 1998 ACM Symposium on Applied Computing*, 1998.

[eBay Inc., 1995] eBay Inc. ebay - new & used electronics, cars, apparel, collectibles, sporting goods & more at low prices. `http://www.ebay.com/`, 1995.

[FIPA, 2002a] FIPA. Fipa abstract architecture specification, 2002. `http://www.fipa.org/specs/fipa00001/`.

[FIPA, 2002b] FIPA. Fipa acl message structure specification, 2002. `http://www.fipa.org/specs/fipa00061/`.

[FIPA, 2002c] FIPA. Fipa communicative act library specification, 2002. `http://www.fipa.org/specs/fipa00037/`.

[FIPA, 2002d] FIPA. Fipa request interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00026/`.

[Foner, 1993] Leonard N. Foner. What's an agent anyway? - A sociological case study. Technical report, MIT Media Lab, May 1993.

[Foner, 1997] Leonard N. Foner. Entertaining agents: A sociological case study. In W. Lewis Johnson e Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 122–129, Marina del Rey, CA, USA, 1997. ACM Press.

[Franklin e Graesser, 1997] Stan Franklin e Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, 1997.

[Galliers, 1989] Julia Rose Galliers. A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict. Technical Report UCAM-CL-TR-172, University of Cambridge Computer Laboratory, 1989.

[Gan *et al.*, 2004] Long Gan, Jiming Liu, e Xiaolong Jin. Agent-based, energy efficient routing in sensor networks. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 472–479, 2004.

[Garcia e Sichman, 2003] Ana Cristina Bicharra Garcia e Jaime Simão Sichman. *Agentes e sistemas multiagentes*, chapter 11, pages 269–306. Manole, Barueri, SP, 2003.

[Genesereth e Ketchpel, 1994] Michael R. Genesereth e Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7), 1994.

[Graham-Cumming, 2004] John Graham-Cumming. POPFile version 0.22.2. `http://popfile.sourceforge.net/`, 2004.

[Grobel e Hienz, 1996] Kirsti Grobel e Hermann Hienz. Fuzzy video-based handshape recognition. In *Proceedings of the 1996 ACM Symposium on Applied Computing*, 1996.

[Guo e Müller, 2004] Yutao Guo e Jörg P. Müller. Multiagent collaborative learning for distributed business systems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1156–1163, 2004.

[Harris, 2003] David Harris. Mercury/32 Mail Transport System version 4.01a. `http://www.pmail.com`, 2003.

[Hassine *et al.*, 2004] Ahlem Ben Hassine, Xavier Défago, e Tu Bao Ho. Agent-based approach to dynamic meeting scheduling problems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1132–1139, 2004.

[Hayes-Roth, 1995] Barbara Hayes-Roth. An architecture for adaptive intelligent systems. In *Artificial Intelligence: Special Issue on Agents and Interactivity*, volume 72, pages 329–365, 1995.

[Herrmann *et al.*, 1996] Christoph S. Herrmann, Steffen Hölldobler, e Antje Strohmaier. Fuzzy conceptual knowledge processing. In *Proceedings of the 1996 ACM Symposium on Applied Computing*, 1996.

[Hosmer, 1993] Hilary. H. Hosmer. Security is fuzzy!: Applying the fuzzy logic paradigm to the multipolicy paradigm. In *Proceedings on the 1992-1993 Workshop on New Security Paradigms*, 1993.

[Huhns e Singh, 1998] Michael N. Huhns e Munindar P. Singh. *Readings in Agents*. Morgan Kaufmann Publishers, 1998.

[IBM, 1997] IBM. The IBM agent. `http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm`, 1997. International Business Machines Corporation.

[Jang *et al.*, 1997] Jyh-Shing Roger Jang, Chuen-Tsai Sun, e Eiji Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice Hall, 1997.

[Jang, 1993] Jyh-Shing Roger Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665–684, 1993.

[Keller *et al.*, 2004] Philipp W. Keller, Felix-Olivier Duguay, e Doina Precup. Redagent-2003: An autonomous, market-based supply-chain management agent. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1182–1189, 2004.

[Kollingbaum e Norman, 2002] Martin J. Kollingbaum e Timothy J. Norman. Supervised interaction — Creating a web of trust for contracting agents in electronic environments. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 272–279, 2002.

[Kulkarni e Muniganti, 1996] A. D. Kulkarni e V. K. Muniganti. Fuzzy neural network models for clustering. In *Proceedings of the 1996 ACM symposium on Applied Computing*, 1996.

[Lee *et al.*, 2004] G. Lee, S. Bauer, P. Faratin, e J. Wroclawski. Learning user preferences for wireless services provisioning. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 480–487, 2004.

[Lopes, 2001] Anderson Araújo Lopes. AgILE: Um modelo de agentes inteligentes para leilão eletrônico. Master's thesis, Universidade Federal Fluminense, 2001.

[Maes, 1994] Pattie Maes. Agents that reduce work and information overload. In *Communications of the ACM*, volume 37 of *7*, pages 31–40. ACM Press, July 1994.

[Maes, 1995] Pattie Maes. Artificial life meets entertainment: Life like autonomous agents. In *Communications of the ACM*, volume 38 of *11*, pages 108–114. ACM Press, 1995.

[Mamdani, 1974] Ebrahim H. Mamdani. Applications of fuzzy algorithms for simple dynamic plant. In *Proc. IEEE*, volume 121, pages 1585–1588. IEEE, 1974.

[Mao e Gratch, 2004] Wenji Mao e Jonathan Gratch. Social judgment in multiagent interactions. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 210–217, 2004.

[Meyer *et al.*, 1993] Chris Meyer, Abraham Kandel, e Dewey Rundus. The triad of fuzzy theory. *SIGAPP Appl. Comput. Rev.*, 1(2):12–15, 1993.

[Microsoft, 2003] Microsoft. Microsoft Agent. `http://www.microsoft.com/msagent/`, 2003. Microsoft Corporation.

[Mueller, 2000] Scott Hazen Mueller. What is spam? `http://spam.abuse.net/overview/whatisspam.shtml`, 2000.

[Mueller, 2005] Scott Hazen Mueller. Spam e-mail blocking and filtering. `http://spam.abuse.net/userhelp/`, 2005.

[Myers e Rose, 1996] John G. Myers e Marshall T. Rose. Post Office Protocol - Version 3. Technical report, Carnegie-Mellon University and Dover Beach Consulting, Inc., 1996. RFC1939.

[Parsons *et al.*, 1999] Simon Parsons, Ola Pettersson, Alessandro Saffiotti, e Michael J. Wooldridge. Robots with the best of intentions. In M. Wooldridge e M. Veloso, editors, *Artificial Intelligence Today*, number 1600 in LNAI, pages 329–338. Springer-Verlag, Berlin, DE, 1999.

[Peters *et al.*, 2005] Tim Peters, Gary Robinson, e Rob Hooft. Spambayes version 1.0.3. `http://spambayes.sourceforge.net/`, 2005.

[Pujol *et al.*, 2002] Josep M. Pujol, Ramon Sangüesa, e Jordi Delgado. Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agents Systems*, pages 467–474, 2002.

[Resnick, 2001] P. Resnick. Internet Message Format. RFC 2822 (Proposed Standard), April 2001.

[Rosenschein e Genesereth, 1985] Jeffrey S. Rosenschein e Michael R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, Los Angeles, California, August 1985.

[Russel e Norvig, 1995] Stuart Russel e Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[Russell e Wefald, 1991] Stuart Russell e Eric Wefald. *Do the right thing: Studies in limited rationality*. Artificial intelligence. MIT Press, Cambridge, Mass, 1991.

[Saeed *et al.*, 1992] Faisel Saeed, K. M. George, e Huizhu Lu. Image contrast enhancement using fuzzy set theory. In *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, 1992.

[Sedbrook, 1998] Tod A. Sedbrook. A collaborative fuzzy expert system for the web. *SIGMIS Database*, 29(3), 1998.

[Smith *et al.*, 1994] David Canfield Smith, Allen Cypher, e Jim Spohrer. Kidsim: Programming agents without a programming language. In *Communications of the ACM*, volume 37 of *7*, pages 55–67. ACM Press, 1994.

[Stegmaier-Stracca e Tschichold-Gürman, 1995] Peter A. Stegmaier-Stracca e Nadine N. Tschichold-Gürman. Cough detection using fuzzy classification. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, 1995.

[Subtil *et al.*, 1996] Pascal Subtil, Noureddine Mouaddib, e Odile Foucaut. A fuzzy information retrieval and management system and its applications. In *Proceedings of the 1996 ACM Symposium on Applied Computing*, 1996.

[Sugeno e Kang, 1988] M. Sugeno e G. T. Kang. Structure identification of fuzzy model. In *Fuzzy Sets and Systems*, volume 28, pages 15–33, 1988.

[Takagi e Sugeno, 1985] T. Takagi e M. Sugeno. Fuzzy identification of systems and its application to modeling and control. In *IEEE Trans. Syst.*, volume SMC-15, pages 116–132. Man. Cybern., 1985.

[Tan e Tang, 2000] K. K. Tan e K. Z. Tang. Simulation of an evolutionary tuned fuzzy dispatching system for automated guided vehicles. In *Proceedings of the 32nd Conference on Winter Simulation*, 2000.

[Tunstel, 1995] Eddie Tunstel. Fuzzy spatial map representation for mobile robot navigation. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, 1995.

[Vivacqua, 1997] Adriana Santarosa Vivacqua. MultiADD: Uma extensão ao modelo ADD para viabilizar design em grupo interdisciplinar. Master's thesis, Universidade Federal Fluminense, 1997.

[Wooldridge e Jennings, 1994] Michael J. Wooldridge e Nicholas R. Jennings. Intelligent agents: Theory and practice. `http://www.csc.liv.ac.uk/~mjw/pubs/ker95/ker95-html.html` (Hypertext version of Knowledge Engineering Review paper), 1994.

[Wooldridge e Jennings, 1995] Michael J. Wooldridge e Nicholas R. Jennings. *Agent Theories, Architectures, and Languages: A Survey*, pages 1–22. Springer-Verlag, Berlin, 1995.

[Wooldridge, 1998] Michael J. Wooldridge. Agent-based computing. *Interoperable Communication Networks*, 1(1):71–97, 1998.

[y López e Luck, 2004] Fabiola López y López e Michael Luck. Normative agent reasoning in dynamic societies. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 732–739, 2004.

[Yu e Singh, 2003] Bin Yu e Munindar P. Singh. Detecting deception in reputation management. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agents Systems*, pages 73–80, 2003.

[Zadeh, 1965] Lotfi Asker Zadeh. Fuzzy sets. *Information and Control*, 3(8):338–353, 1965.

[Zheng e Kainz, 1999] Ding Zheng e Wolfgang Kainz. Fuzzy rule extraction from GIS data with a neural fuzzy system for decision making. In *Proceedings of the Seventh ACM International Symposium on Advances in Geographic Information Systems*, 1999.