ISSN 2177-2096

WORKSHOP-ESCOLA

de Sistemas de Agentes, seus Ambientes e Aplicações

Proceedings WESAAC 2014

Oitavo Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações

28 - 30 de Maio 2014
Porto Alegre
Rio Grande do Sul

Editores:
Viviane Torres da Silva (UFF)
Rafael H. Bordini (PUCRS)
Felipe Meneguzzi (PUCRS)

Patrocínio:

Organização:

W926 Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (8. : 2014 : Porto Alegre, RS)
Anais do VIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações — VIII WESAAC / Viviane Torres da Silva, Rafael H. Bordini, Felipe Meneguzzi, (Eds.) – Porto Alegre : PUCRS, 2014

220 p., il.

1. Agentes. 2. Sistemas multi-agentes. 3. simulações. 4. Aplicaçõe s. 5. Congresso. I. Silva, Viviane Torres da II. Bordini, Rafael H. III. Meneguzzi, Felipe. IV. WESAAC (8. : 2014 : Porto Alegre, RS). V. Título

CDD 006.3

I. Artigos Completos

II. Pôsters

Integrando agentes AgentSpeak(L) em ambientes pervasivos educacionais

Alexandre de O. Zamberlan[1], Reiner F. Perozzo[2], Guilherme C. Kurtz [2] Giovani R. Librelotto[3], Solange B. Fagan[1]

[1]Programa de Pós-Graduação em Nanociências
Centro Universitário Franciscano

[2]Curso de Ciência da Computação
Centro Universitário Franciscano
97010-491 – Santa Maria – RS

[3]Centro de Tecnologia – Departamento de Eletrônica e Computação
Universidade Federal de Santa Maria
97105-900 – Santa Maria – RS

{alexz,reiner.perozzo,guilhermekurtz}@unifra.br

librelotto@inf.ufsm.br, solange.fagan@gmail.com

Abstract. *This paper aims to approach the cognitive agents from BDI theory to the area pervasive computing. This approximation will be given by mapping the behavioral specification of these agents in the language AgentSpeak(L) in a pervasive environment. We present an example in the educational context, i.e, an architecture of intelligent agents embedded in a pervasive environment, which assume the roles multimedia projector, automated blinds, audio system and lighting system, and integration of these devices.*

Resumo. *Este artigo propõe aproximar a teoria BDI de agentes cognitivos, especificados e implementados em AgentSpeak(L) e em seu interpretador Jason, com a área de Computação Pervasiva, por meio de um mapeamento de especificações comportamentais desses agentes num ambiente pervasivo. Busca-se apresentar uma situação exemplo baseada em dinâmica educacional, ou seja, apresentar uma arquitetura contendo agentes inteligentes que sejam associados a um ambiente pervasivo com dispositivos como projetor multimídia, persianas automatizadas, sistema de iluminação e sistema de áudio de uma sala de aula, de forma que esses dispositivos seriam integrados gerando um ambiente inteligente.*

1. Introdução

A computação está intrínseca no cotidiano humano, seja em residências, hospitais, trânsito, escolas, etc. Caminha-se, certamente, para um mundo em que computadores, em um ambiente qualquer, estarão voltados para tarefas específicas, mas interagindo uns com os outros para realizar ações em prol de um usuário. Há fortes indícios que movimentos estão ocorrendo para transformar espaços físicos em espaços computacionais inteligentes, onde os computadores podem oferecer serviços aos usuários não importando o local e nem a hora.

Um ambiente educacional equipado com dispositivos espalhados numa sala de aula poderia conferir características pervasivas. Esses equipamentos serviriam para projeção de *slides* em sistema multimídia e anotações em um quadro branco eletrônico. O ambiente integraria esses dispositivos, oferecendo suporte aos professores e aos alunos em suas atividades de aula.

De acordo com alguns dos principais autores da área, [Weiser 1991] e [Saha and Mukherjee 2003], assume-se que a Computação Pervasiva é a computação disponível em qualquer lugar, a qualquer tempo, e que o usuário possa usar qualquer dispositivo para ter acesso ao seu ambiente computacional. Sabendo-se, também, que as características básicas de agentes inteligentes são autonomia, mobilidade e proatividade, acredita-se que isso possa motivar a aproximação dessas duas áreas. Aproximar a teoria BDI (*Belief*, *Desire*, *Intention*) de agentes e Sistemas Multiagentes cognitivos no contexto da Computação Pervasiva pode ser concretizada pela integração da linguagem de especificação AgentSpeak(L) e seu ambiente de interpretação Jason com uma situação exemplo de Computação Pervasiva.

As aplicações pervasivas devem ser proativas, isto é, precisam identificar o que o usuário deseja e providenciar ações devidas no momento correto. Na computação pervasiva, o sistema se utiliza do contexto formado pelas informações relacionadas ao ambiente, seja por meio de sensores, seja pela geração ou envio a partir de algum dispositivo. Para que aplicações pervasivas funcionem adequadamente, é essencial que percepção de contexto e execução de tarefas sejam características básicas [Helal et al. 2005]. Segundo [Pereira and Librelotto 2012], essa percepção do contexto e a capacidade de utilizá-lo para executar tarefas são o que caracterizam uma aplicação sensível ao contexto (*context-awareness*) e também um sistema multiagente, conforme [Bordini et al. 2007]. Dessa forma, em termos práticos, o contexto englobaria sistema de áudio, sistema de vídeo, persianas automatizadas, projetor multimídia, *smartphones*, entre outros, sendo cada um desses dispositivos representados por agentes inteligentes.

Assim, propõe-se neste trabalho uma forma de realizar o mapeamento de especificações comportamentais de agentes cognitivos baseados no modelo BDI presentes em dispositivos de um ambiente pervasivo, com a finalidade de estender os comportamentos inteligentes desses agentes no ambiente proposto. Esse modelo de agentes é bastante estudado e encontra-se consolidado, por isso acredita-se que seja possível a aproximação com a área de Computação Pervasiva.

Para auxiliar no entendimento do artigo, o texto foi dividido em seções. A Seção 2 aborda o contexto de Computação Pervasiva: noções básicas e requisitos para se projetar um ambiente pervasivo. A Seção 3 trata dos conceitos sobre Sistemas Multiagentes, a teoria BDI de agentes inteligentes e suas tecnologias de especificação e implementação. A Seção 4 apresenta alguns trabalhos relacionados. A Seção 5 mostra a ideia geral da proposta deste artigo, bem como um estudo de caso no contexto educacional. Finalmente, algumas considerações e as referências bibliográficas do trabalho.

2. Computação Pervasiva

A pesquisa na área de Computação Pervasiva trabalha a ideia de que Computação Ubíqua é a computação que se move para fora das estações de trabalho e computadores pessoais e torna-se pervasiva no cotidiano do ser humano. A Computação Pervasiva possui como

referência a obra de [Weiser 1991]. Marc Weiser escreveu há mais de vinte anos que, no futuro, computadores estariam presentes nos mais simples objetos: etiquetas de roupas, xícaras de café, interruptores de luz, canetas, etc, de forma invisível para o usuário.

Os termos Computação Pervasiva, Computação Ubíqua e Computação Móvel são, muitas vezes, utilizados como sinônimos em alguns textos, entretanto, em [Lyytinen and Yoo 2002] e [Soldatos et al. 2007], mostra-se que eles são conceitualmente diferentes. A Computação Ubíqua integra mobilidade em larga escala com a funcionalidade da Computação Pervasiva, que define a estrutura do ambiente[Soldatos et al. 2007]. Segundo [Lyytinen and Yoo 2002], a Computação Móvel baseia-se na capacidade de mover fisicamente serviços computacionais, ou seja, o computador torna-se um dispositivo sempre presente, possibilitando ao usuário utilizar serviços que um computador oferece independentemente de sua localização. Contudo, esse recurso possui limitações, pois um dispositivo não é capaz de obter, de forma flexível, informação sobre o contexto em que a computação ocorre e ajustá-la corretamente. Já a Computação Ubíqua aproveita-se dos avanços da Computação Móvel e da Computação Pervasiva. A Computação Ubíqua surge da necessidade de integrar mobilidade com a funcionalidade da Computação Pervasiva, isto é, qualquer dispositivo computacional em movimento com um usuário pode construir, dinamicamente, modelos computacionais dos ambientes e configurar seus serviços dependendo da necessidade [Araújo 2003].

Esses autores também acreditam que o potencial de aplicações da Computação Ubíqua é limitado apenas pela imaginação, ou seja, com a conectividade, monitoramento e a coordenação de dispositivos localizados em casas, edifícios e carros inteligentes, através de redes sem fio (locais ou longa distância), a aplicação seria, por exemplo, para o controle de temperatura, luzes e umidade de uma residência, ou aplicações colaborativas com suporte à mobilidade. Estudos em Computação Ubíqua estão sendo realizados por pesquisadores do mundo todo, que vão de protótipos de rede que provêem acesso básico a qualquer tipo de dispositivo sem fio de forma transparente, segura, com tratamento de contexto, compressão, entrega e apresentação de conteúdo multimídia, até a adaptação da aplicação e da apresentação multimídia aos dispositivos do usuário [Araújo 2003].

Finalmente, segundo [Helal et al. 2005], a operacionalização de um ambiente pervasivo depende basicamente de três requisitos fundamentais:

- um sistema de automação contendo sensores, atuadores, controladores e interfaces humano-computador;
- uma infraestrutura de comunicação de dados para troca de informações entre dispositivos e entre usuários desse ambiente;
- sistemas inteligentes, cientes de contexto, adaptáveis, evolucionários e capazes de atender as necessidades dos usuários em suas atividades diárias, oferecendo suporte inteligente.

3. Sistemas Multiagentes

A construção de ambientes inteligentes representados por entidades de software com comportamentos sofisticados, com intensa interação e compartilhamento de recursos e objetivos, faz com que Sistemas Multiagentes sejam aplicados em diferentes propósitos. Segundo [Bordini and Hübner 2005], um agente é um software que age em prol do usuário ou de outro programa, obedecendo a regras de relacionamento do ambiente em que está

inserido. Tais ações comportamentais são decididas quando melhor lhe convir. Essa ideia é que agentes não são somente invocados exclusivamente por uma tarefa, mas também ativados por decisão própria. Existem diversos tipos de agentes, entre eles os cognitivos, que podem ser baseados em estados mentais, com capacidade de raciocínio, ou seja, capazes de obedecer um plano, ou planos, de ações que os levam a um estado pretendido. Esse tipo de agente possui características particulares como [Bordini and Hübner 2005] e [Vieira et al. 2006]: autonomia funcional; encontra-se continuamente em funcionamento; é sociável, ou seja, possui capacidade de interação; o mecanismo de controle é deliberativo; possui memória; e as sociedades são formadas por poucos agentes.

As estruturas desses agentes são constituídas por componentes mentais, como crenças, desejos, capacidades, escolhas e compromissos. Dentro das arquiteturas baseadas em estados mentais, encontra-se a abordagem de BDI, que tem o nome atribuído aos estados mentais: crenças (*beliefs*), desejos (*desires*) e intenções (*intentions*). Essa arquitetura representa seus processos internos através desses estados mentais e define um mecanismo de controle que seleciona de maneira racional o curso das ações [Bordini et al. 2007].

O principal enfoque dos Sistemas Multiagentes é promover mecanismos que possibilitem a criação de sistemas computacionais a partir de entidades de softwares autônomas que interagem através de um ambiente compartilhado por todos agentes de uma sociedade, e sobre o qual os agentes atuam alterando seu estado [Vieira et al. 2006]. Entre as tecnologias existentes para especificar e implementar agentes BDI, há a linguagem de implementação, AgentSpeak(L) e o interpretador Jason que possibilita a comunicação entre agentes [Bordini and Hübner 2005].

3.1. A Linguagem AgentSpeak(L) e o interpretador Jason

De acordo com [Bordini and Hübner 2005] e [Vieira et al. 2006], em AgentSpeak(L), um agente corresponde à especificação de um conjunto de crenças que formarão a base de crenças iniciais e um conjunto de planos. A base crenças de um agente é formada pela coleção de átomos de crenças e literais de crença: um átomo de crença forma um predicado de primeira ordem na notação lógica usual; as literais de crença são formadas por átomos de crenças ou suas negações. Os planos fazem referência a ações básicas que um agente é capaz de executar em seu ambiente. Essas ações são definidas por atributos com símbolos predicativos especiais (símbolos de ação) usados para atingir ações de produtos predicados. Um plano é formado por um evento ativador (propósito do plano), seguido de uma conjunção de literais de crença que representam um 'contexto'. O contexto deve ser conseqüência lógica do conjunto de crenças do agente no momento em que o evento é selecionado pelo agente para o plano ser considerado 'aplicável'. O restante do plano é seqüência de ações básicas ou sub-objetivos que o agente deve atingir ou testar quando uma instância do plano é selecionada para execução [Bordini and Hübner 2005].

Jason é um interpretador para uma extensão da linguagem AgentSpeak(L), que possibilita comunicação entre agentes baseada na teoria de atos de fala, por meio de diretivas de comunicação. A ferramenta Jason é implementada em Java (multiplataforma) e disponibilizada como código aberto sob a licença GNU LGPL [Bordini et al. 2007]. Além de interpretar a linguagem AgentSpeak(L) original, essa ferramenta possui características, como por exemplo [Bordini and Hübner 2005]: tratamento de falha em planos;

suporte para o desenvolvimento de ambientes (o ambiente é programado em Java); possibilidade de executar o SMA distribuidamente em uma rede; possibilidade de especializar em Java as funções de seleção de planos, e toda a arquitetura do agente; possui um ambiente de desenvolvimento, utilizando o jEdit (editor de texto de código aberto), ou em forma de um *plugin* para a IDE Eclipse.

4. Trabalhos correlatos

Em relação aos trabalhos correlacionados, foram encontrados trabalhos que de alguma forma propuseram construções de ambientes pervasivos, ou que utilizaram tecnologias que possibilitariam o desenvolvimento desses ambientes.

No trabalho de [Gárate et al. 2005], foi proposto uma interface na forma de mordomo (como um agente) que faz a comunicação dos usuários com os dispositivos do ambiente pervasivo. Em [da Silva et al. 2008], foi apresentado o modelo OCtoPUS que realiza a categorização de perfis de usuários baseada em seus comportamentos em um ambiente ubíquo. O modelo proposto utiliza tecnologias de web semântica para representar o conhecimento dos usuários.

A proposta SemantiCore Mobile [Escobar et al. 2007] permite o desenvolvimento de aplicações multiagentes em dispositivos móveis. Nela, foi abordado o uso de agentes e suas características em dispositivos móveis por meio do *framework* SemantiCore. Já [Wolski 2009] introduziu localização na plataforma Semanticore para criação de aplicações pervasivas baseadas em agentes de software, apresentando uma proposta de extensão da plataforma para que gere aplicações pervasivas orientadas a agentes.

Finalmente, [Junior et al. 2012] constatou que aplicações colaborativas vêm sendo utilizadas na integração de dispositivos computacionais em sala de aula. Entretanto, essa implementação expõem o programador a problemas de nível de sistema, tais como comunicação e concorrência. Dessa forma, esse trabalho buscou tornar os problemas transparentes ao desenvolvedor através de primitivas de compartilhamento de conteúdo, por meio de definições de operações de movimentação de conteúdo, tais como, mover, clonar e espelhar, que servem como um modelo de comunicação de alto nível para a construção dessas aplicações.

As obras citadas, de certa forma, trabalham com a ideia de agentes inteligentes ou classificação de perfis para a construção de ambientes pervasivos. Em [Wolski 2009], o autor afirma que existem várias plataformas para o desenvolvimento de sistemas multiagentes (SMAs) como JADE, Jason e SemantiCore e, nenhuma delas possui características para geração de aplicações pervasivas orientadas a agentes. Por essa outra situação, surge a motivação da realização deste estudo. Além do que, nenhum desses trabalhos utilizou as tecnologias AgentSpeak(L) e Jason.

5. Mapeamento de Agentspeak(L) para ambientes pervasivos: estudo de caso

Um ambiente educacional com características pervasivas poderia, por exemplo, ser composto por alguns dispositivos inseridos numa sala de aula, como para projeção de slides em sistema multimídia, anotações em um quadro branco eletrônico, o uso ou não de microfone e caixas de som. Esse ambiente seria formado por sistemas de iluminação e áudio, de projeção multimídia, de controle de persianas, de posicionamento (levantamento ou

rebaixamento) da tela de projeção e/ou quadro branco e de captação e distribuição de material produzido pelo professor (*slides*, textos, vídeos, etc). A integração desses sistemas daria suporte aos professores em suas atividades de reprodução de material didático e aos alunos na anotação das aulas de forma personalizada.

Numa visualização empírica dessa aplicação pervasiva, sempre que o professor utilizasse o quadro branco eletrônico, o sistema registraria os desenhos feitos e disponibilizaria essa informação sob a forma de hiperdocumentos multimídia sincronizados aos alunos conectados presentes na sala. Além disso, quando o professor chegasse em sala, com sua apresentação armazenada em seu *smartphone*, o ambiente se adequaria à exibição desse material, ou seja, as persianas e a tela de projeção seriam baixadas (e o quadro branco levantado), o sistema multimídia receberia o arquivo contendo a apresentação e entraria em modo de projeção, o sistema de áudio entraria em prontidão. Por fim, o professor poderia configurar o arquivo da apresentação para que fosse enviado, imediatamente, a todos os alunos em sala que estivessem conectados.

Sendo assim, faz-se necessário a descrição da arquitetura conceitual desse ambiente, constituída de seu *layout*, componentes, tecnologias, entre outros.

5.1. Arquitetura proposta

A arquitetura está orientada ao gerenciamento de serviços autônomos em ambientes de Computação Pervasiva, sendo que os agentes fariam o gerenciamento de cada um desses serviços. A arquitetura é constituída pelos componentes que seguem, conforme ilustrado na Figura 1.

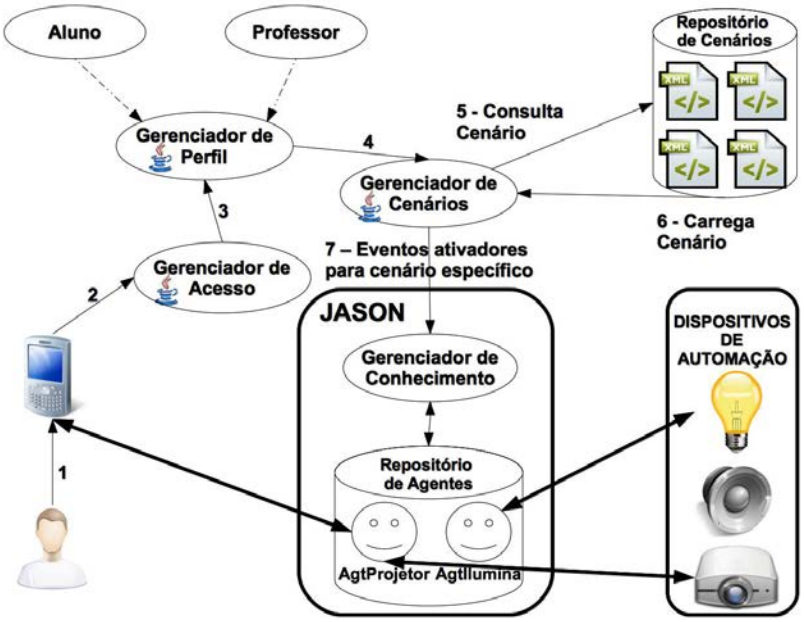


Figura 1. Ambiente Pervasivo proposto - conceitual.

- Gerenciador de Acesso: recebe e trata as conexões de acesso dos dispositivos móveis cadastrados;
- Gerenciador de Perfil: uma vez realizada a conexão, deve-se carregar o perfil do usuário conectado, seja ele aluno ou professor. Dependendo do perfil, haverá

recursos e permissões diferenciados, como por exemplo, enviar um arquivo ou acionar um dispositivo de automação;
- Gerenciador de Cenários: responsável por consultar e carregar os cenários do repositório de cenários conforme o perfil do usuário conectado, dos recursos e permissões associados a ele;
- Repositório de Cenários: cada cenário é um arquivo XML contendo a descrição de agentes e nome dos eventos ativadores (referentes ao dispositivo de automação específico);
- Gerenciador de Conhecimento: faz a relação dos cenários carregados e ações a serem realizadas pelos agentes;
- Repositório de Agentes: contém as implementações de todos os agentes (aspectos estruturais e funcionais) que representam os dispositivos de automação. Neste componente que a linguagem AgentSpeak(L) e o interpretador Jason se fazem presentes. A linguagem para especificar os comportamentos dos agentes, baseados em planos ativados por eventos externos ou eventos internos (de comunicação entre os agentes), tudo isso operacionalizado e integrado pelo interpretador Jason;
- Dispositivos de Automação: lâmpadas, projetor multimídia, caixas de som, persiana, quadro branco, etc.

Por fim, destacam-se, neste trabalho, os componentes Gerenciador de Cenários e Gerenciador de Conhecimento, isso porque os eventos serão disparados pelo Gerenciador de Cenários (carregados do Repositório de Cenários) para o Gerenciador de Conhecimento que se relaciona com o Repositório de Agentes. O agente conhece o protocolo de comunicação (via Jason) e implementa a interface de acesso ao Dispositivo de Automação.

5.2. Tecnologias

Em [Araújo 2003], é feita uma análise sobre dispositivos essenciais para a construção de ambientes pervasivos. Entre os citados, destacam-se controles, sensores e atuadores para residências e automóveis, eletrodomésticos, ar-condicionado, aquecedor, relógios e etiquetas inteligentes, além de toda a linha branca de utensílios domésticos, TVs, celulares, PDAs, consoles de jogos e muito mais. Esses dispositivos são categorizados como controles inteligentes, utensílios inteligentes, dispositivos de acesso à informação ou sistemas de entretenimento. Neste trabalho foca-se em controles inteligentes, que, por serem muito pequenos, podem ser integrados a lâmpadas, interruptores, sensores e atuadores. Essas aplicações variam de sensores (por exemplo, em portas para detectar a entrada de pessoas e seus *smartphones*) e atuadores (tais como, para acender/apagar lâmpadas específicas) à controle de projeção (sensores e atuadores para ligar/desligar/programar os sistemas de multimídia e áudio da sala de aula). Os controles são conectados a redes domésticas e gerenciados local ou remotamente.

Em relação à implementação de aplicações para ambientes pervasivos, há uma relação limitada de linguagens de programação disponíveis, como Java, C e C++, além de algumas linguagens proprietárias. A linguagem Java oferece independência de plataforma do código compilado e atende aos requisitos exigidos por dispositivos pequenos.

Os dispositivos de um sistema pervasivo processam os códigos executáveis e seus respectivos dados direto de sua localização em memórias RAM ou ROM. Ao desenvolver aplicações para dispositivos nesses ambientes, os programadores devem levar em

consideração algumas restrições, como capacidade limitada de entrada de dados, processamento limitado, memória, armazenamento persistente e vida da bateria, alta latência, largura de banda limitada e conectividade intermitente. A solução estratégica para o desenvolvimento de aplicações para dispositivos pequenos é remover características desnecessárias. E, se possível, tornar essas características como uma aplicação secundária e separada, pois aplicações menores usam menos memória no dispositivo e requerem menos tempo de instalação. Além disso, mover as tarefas de maior processamento e consumo para o servidor, permite que o dispositivo móvel trate a interface e utilize o mínimo de computação.

Dessa forma, a tecnologia Java seria uma opção para implementar o sistema pervasivo, principalmente por meio do interpretador Jason, que, como já mencionado, possibilita o desenvolvimento de ambientes, executa o sistema multiagente distribuidamente em uma rede e específica na linguagem Java as funções de seleção de planos e a arquitetura do agente.

5.3. Especificação AgentSpeak(L)

Para a modelagem ou especificação do Sistema Multiagente proposto, utilizou-se a metodologia Prometheus, que esta dividida em três etapas. A primeira etapa, tida como especificação do sistema, concentra-se em identificar os objetivos e as funcionalidades básicas do sistema, além das entradas (percepções) e saídas (ações). A segunda, Projeto Arquitetural, utiliza as saídas da etapa anterior para determinar quais os tipos de agentes que o sistema irá conter, e como eles irão interagir entre si. E a última, Projeto Detalhado, tem como objetivo modelar e detalhar a arquitetura interna dos agentes, e como eles deverão cumprir as suas tarefas [Padgham and Winikoff 2004].

Na Figura 2 é possível visualizar a legenda com os elementos básicos da diagramação em Prometheus, como atores, agentes, cenários, objetivos, planos, percepções, mensagens, base de dados e protocolo de comunicação.

Figura 2. Legenda Prometheus.

Na fase de especificação do sistema, assume-se dois principais objetivos do Sistema Multiagente, que são ”Professor chega na sala”e ”Professor chega na sala com arquivo para apresentação”, além dos subobjetivos atrelados. Na Figura 3 é possível observar as percepções ”Chegada de professor”e ”Chegada de arquivo marcado para apresentação”, que são eventos ativadores das ações como ”Ligar luzes para projeção”, ”Baixar cortinas”, entre outras, todas elas vinculadas aos papéis ”Prepara Sala de Projeção”, ”Distribui Arquivo”e ”Prepara Sala Tradicional”.

Finalmente, na Figura 4, é possível observar os agentes assumindo os papéis do sistema multiagente, percebendo ou sensorando o ambiente e disparando mensagens e/ou planos para que a sala entre ou não em configuração de apresentação, por exemplo.

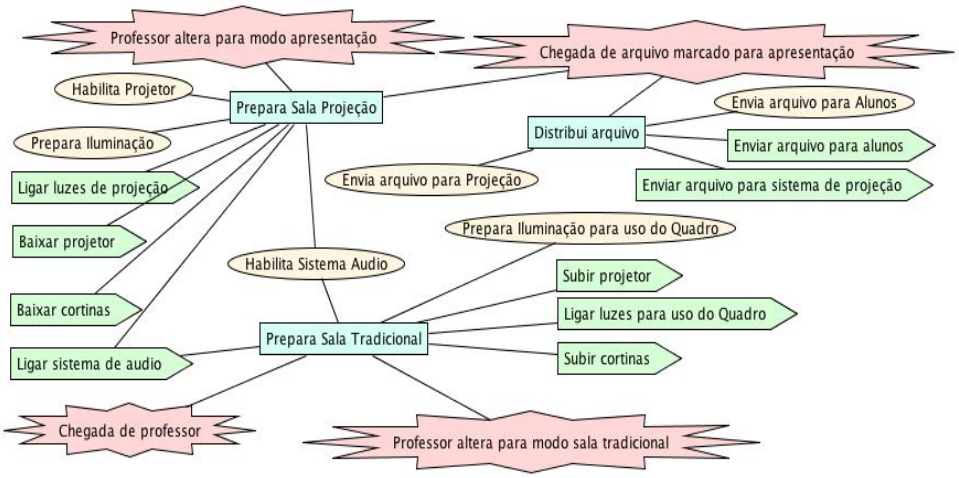


Figura 3. Papéis do Sistema Multiagente.

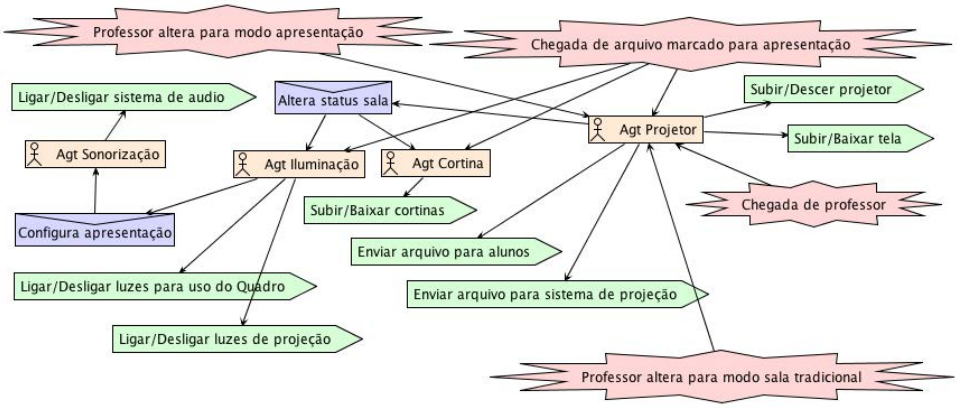


Figura 4. Visão geral do SMA.

5.4. Implementação Jason

Uma vez a modelagem estruturada, parte-se para a implementação do Sistema Multiagente por meio do interpretador Jason. Define-se, então, o projeto do sistema multiagente, como se pode visualizar na Figura 5. No arquivo (.mas2j), definem-se a infraestrutura (centralizada ou distribuída) de funcionamento, o ambiente (.java) e todos os agentes envolvidos no ambiente.

Na Figura 6 é possível observar uma implementação de comportamento do agente ”agtProjetor”, em que há um plano que trata o evento externo ”professor(emSala)”, gerado pelo ambiente. Esse plano só é disparado caso exista o contexto ou a condição para isso, que são as crenças presentes no agente ”agtProjeto”sobre a sala estar livre e o professor estar com arquivo da apresentação. Uma vez o plano tendo contexto, disparam-se ações que devem ser executadas no ambiente, como ”enviarArquivoParaAlunos”e ”enviarArquivoParaProjecao”. Uma atualização na base de crença também é executada, ou seja, retirada da crença ”statusSala(livre)”e adição de nova crença ”statusSala(ocupado)”. Além disso, uma mensagem é encaminhada a todos os agentes para que executem o plano ”ambiente(paraApresentacao)”.

As ações encaminhadas para execução no ambiente são implementadas no arquivo

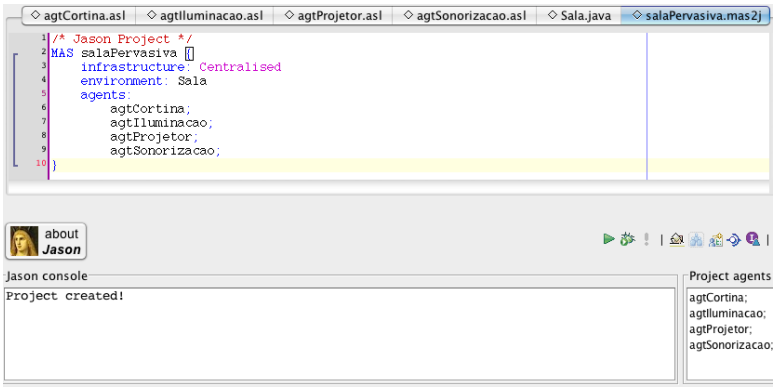


Figura 5. SMA em Jason da Sala Pervasiva.

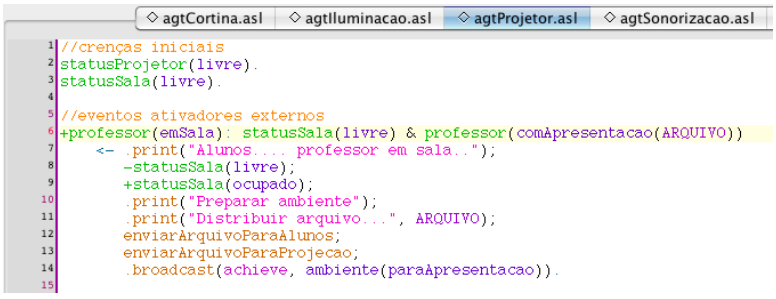


Figura 6. Implementação dos comportamentos do agente projetor.

”Sala.java”, que é o ambiente deste SMA. Na Figura 7, é possível visualizar o método ”*executeAction()*”que trata as ações encaminhadas pelos agentes: ”enviarArquivoPara-Projecao”e ”enviarArquivoParaAlunos”.

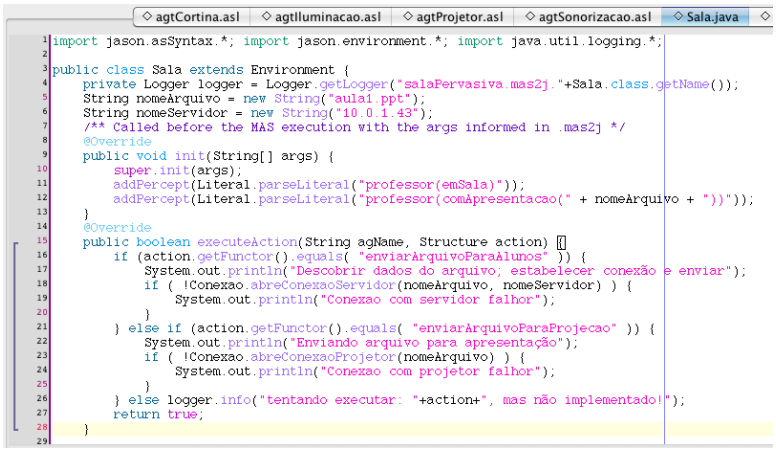


Figura 7. Implementação das ações pelo ambiente.

6. Considerações Finais

Ao longo do texto, buscou-se discutir a Computação Pervasiva relacionada a Sistemas Multiagentes, unindo o melhor das duas áreas. Enquanto a pervasividade agrega a habilidade de perceber o ambiente, agentes somam quanto a autonomia e proatividade. Porém,

ambas possuem o caráter de sistemas reativos (reagindo a eventos no ambiente), flexibilidade, adaptabilidade e mobilidade. Além disso, são capazes de gerar comunicação entre seus elementos. Por isso, acredita-se, com este estudo, que a área de Sistemas Multiagentes pode ser um meio eficaz de operacionalizar a Computação Pervasiva. Acredita-se, também, que o uso da teoria BDI, por meio da linguagem AgentSpeak(L) e seu interpretador Jason, foi fundamental para a integração dessas duas áreas, uma vez que o Jason fornece recursos de comunicação e concorrência entre os agentes.

A área de Sistemas Multiagentes encontra-se consolidada, com metodologias e ferramentas para projeção e implementação de sistemas inteligentes. Entre as diversas tecnologias existentes, destacam-se a linguagem AgentSpeak(L) e o interpretador Jason que, de fato, possibilitam a concretização de sistemas desse tipo. O interpretador, por basear-se na linguagem Java, auxilia sobremaneira quanto a portabilidade e integração dos componentes que compõem o sistema pervasivo, por ser interpretada, é executada nos mais diversos equipamentos de forma distribuída, inclusive.

Como trabalhos futuros, propõe-se um estudo experimental quantitativo para avaliar o desempenho do sistema multiagente construído. Também, está previsto a implementação da integração dos demais processos de gerenciamento da arquitetura proposta para o estudo de caso.

Finalmente, registra-se que com este estudo foi possível verificar a aproximação dessas áreas.

Referências

Araújo, R. B. (2003). Computação ubíqua: Princípios, tecnologias e desafios. In *Computação Ubíqua: Princípios, Tecnologias e Desafios*, volume 1, pages 1–71. SBC, 1 edition.

Bordini, R. H. and Hübner, J. F. (2005). BDI agent programming in agentspeak using Jason (tutorial paper). In *CLIMA*, pages 143–164.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*. Wiley.

da Silva, D. M., Barbosa, J., and Vieira, R. (2008). OCtoPUS: um modelo para classificação de perfil de usuários usando trilhas em sistemas ubíquos. In *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, WebMedia '08, pages 185–188, New York, NY, USA. ACM.

Escobar, M. S., Ries, L., P., L. A., and M., B. (2007). Semanticore mobile - permitindo o desenvolvimento de aplicações multiagentes em dispositivos móveis. In *I Workshop on Pervasive and Ubiquitous Computing (WPUC)*.

Gárate, A., Herrasti, N., and López, A. (2005). GENIO: an ambient intelligence application in home automation and entertainment environment. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, sOc-EUSAI '05, pages 241–245, New York, NY, USA. ACM.

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. (2005). The gator tech smart house: A programmable pervasive space. *Computer*, 38(3):50–60.

Junior, M. R., Freitas, L., Massarani, M. A., da Rocha, R., and Costa, F. (2012). Ucle: Um middleware de computação ubíqua para compartilhamento de conteúdo em salas de aula inteligentes. In *SBCUP - IV Simpósio Brasileiro de Computação Ubíqua e Pervasiva*. SBC.

Lyytinen, K. and Yoo, Y. (2002). Issues and challenges in ubiquitous computing. *Commun. ACM*, 45(12):62–65.

Mohsin, M. and Ahmad, A. (2014). Genetically-encoded nanosensor for quantitative monitoring of methionine in bacterial and yeast cells. *Biosensors and Bioelectronics*, 59:358–364. cited By (since 1996)0.

Padgham, L. and Winikoff, M. (2004). *Developing Intelligent Agent Systems: A practical guide.* John Wiley & Sons.

Pereira, H. and Librelotto, G. (2012). *ARCP: uma arquitetura para a utilização de computação nas nuvens nos ambientes de computação pervasiva.* Amazon.

Saha, D. and Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *Computer*, 36(3):25–31.

Soldatos, J., Dimakis, N., Stamatis, K., and Polymenakos, L. (2007). A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications. *Personal Ubiquitous Comput.*, 11:193–212.

Vieira, R., Moreira, Á. F., Bordini, R. H., and Hübner, J. F. (2006). An agent-oriented programming language for computing in context. In *IFIP PPAI*, pages 61–70.

Weiser, M. (1991). The computer for the 21st century. In Baecker, R. M., Grudin, J., Buxton, W. A. S., and Greenberg, S., editors, *Human-computer interaction*, pages 933–940. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Wolski, L. Z. (2009). Introduzindo localização na plataforma semanticore para criação de aplicações pervasivas baseadas em agentes de software. Master's thesis, PUCRS.

On the Checking of Indirect Normative Conflicts

Jean de Oliveira Zahn, Viviane Torres da Silva

Computer Science Department – Universidade Federal Fluminense (UFF)
24.210-240 – Niterói – RJ – Brazil

{jzahn, viviane.silva}@ic.uff.br

Abstract. *In open multi-agent systems, norms are being used to regulate the behavior of the autonomous, heterogeneous and independently designed agents. Norms describe the behavior that can be performed, that must be performed, and that cannot be performed in the system. One of the main challenges on developing normative systems is that norms may conflict with each other. Two norms are in conflict when the fulfilment of one norm violates the other and vice-versa. The majority works that deal with the checking of normative conflicts are not able to detect conflicts that depend on how the entities are related and how the actions are connected. They are only able to detect conflicts when the two norms regulate the same behavior executed by the same entity. In this paper, we present an approach able to check for conflicts between norms that regulate the execution of different, but related action. We describe four relationships that relate the actions of a domain and present an algorithm for the checking of indirect conflicts, i.e., conflicts between norms that do not govern the behavior of the same entity and/or that do not regulate the same action.*

1. Introduction

In open multi-agent systems, norms have been used to regulate the behavior of autonomous and heterogeneous entities by stating permissions, prohibitions and obligations. Due to the numerous norms that may be necessary to govern the entities of a given system, the identification of conflicts among such norms is one of the main challenges in the area.

A normative conflict arises when the fulfilment of one norm implies on the violation of the other. Several studies propose the identification of normative conflicts and the resolution of such conflicts. However, in the majority, only simple and direct conflicts are detected, i.e., conflicts between norms that govern the same behavior executed by the same entity. These approaches do not detect indirect conflict, i.e., conflicts between norms that govern different but related behavior executed by different but related entities.

The detection of indirect conflicts is only possible when the conflict checker considers the characteristics of the application domain. It is fundamental to figure out the domain-dependent relationships among the entities and the actions identified in the norms when checking for conflicts.

In this paper, we extend our preliminary work presented in [Silva 2013] on the identification of the relationships between entities and actions and on the definition of a normative conflict checker algorithm that consider those relationships. The main extensions presented in this paper are: (i) in the previous version, only norms defining

obligations and prohibitions were considered. In this new version, we also analyzed conflicts including permissions; (ii) due to the inclusion of permission, the two relationships defined in the previous version were extended to consider conflicts among prohibitions and permissions; (iii) two new actions' relationships were defined and the possible conflicts between permissions, prohibitions and obligations that may rise due to such relationships were analyzed; and at least but not last (iv) we present in detail the normative conflict checker algorithm that consider the relationships mentioned in the paper.

The remainder of this paper is divided in 5 sections. Section 2 presents the background material about the definition of norms and the relationships among the entities of the multi-agent system. Section 3 describes four relationships used to link actions and Section 4 presents the proposed conflict checker algorithm. Section 5 describes some related work and, finally, Section 6 states some conclusions and future work.

2. Background

In this section, we present the background material need to understand our approach.

2.1 Norm Definition

According to [Figueiredo et al. 2011] a norm prohibits, permits or obliges an entity to execute an action in a given context during a certain period of time. Several normative specifications, modelling languages, methodologies and organizational models define norms in similar ways. In all of them, a norm is associated with a deontic concept, an entity and an action (or state) that is being regulated.

Definition (Norm): *A norm n is a tuple of the form {deoC, c, e, a, ac, dc, s} where deoC is a deontic concept from the set {obligation, prohibition or permission}, c ∈ C is the context where the norm is defined, e ∈ E is the entity whose behavior is being regulated, a ∈ A is the action being regulated, ac ∈ Cd indicates the condition that activates the norm, dc ∈ Cd is the condition that deactivates the norm and s indicates the state of the norm from the set {fulfilled, violated, none}. None indicates that the norm has not been fulfilled or violated yet.*

The context of a norm indicates the scope where the norm is defined. A norm must be fulfilled only when the entity is executing in such context. Outside its context the norm is not valid. In this paper we consider that a norm can be defined in the context of an organization or of an environment that is the habitat of the entities.

A norm can be defined to regulate the behavior of an agent itself, of an organization (or group of agents) – meaning that all agents playing roles in such organization must fulfil the norm –, or of a role – meaning that all agents playing such role must fulfil the norm.

The activation and deactivation conditions can state an event that can be a date, the execution of an action, the fulfilment of a norm, etc. In this paper, we will focus on the specification of a date.

2.2 Entities Relationship

The four relationships used to relate the entities that are being considered in this paper were defined following [Silva 2013]. In that paper, we describe seven relationships: five relationships between entities (as presented below) and two relationships between actions (that are detailed in Section 3).

Inhabit: it relates an entity to the environment that it inhabits. *If a norm applies in the scope of an environment, such norm applies to all entities that inhabit such environment.*

Play: it relates an entity to the roles that it can play. *If a norm applies to a role, it applies to all agents (or organization) playing such role.*

PlayIn: it relates an entity to the organization where the entity is playing role. *If a norm applies to an organization, it applies to all entities playing roles in such organization.*

Ownership: it describes the roles defined in the scope of an organization. *If a norm applies to an organization, it applies to all roles being played in such organization.*

3. Actions Relationship

In this section, we describe four kinds of relationships that can be used to link actions. In order to exemplify such relationships, let's consider the four very simple examples below:

E.g.1: (*refinement relationship*) to walk and to drive are actions that specialize to move. When the agent is walking or is driving we can say that it is moving.

```
(superaction, subaction, refinement)
(to move, to walk, refinement)
(to move, to drive, refinement)
```

E.g.2: (*composition relationship*) to govern the multi-agent system is an action that implies the execution of three other actions: to find out violations, to find out fulfilments and to apply sanctions.

```
(wholeAction, partAction, composition)
(to govern, to findViolations, composition)
(to govern, to findFulfilments, composition)
(to govern, to applySanctions, composition)
```

E.g.3: (*orthogonal relationship*) to walk and to drive are orthogonal actions that cannot be executed simultaneously by the same or related entities.

```
(action, action, orthogonal)
(to walk, to drive, orthogonal)
```

E.g.4: (*dependency relationship*) to find out violations is a precondition to apply the sanctions. Therefore, we may say that these two actions are related by the dependency relationship as follows:

```
(dependent, client, dependency)
(to applySanctions, to findOutViolations, dependency)
```

3.1. Action Refinement

When the refinement relationship is defined between two actions, there is an action called *subaction* that refines another called *superaction* (that is an abstract action). The execution of the *subaction* achieves the goal of executing the *superaction*, and may also achieve other goals. If there are more than one *subactions* for a given *superaction*, the execution of any *subaction* achieves the goal of executing the *superaction*.

- **Obligation**: If the norm applied to the *superaction* is an obligation, it means that the entity, whose behavior is being regulated by the norm, is obliged to execute the *superaction*.

Fulfillment and violation: If the *superaction* has more than one *subaction* and knowing that the states achieved by the *superaction* are a subset of the states achieved by any *subaction*, when one of the *subactions* is executed (in the period during while the norm is active), the entity fulfills its obligation. In order to illustrate such case, let's consider that there is a norm obligating an entity *to move*. If it *walks* or if it *drives*, it will fulfil the norm.

Conflicts: A conflict between the obligation applied to the *superaction* and the norms applied to the *subactions* will arise only if all the *subactions* are being prohibited.

- **Prohibition**: If the norm applied to the *superaction* is a prohibition, it means that the entity, whose behavior is being regulated by the norm, is prohibited to execute the *superaction* and achieve any of its states.

Fulfilment and violation: If the *superaction* has more than one *subaction* and knowing that the states achieved by the *superaction* are a subset of the states achieved by any of its *subactions*, if the entity executes any *subaction* (in the period during while the norm is active), it will be violating its prohibition. For instance, let's assume that there is a norm prohibiting an entity *to move*. If it *walks* or if it *drives* it will be violating the norm.

Conflicts: The entity whose behavior is being regulated by the prohibition applied to the *superaction* should not execute any of the *subactions* in order to avoid the violation of the prohibition.

- **Permission**: If the norm applied to the *superaction* is a permission, it means that the entity, whose behavior is being regulated by the norm, is permitted to execute the *superaction* and achieve its states.

Fulfillment and violation: By knowing that the states achieved by the *superaction* are, only a subset of the states achieved by any *subaction*, the permission for executing the *superaction* is not granted by the *subactions*. We can say that the entity is partially permitted for executing the *subactions* since it is permitted for achieving only the states related to the execution of the *superaction*. Therefore, if the entity is permitted *to move* it may or not be permitted *to drive* and *to walk*.

Conflicts: A conflict between the permission applied to the *superaction* and the norms applied to the *subactions* will arise only if all the *subactions* are being prohibited.

3.2. Action Composition

If the composition relationship is defined between two actions, it means that there is an action called *part* that is part of the action called *whole* and that the whole action is an abstract action. The states achieved by executing the *whole* action are the union of the states achieved by executing all its *parts*. Therefore, in order to achieve the goals of executing the *whole* action it is necessary to execute all its *parts*.

- **Obligation**: If the norm applied to the *whole* action is an obligation, it means that the entity is obliged to execute the *whole* action and achieve its states.

Fulfillment and violation: If the *whole* action has more than one *part* and knowing that the states achieved by each *part* are a subset of the states achieved by the *whole*, the entity is obliged to execute all its parts (in the period during while the norm is active) in order to fulfill the obligation applied to the *whole action*. If one of the *parts* is not executed, the norm will be violated. If there is a norm obligating an entity of *governing* a MAS, in order to fulfill such norm the entity needs *to find out the violations* and the *fulfillments* and *to apply the sanctions*.

Conflict: If there is a norm prohibiting the execution of any *part action*, a conflict will arise between such norm and the norm applied to the *whole action*.

- **Prohibition**: If the norm applied to the *whole* action is a prohibition, it means that the entity is prohibited to execute the *whole* action and achieve any of its states.

Fulfilment and violation: If the *whole action* has more than one *part*, the agent will fulfil the prohibition if it does not execute one of the *parts* (in the period during while the norm is active). The agent is only violating the prohibition if it executes all the *parts*. For instance, if there is a norm prohibiting an entity of *governing* a MAS, the act of *find out violations* or *fulfilments* or of *applying sanctions* does not violate the norm.

Conflict: Since the violation of a prohibition applied to the *whole* action will only occur if the entity executes all its *parts*, conflicts will only arise if there are norms obligating the entity to execute all the *part actions*.

- **Permission**: If the norm applied to the *whole action* is a permission, it means that the entity is permitted to execute the *whole action* and achieve its states.

Fulfillment and violation: If the *whole action* has more than one *part* and knowing that the states achieved by each *part* are a subset of the states achieved by the *whole*, the entity is also permitted to execute all its *parts* (in the period during while the norm is active). Following the example of *governing* a MAS, if there is a norm permitting an entity of *governing*, it is also permitting the entity *to find out violations* and *fulfilments* and *to apply sanctions*.

Conflicts: Knowing that the permission for executing the *whole action* is propagated to the *part actions*, a conflict will arise if a norm prohibits the execution of any *part action*.

3.3. Action Orthogonal

If an orthogonal relationship is defined between two actions, it means that both actions cannot be executed at the same time by the same or related entities.

- **Obligation**: As stated before, if a norm applied to an action is an obligation, it means that the entity must execute such action in order to fulfill the norm.

Fulfillment and Violation: If there are two obligations whose activation period intersects, that regulate related entities (in the same context) and are applied to orthogonal actions, it means that the fulfillment of one norm will violate the other, and vice-versa. For instance, when an agent is obliged *to walk* and also obliged *to drive* at the same time, if it fulfills one norm it will violated the other since it cannot execute both actions at the same time.

If there is one obligation and one permission applied to orthogonal actions regulating the behavior of related entities (in the same context) in period that intersects,

we may say that there is a potential conflict. For instance, when an agent is obliged *to walk* and permitted *to drive* at the same time, if it *drives*, it will violate the obligation.

Conflict: The conflict between two norms applied to orthogonal actions will occur if one norm obligates the execution of an action and the other obligates or permits the execution of the other actions.

- **Prohibition**: If there are two prohibitions applied to orthogonal actions, it means that the entities (or the related entities) cannot execute those actions.

Fulfillment and violation: In case of orthogonal actions, the fulfillment of a prohibition does not imply in the violation of the other. Both norms can be fulfilled at the same time. For instance, if an entity is prohibited *to walk* and *to drive* at the same time, the fulfillment of the first norm does not imply on the violation of the other.

Conflict: There is no conflict between two norms that prohibit the execution of orthogonal actions.

- **Permission**: If there are two permissions applied to orthogonal actions, it means that the entities (or related entities) are free to choose to execute or not these actions.

Fulfillment and Violation: The fulfillment of a permission does not imply on the fulfillment or violation of the other. However, if one orthogonal action is executed, the entity will not be able to execute the other action even though being permitted.

Conflict: There is no conflict between two norms that permit the execution of orthogonal actions.

3.4. Action Dependency

If a dependency relationship is defined between two actions, it means that there is an action that must be executed before another action. If the *client action* is not executed, the *dependent action* cannot be executed.

- **Obligation**: If there is an obligation applied to a *dependent action*, it means that the entity is obliged to execute such action.

Fulfillment and violation: in order to fulfill the obligation the entity must be able to execute all *client actions* before executing the *dependent action*. If the entity is unable to execute one of the *client/precondition* action (i.e., if the entity is prohibited to execute one of the clients), the entity will violate the obligation. If there is a norm obliging an entity *to apply sanctions*, the same entity must be able to first *find out the violations* since such action is the *client to apply sanctions*.

Conflicts: If there is an obligation governing the behavior of an entity and applied to a *dependent action*, there must not be any prohibition governing the behavior of the same (or related) entity and applied to any client action at the same period of time and in the same context.

- **Prohibition**: If there is a prohibition applied to a dependent action, it means that the entity is prohibited to execute such action.

Fulfillment and violation: the execution or not of the *clients* of a *dependent* action being prohibited does not imply on the fulfillment or violation of such prohibition. The prohibition can be fulfilled independently of the precondition actions that may have been

executed. For instance, even being prohibited *to apply sanctions*, the entity can *find out violations*. Such execution does not imply on the fulfillment or violation of the prohibition.

Conflicts: There is no conflict between a norm that prohibits the execution of a dependent action and any other norm applied to the precondition actions.

- **Permission**: If there is a permission applied to a *dependent action*, it means that the entity is permitted to execute such action.

Fulfillment and violation: in order to be able to uses its permission to execute the *dependent action*, the client actions must not be prohibited. The permission for executing the *dependent action* can only be used if the entity is able to execute its *client actions*.

Conflicts: If there is a permission governing the behavior of an entity and applied to a dependent action, there must not be any prohibition governing the behavior of the same (or related) entity and applied to any client action at the same period of time and in the same context.

4. Conflict Checker

In this section, we present the algorithm for the checking of direct and indirect normative conflicts that was implemented following the conflict cases described in the previous section. Our approach is based on the rewriting of the norms, what it similar to the unification approach used in [6]. If the scope of a norm includes the scope of another norm, the more general norm can be rewritten to comply with the more specific norm.

The main algorithm (algorithm 5) uses auxiliary functions that checks (i) if the contexts of two norms are equal or related (algorithm 1); (ii) if the entities whose behavior is governed by the norms are the same or are related (algorithm 2); (iii) if the periods during while the norms are active intersect (algorithm 3); and (iv) if the actions being governed by the norms are equal or related (algorithm 4).

As stated before, Algorithm 1 indicates if the contexts of the two norms are equal or related. In case both contexts are *organizations*, it checks if one is a *suborgnaization* of another. In case one context is an *organization* and another an *environment*, it checks if the *organization* inhabits the *environment*. In case both contexts are environments, it checks if one is a *subenvironment* of another.

Algorithm 2 starts by checking if the entities are the same. If not, it figures out if the entities are related by one of the following relationships, as described in [9]: *hierarchy* (that indicates the entities inhabiting the environment), *play* (that indicates the entities playing roles), *playin* (that represents the organizations where the entities are playing roles) and *ownership* (that states roles defined in organizations).

Algorithm 1 Function: Verifying Context Relationship

```
Require: n1 and n2 as parameter
function contextsRelationship(n1, n2)
if (n1.c = n2.c) then
   return true
else
   if (n1.c.cType = organization) and
      (n2.c.cType = organization) then
      if (n1.c ⊂ n2.c) or (n2.c ⊂ n1.c) then
         return true
      endif
   endif
   if (n1.c.cType = environment) and
      (n2.c.cType = environment) then
      if (n1.c ⊂ n2.c) or (n2.c ⊂ n1.c) then
         return true
      endif
   endif
   if (n1.c.cType = organization) and
      (n2.c.cType = environment) then
      if (n1.c ⊂ n2.c) then
         return true
      endif
  endif
endif
return false
endfunction
```

A

Algorithm 2 Function: Verifying Entities Relationship

```
Require: n1 and n2 as parameter
function entitiesRelationship(n1, n2)
if (n1.e = n2.e) then
   return true
else
   if (checkEntitiesRelationship(n1.e, n2.e) = hierarchy)
then
      return true
   endif
   if (checkEntitiesRelationship(n1.e, n2.e) = play) then
      return true
   endif
   if (checkEntitiesRelationship(n1.e, n2.e) = playin) then
      return true
   endif
   if (checkEntitiesRelationship(n1.e, n2.e) = ownership)
then
      return true
   endif
endif
return false
endfunction
```

B

Figure 1. (A) Verifying context relationship and (B) verifying entities relationship

The third algorithm is responsible to check if the periods during while the norms are active intersect. If a norm is deactivated after the activation of another, the activation periods intersect. Note that in case the actions are related by the dependency relationship it is not necessary to check if the activation periods intersect. We need (only) to guarantee that the client action can be executed before the dependent action.

Algorithm 3 Function: Verifying Time Intersect

```
Require: n1 and n2 as parameter
function timeIntersect(n1, n2)
   if (checkActionsRelationship(n1.a, n2.a) = dependency) then
      return true
   else
      if (((n1.ac <= n2.ac) and (n1.dc >= n2.ac)) or
         ((n1.ac >= n2.ac) and (n1.ac <= n2.dc)) then
         return true
      endif
   endif
   return false
endfunction
```

Figure 3. Verifying time intersect.

Algorithm 4 checks if the actions identified in the two norms are equal or are related. If the actions are not equal, the algorithm checks if they are related by one of the relationships described in Section 3, as follows:

- *Refinement relationship*: (i) If the execution of the *superaction* is being prohibited and the *subaction* is being obliged, the algorithm concludes that there is a potential conflict. (ii) If the execution of the *superaction* is being obliged, it is necessary to check if there is at least one *subaction* that can be executed, i.e., if there is not a norm prohibiting the execution of such action. If all *subaction* cannot be executed, it means that there is a potential conflict. (iii) Similar to an obligation applied to the *superaction*, if the execution of the *superaction* is being permitted, it is necessary to check if there is at least one *subaction* that can be executed. If not, there is a potential conflict.

```
Algorithm 4 Function: Verifying Actions Relationship
Require: n1 and n2 as parameter
function actionsRelationship(n1, n2)
  if (n1.a = n2.a) then
    return true
  else
    if (checkActRelationship(n1.a, n2.a) = refinement) and
      ((n1.deoC = O and checkAllSubActions(n1.a, F)) or
       (n1.deoC = F and n2.deoC = (O or P)) or
       (n1.deoC = P and checkAllSubActions(n1.a, F)) ) then
          return true
    endif
    if (checkActRelationship(n2.a, n1.a) = refinement) and
      ((n2.deoC = O and checkAllSubActions(n2.a, F)) or
       (n2.deoC = F and n1.deoC = (O or P)) or
       (n2.deoC = P and checkAllSubActions(n2.a, F)) ) then
          return true
    endif
    if (checkActRelationship(n1.a, n2.a) = composition) and
      ((n1.deoC = O and checkAnyPartActions(n1.a, F)) or
       (n1.deoC = F and checkAllPartActions(n1.a, O)) or
       (n1.deoC = P and checkAnyPartActions(n1.a, F)) ) then
          return true
    endif
    if (checkActRelationship(n2.a, n1.a) = composition) and
      ((n2.deoC = O and checkAnyPartActions(n2.a, F)) or
       (n2.deoC = F and checkAllPartActions(n2.a, O)) or
       (n2.deoC = P and checkAnyPartActions(n2.a, F)) ) then
          return true
    endif
    if (checkActRelationship(n1.a, n2.a) = orthogonal) and
      ((n1.deoC = O and n2.deoC = O) or
       (n1.deoC = O and n2.deoC = P) or
       (n2.deoC = O and n1.deoC = P) ) then
          return true
    endif
    if (checkActRelationship(n1.a, n2.a) = dependency) and
      ((n1.deoC = O and checkAnyClientActions(n1.a, F)) or
       (n1.deoC = P and checkAnyClientActions(n1.a, F)) ) then
          return true
    endif
    if (checkActRelationship(n2.a, n1.a) = dependency) and
      ((n2.deoC = O and checkAnyClientActions(n2.a, F)) or
       (n2.deoC = P and checkAnyClientActions(n2.a, F)) ) then
          return true
    endif
  endif
  return false
endfunction
```

Figure 4. Verifying actions relationship.

- *Composite relationship*: (i) if the execution of the *whole action* is being obliged and the *part action* is being prohibited, the algorithm concludes that there is a potential conflict. (ii) If the *whole action* is being prohibited, and the *part action* is being obliged, it is necessary to check if the other part actions are also being obliged or not. If all *part actions* are being obliged, it means that if the agent fulfills its obligations, it will violate the prohibition. Therefore, if all *part actions* are being obliged, the algorithm concludes that there is a potential conflict. (iii) If the execution of the *whole action* is being permitted and a *part action* is being prohibited, the algorithm concludes that there is a potential conflict. If the agent follows its permission it will need to execute the *part action* that is being prohibited.
- *Orthogonal relationship*: if both norms are obliging the execution of orthogonal actions, the algorithm concludes that there is a potential conflict. If one norm obliges and the other permits, it is also a case of potential conflict.
- *Dependency relationship*: if the *client action* is being prohibited and the *dependent action* is being obliged or permitted, the algorithm concludes that there is a potential conflict.

At last but not least, algorithm 5 is responsible to coordinate all other algorithms. It calls the others in sequence and informs if the norms are in conflict or not. In order to exemplify the algorithm, let's consider the following norms described according to the definition in Section 2.

Norm 1: In a workshop, the attendees are obliged to make silence during the presentation of speakers.

```
N1:{obligation, workshop, attendee, makeSilence, talkStarted,
talkFinished,_}
```

Norm 2: In a given section of a workshop, the chair is obliged to tell the speaker that (s)he has 5 minutes to finish the talk.

N2:*{obligation, section, chair,tell(5minutes), talkStarted, talkFinished,_}*

Algorithm 5 Conflict Checker Main

Require: The *norm* base (represented by *norms*)
Require: The n_1 and n_2 from the *norm* base
if *contextsRelationship*(n_1, n_2) **then**
  if *entitiesRelationship*(n_1, n_2) **then**
    if *timeIntersect*(n_1, n_2) **then**
      if *actionsRelationship*(n_1, n_2) **then**
        return *Norms are in conflict!*
      endif
    endif
  endif
endif
return *Norms are not in conflict!*

Figure 5. Conflict checker main.

Let's assume that the application domain states that workshops are composed of sections and that the section chair is an (kind of) attendee. In addition, it describes that to make silence is the opposite of to tell.

(workshop, section, hierarchy)
(attendee, chair, hierarchy)
(makeSilence, tell, orthogonal)

Based on the domain description above and executing algorithm 5, we can conclude that N1 is in conflict with N2. In the first step of algorithm 5, the contexts of the norms are checked and algorithm 1 concludes that they are related. Since norm1 is applied in the context of workshops and workshops are composed of sections, N1 can be rewritten to:

N1a:*{obligation, section, attendee, makeSilence, talkStarted, talkFinished,_}*

The second step of algorithm 5 calls algorithm 2 to check if the entities indicated in the norms are related. Algorithm 2 concludes that they are related by the hierarchy relationship and N1a is written to:

N1b:*{obligation, section,chair, makeSilence, talkStarted,talkFinished,_}*

The third step of algorithm 5 concludes that the time during with the norms are active is exactly the same. The fourth step calls algorithm 4 that checks the relationship between the actions and the deontic concepts of the norms are analyzed. Since the actions being governed by the norms are orthogonal and the deontic concepts are obligation, the algorithm concludes that N1b and N2 are in conflict.

5. Related Work

There are several works on the checking of normative conflicts and on the resolution of those conflicts. However, the majority focuses on the checking of simple conflicts. In [Kollingbaum et al. 2008], [Vasconcelos et al. 2007], [Vasconcelos et al. 2009] and [Vasconcelos and Norman 2009] the authors presents approaches for the checking of normative conflicts/inconsistencies that do only consider norms applied to the same action. Indirect conflicts are thus not detected.

Indirect conflicts are detected in works such as [Dung and Sartor 2011], [Gaertner et al. 2007], [Garcia-Camino et al. 2006], [Kollingbaum et al. 2008a] and [Vasconcelos et al. 2007]. The approaches in [Gaertner et al. 2007] and [Garcia-Camino et al. 2006]

take into account the normative position when checking for conflicts. Normative position describes activities that are propagated to other activities. The approach presented in [Gaertner et al. 2007] considers that multiple, concurrent and related activities are executed by agents and present a conflict checker that takes that into account. In [Garcia-Camino et al. 2006] the authors consider the composition relationship between activities, i.e., an activity can be composed of several sub-activities.

In [Kollingbaum et al. 2008b] and [Vasconcelos et al. 2007] the normative conflict checker considers indirect conflicts by taking into account the domain specific relationships among actions. Two relationships among actions are defined (composition and delegation) and they also use unification to find out the norms that overlap. We claim that such approaches are incomplete since they consider only the relationships among actions and ignore the relationships among the entities.

The work presented in [Dung and Sartor 2011] focuses on conflicts between norms defined in different contexts. Similar to such approach, the works presented in [Li et al. 2013a] and [Li et al. 2013b] are able to detect conflict among different laws defined in different jurisdictions.

In [Silva 2013] the author presented an algorithm to detect conflicts between two norms. The algorithm focuses on detecting conflicts between a prohibition and an obligation that do not govern the behavior of the same entity, but entities that are somehow related. In addition, this first version of the conflict checker algorithm can also identify conflicts between a prohibition and an obligation that are applied to different actions related by the refinement and composition relationships. In this paper, the algorithm was extended to be able to detect conflicts between prohibition and permission and to consider two new kinds of relationships that link actions: orthogonal and dependency.

6. Conclusion and Future Work

This work presents the identification of normative conflicts that can only be found when considering the system characteristics. In [Silva 2013] the author detailed the possible relationships that can be used to relate the entities of the systems. In this paper, we focus on presenting some relationships that can be used to relate the actions executed by those entities. To consider those relationships is fundamental on the identification of the interdependencies between the norms and on the checking of conflicts.

One important limitation of the current work is that it is limited to analyze the relationships among actions. A norm can be used to regulate the execution of an action but also the achievement of a state. Therefore, we are now working on describing relationships between states and defining the connections between states and actions in order to be able to check for conflicts among norms that regulate the execution of an action and the achievement of a state.

In addition, we are in the process of implementing the algorithm by using Jess, a rule engine for Java platform (http://herzberg.ca.sandia.gov/). We are developing an expert system able to detect conflicts between two norms by considering the system characteristics, to indicate to the user why the norms are in conflict, to provide possible conflict resolutions, and to apply one of the possibilities, it the user wishes to do so.

References

Dung, P., Sartor, G. (2011) “The modular logic of private international law”, In: Artificial Intelligence and Law. Springer, 19(2-3), pp. 233-261.

Figueiredo, K., Silva, S., Braga, C. (2011) “Modeling Norms in Multi-agent Systems with Norm-ML”, In: International Workshop on Coordination, Organizations, Institutions and Norms VI. LNAI 6541, Springer, pp. 39-57.

Gaertner, D., Garcia-Camino, A., Noriega, P., Vasconcelos, W. (2007) “Distributed Norm Management in Regulated Multi-agent Systems”, In: International Conference on Autonomous Agents and Multiagent Systems. ACM, pp. 624-631.

Garcia-Camino, A., Noriega, P., Rodrigues-Aguilar, J. (2006) “An Algorithm for Conflict Resolution in Regulated Compound Activities”, In: Engineering Societies in the Agents World VII, LNCS 4457, Springer, pp. 193-208.

Kollingbaum, M., Vasconcelos, W., Garcia-Camino, A., Norman, T. (2008a) “Managing Conflict Resolution in Norm-Regulated Environments”, In: Engineering Societies in the Agents World VIII, LNCS 4995, Springer, pp. 55-71.

Kollingbaum, M., Vasconcelos, W., Garcia-Camino, A., Norman, T. (2008b) “Conflict Resolution in Norm regulated Environments via Unification and Constraints”, In: Declarative Agent Languages and Technologies V, LNCS 4897,Springer,pp.158-174.

Li, T., Balke, T., De Vos, M., Padget, J., Satoh, K. (2013a) “Legal Conflict Detection in Interacting Legal Systems”, In: The 26th International Conference on Legal Knowledge and Information Systems (JURIX).

Li, T., Balke, T., De Vos, M., Satoh, K., Padget, J. (2013b) “Detecting Conflicts in Legal Systems”, In: New Frontiers in Artificial Intelligence. LNCS 7856, Springer, pp. 174-189.

Silva, V., (2013) “Normative Conflicts that Depend on the Application Domain”, In: Int. Workshop on Coordination, Organizations, Institutions and Norms, p.p. 119-130.

Vasconcelos, W., Kollingbaum, M., Norman, T. (2007) “Resolving conflict and inconsistency in norm regulated virtual organizations”, In: International Conference on Autonomous Agents and Multiagent Systems. ACM, pp. 632-639.

Vasconcelos, W., Kollingbaum, M., Norman, T. (2009) “Normative conflict resolution in multi-agent systems”, In: Journal of Autonomous Agents and Multi-Agent Systems. ACM, 19(2), pp. 124-152.

Vasconcelos, W., Norman, T. (2009) “Contract Formation through Preemptive Normative Conflict Resolution”, In: International Conference of the Catalan Association for Artificial Intelligence. ACM, pp. 179-188.

Uma Abordagem Baseada em Agentes para Planejamento e Monitoramento de Serviços de Saúde

Nécio de Lima Veras[1], **Mariela I. Cortés**[2], **Gustavo A. Lima de Campos** [2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Rodovia CE-075, s/n - Aeroporto – Tianguá - Ceará

[2]Universidade Estadual do Ceará (UECE) - Fortaleza, CE - Brasil

necio.veras@ifce.edu.br, {mariela, gustavo}@larces.uece.br

Abstract. *The continuous improvement of the offered services is a challenge, specially when the resources are scarce. The National Program for Improving Access and Quality of Primary Care (PMAQ), foments the continuous growth of the access and quality of health services based on self-evaluation, planing and monitoring of the health teams work. This paper presents an intelligent computer architecture to give support to the health services planning oriented by goals, assisting in the evaluation of the teams' performance and their progressive growth considering resources and local demands. The solution specifies agents able to plan and monitor the services to increase teams' performance according the PMAQ indicadors.*

Resumo. *A melhora contínua da prestação de serviços é um desafio, especialmente com escassez de recursos. O Programa Nacional de Melhoria do Acesso e da Qualidade da Atenção Básica (PMAQ) promove o contínuo crescimento do acesso e qualidade dos serviços com base na autoavaliação, planejamento e monitoramento do trabalho das equipes de saúde. Este artigo apresenta uma arquitetura computacional inteligente para dar suporte ao planejamento de serviços orientado por metas, auxiliando na avaliação do desempenho das equipes e da sua melhoria progressiva considerando recursos e demandas locais. A solução especifica agentes capazes de planejar e monitorar os serviços para melhorar o desempenho das equipes conforme os indicadores do PMAQ.*

1. Introdução

O Programa Nacional de Melhoria do Acesso e da Qualidade da Atenção Básica (PMAQ-AB), pressupõe 47 indicadores agrupados em 7 áreas. Seu objetivo é criar um ciclo contínuo de crescimento do acesso às ações e serviços de saúde pelos usuários, para alcançar um padrão de qualidade comparável nacional, regional e localmente [BRASIL 2012]. A proposta do PMAQ é criar nas equipes uma cultura de aperfeiçoamento contínuo implementada por meio de autoavaliação, monitoramento, educação permanente e apoio institucional. No entanto, para lograrem êxito neste processo, as equipes da atenção básica (EAB) enfrentam grandes desafios. Entre eles, destaca-se a necessidade de qualificação nos processos de trabalho das equipes, de forma a orientar o trabalho em função de prioridades, metas e resultados, definidos em comum acordo pela equipe, gestão municipal e comunidade [Ladeira 2011].

O Ministério da Saúde (MS), considerando o conjunto dos desafios enfrentados pelas EAB, estruturou o Programa usando sete diretrizes básicas [BRASIL 2012]. Idealizou-se dentre outras coisas, a utilização de parâmetros de comparação entre as equipes considerando diferentes realidades; o melhoramento incremental e contínuo de indicadores de acesso e qualidade que qualificam a organização e trabalho da Equipe; o desenvolvimento cultural de negociação e contratualização em função de compromissos e resultados pactuados. Percebe-se que estão envolvidos no processo o planejamento orientado pela busca de resultados e o monitoramento e acompanhamento de indicadores de saúde. Tendo em vista que a unidade de avaliação do PMAQ é a equipe de saúde, o MS estabeleceu os 47 indicadores como um meio de mensurar o desempenho das equipes em relação ao acesso e a qualidade dos serviços oferecidos por elas. Em [BRASIL 2012] é possível visualizar as fórmulas para calcular cada um dos indicadores com suas respectivas variáveis.

A informática em saúde relaciona-se às atividades de cuidado à saúde e pode ser um importante aliado estratégico para o avanço da Saúde Coletiva. Entretanto, apesar da grande difusão de tecnologias da informação e comunicação na área da Saúde Pública, em um nível municipal ainda existe uma demanda crescente da gestão dos serviços e da produção das informações em saúde [Silva et al. 2012] [Barbosa and Forster 2010]. Existem poucos projetos visando tornar o PMAQ aplicável no dia a dia de trabalho dos profissionais da área de saúde, portanto, o desenvolvimento de ferramentas que possibilitem automatizar e otimizar o planejamento de atividades poderia propiciar um aumento da produtividade e da satisfação com o trabalho e, consequentemente, melhorar o serviço de atendimento aos usuários.

O presente artigo propõe uma arquitetura abstrata de um sistema inteligente capaz de atuar no planejamento orientado por metas e monitoramento dos serviços prestados pelas EAB através de uma agenda eletrônica, conforme os recursos disponíveis e a demanda requerida. A solução envolve a definição de dois agentes, de planejamento e monitoramento respectivamente.

2. Trabalhos relacionados

A ferramenta computacional apresentada em [Silva et al. 2012] tem como função principal calcular indicadores de qualidade com base no PMAQ com objetivo de apresentar relatórios de apoio para que a equipe de saúde possa monitorar os serviços prestados. O trabalho mostrou-se capaz de auxiliar as equipes de saúde na implantação e manutenção do PMAQ, especialmente, no que se refere ao monitoramento dos indicadores estabelecidos pelo programa, porém, não usou de métodos inteligentes para propor um planejamento ou estratégia para corrigir ou, ainda, prever possíveis deficiencias da equipe em relação à previsão dos indicadores.

No trabalho de [Barbosa and Forster 2010], os autores apresentam um estudo sobre a percepção dos profissionais da atenção báscia (AB) em relação ao uso do Sistema de Informação em Saúde (SIS) como ferramenta facilitadora de suas ações, principalmente, no âmbito do planejamento. Nas conclusões, percebe-se que o planejamento automatizado de ações na saúde é algo que parece distante da realidade de muitas unidades básicas de saúde (UBS) da AB no modelo de saúde brasileiro, tendo em vista que as ferramentas disponíveis não estão especializadas para este fim e, mesmo que estivessem, ficou

evidenciado uma subutilização do sistema por parte da Equipe alvo da pesquisa.

Várias pesquisas sobre Agentes de Informação Médica (MIA) concentram-se na implantação de sistemas multiagentes para resolver problemas de planejamento na área de saúde. Em [Braun et al. 2005] o principal objetivo foi obter um planejamento multiagente eficiente para ambientes dinâmicos de planejamentos em saúde. Os detalhamentos dos subprojetos englobados no trabalho mostram boas contribuições para arquiteturas de sistemas inteligentes aplicados à área da saúde. No entanto, os mesmos não são estendidos ao planejamento de atendimentos na atenção primária.

Em [Vermeulen et al. 2009] foi apresentada uma abordagem adaptativa para otimização automática de agendamento de recursos. O agendamento eficiente de consultas de pacientes em recursos dispendiosos é uma tarefa complexa e dinâmica. O trabalho trata do uso da Inteligência Artificial na resolução de problemas da área médica, especificamente, otimização dinâmica para o planejamento de atendimentos e alocação de recursos em um Hospital Médico Acadêmico. A modelagem matemática do problema e da simulação, assim como os resultados gerados pelas simulações, apresentam soluções satisfatórias para os problemas elencados e podem ser usados como modelos para a criação/adaptação de uma nova abordagem ou simulação para o caso do planejamento de atendimentos em saúde na atenção primária, porém levando em conta os indicadores propostos no PMAQ.

3. Um sistema multiagente para o planejamento e monitoramento de serviços de saúde

Este artigo apresenta uma abordagem computacional com propriedades inteligentes materializada para a solução de dois problemas, definidos informalmente da seguinte forma: (a) alocação de serviços de saúde em uma agenda dinâmica otimizada conforme demandas locais e orientada por metas e (b) monitoramento dos serviços ofertados objetivando calcular os valores dos indicadores definidos pelo PMAQ a partir da agenda otimizada.

A solução proposta abrangendo ambos os problemas incorpora heurísticas para o problema de alocação e otimização do agendamento, e a tecnologia de agentes inteligentes para o monitoramento do desempenho, cuja adequação já vem sendo testada para o domínio da saúde [Braun et al. 2005]. Para isso, idealizou-se uma arquitetura de sistema inteligente englobando os aspectos fundamentais da proposta. A Figura 1 ilustra abstratamente a arquitetura idealizada para a materialização da abordagem.

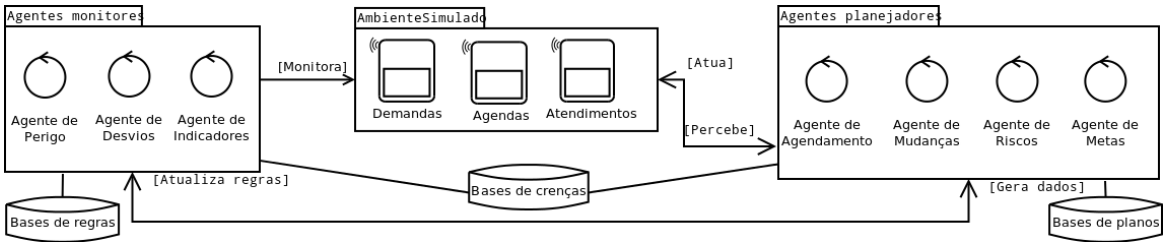


Figura 1. Arquitetura do sistema inteligente

O *AmbienteSimulado* é um *workspace* que contém artefatos de ambiente da tecnologia Cartago [Ricci et al. 2009], e é responsável por simular cenários específicos para

demandas dos serviços ofertados, emitindo sinais para os agentes que o monitoram. Os *Agentes Monitores* percebem as alterações ambientais no *AmbienteSimulado* e executam ações segundo as suas respectivas bases de regras. Estes, comumente, possuem uma estrutura reflexiva (descrita em [Russell and Norvig 2003]), onde seu comportamento é regido por regras condição-ação. Vale frisar que as bases de regras sofrem atualizações do outro grupo de agentes, os *Agentes Planejadores*.

O grupo monitor é formado por três agentes, são eles: Agente de Perigo, Agente de Desvios e Agente de Indicadores. O presente artigo detalha na Seção 5.2 o Agente de Indicadores. Esse agente tem como missão calcular indicadores com fórmulas pré-estabelecidas contidas em sua base de regras. No grupo dos *Agentes Planejadores* existem agentes que realizam tarefas de planejamento com bases (ou bibliotecas) de planos. O grupo é formado por Agente de Agendamento, Agente de Mudanças, Agente de Riscos e Agente de Metas. O trabalho atual especifica o agente responsável por planejar uma agenda dinâmica de serviços para cada EAB, o Agente de Agendamento. A agenda é orientada por metas e visa otimizar o cumprimento para as demandas existentes. Este agente incorpora a formulação da agenda dinâmica (formalmente descrito na Seção 4) como fundamento para executar seus planos por meio de heurísticas de otimização.

4. Formalização da agenda dinâmica para o Agente de Agendamento

Partindo da arquitetura apresentada na Seção 3, esta seção objetiva formalizar matematicamente uma agenda para ofertar serviços de saúde. A alocação de um serviço qualquer (S_x) pertencente ao conjunto dos serviços ofertados (S) em uma agenda (G) caracteriza um atendimento apenas quando o serviço for efetivamente executado em um ambiente de atendimento. Neste trabalho, para o planejamento de uma agenda semanal associada a um profissional G_s^P, define-se que esta é representada como uma matriz composta por c colunas e l linhas que indicam a alocação de serviços S_{lc} que pertencem à S na agenda, onde c representa os dias da agenda e l a unidade de tempo de execução do serviço, formalmente definida como:

$$G_s^P = (S_{lc})_{l \times c} = \begin{pmatrix} S_{11} & S_{12} & \dots & S_{1c} \\ S_{21} & S_{22} & \dots & S_{2c} \\ \vdots & \vdots & \vdots & \vdots \\ S_{l1} & S_{l2} & \dots & S_{lc} \end{pmatrix}, \; tal\ que\ c \leq 7 \;\; e \;\; l \leq 8 \tag{1}$$

Uma agenda mensal de um profissional G_m^P pode ser visualizada como um conjunto de até cinco agendas semanais, dado por:

$$G_m^P = \{G_{s_1}^P, G_{s_2}^P, G_{s_3}^P, G_{s_4}^P, G_{s_5}^P\} \tag{2}$$

Para o modelo proposto, segundo [BRASIL 2012], observou-se a relação entre os serviços da agenda e os indicadores. Viu-se que (1) certos indicadores são influenciados pela realização dos serviços de uma agenda e podem ser calculados a partir dos serviços alocados na agenda; (2) outros indicadores não sofrem nenhuma alteração em seus valores por conta dos serviços de uma agenda; e (3) existem indicadores que influenciam no planejamento da alocação de serviços de uma agenda e portanto são determinantes para a valoração das quantidades dos serviços a serem alocados em uma agenda.

Como resultado, a partir do universo de indicadores do Programa foi extraído um subconjunto de indicadores que estão ligados com o planejamento a partir de demandas e capacidades locais para atendimentos nas UBS. A Tabela 1 relaciona os serviços identificados com os indicadores. Os quinze indicadores relacionados pela Tabela 1 exercem

Tabela 1. Mapeamento entre serviços ofertados e indicadores do PMAQ/2012

Serviço Ofertado	Indicador(es)
Atendimento de Pré-natal	1.2
Prevenção do câncer ginecológico	1.6
Visitas domiciliares	Pessoas acompanhadas no domicílio
Consulta de puericultura	2.1
Consulta médica	2.1, 2.5,2.6, 5.1
Consulta de Hipertensos e Diabéticos	3.3, 3.4
Ação coletiva de escovação dental supervisionada	4.1
Consulta Odontológica	4.2
Consulta Odontológica à Gestante	4.3
Consulta de enfermagem	5.10
Vigilância (tuberculose e hanseníase)	6.1, 6.2

influência direta no processo de planejamento da alocação dos serviços prestados para um conjunto de agendas. O processo de planejamento de uma agenda individual envolve o cálculo de demandas para os serviços ofertados e a determinação de metas almejadas pela EAB. A formulação contempla que determinados serviços podem ser executados por um ou mais profissionais e que a execução de um serviço envolve um custo de tempo. Para este trabalho, os valores para os custos de tempo (em minutos) foram estabelecidos conforme [Brasil 2002], no entanto, a abordagem permite alterar os valores de acordo com possíveis especificidades locais.

Sobre o planejamento de metas para a oferta dos serviços alocados na agenda, idealmente estas devem ser contratualizadas pela Estratégia de Saúde da Família (ESF) municipal com as equipes aderentes ao PMAQ de forma prévia à adesão ao Programa. Como a pactuação de metas não possui caráter obrigatório para as equipes aderentes ao Programa, então deve ser realizada nas situações em que o gestor municipal entender a necessidade e/ou importância de definir previamente as metas para parte ou para a totalidade dos indicadores. Isso torna a proposta flexível e capaz de permitir um planejamento realista da UBS em relação à sua pactuação.

O modelo proposto considera que as metas serão individualizadas conforme o serviço ofertado. A ideia é definir metas locais, calcular as demandas para estimar o quantitativo dos serviços ofertados para um profissional (D_P^S) e alocá-los em uma agenda para posterior cálculo de qualidade da agenda da equipe. O volume (em horas) de trabalho de um profissional em relação a um serviço $S_x \in S$ ($V_P^{S_x}$) no contexto de G_m^P indica quanto o serviço S_x foi alocado na agenda. Dado por:

$$V_P^{S_x} = count(G_m^P, S_x) \tag{3}$$

Vale destacar que $count$ é uma função binária capaz de contabilizar quantas horas um determinado serviço $S_x \in S$ foi alocado em G_m^P. Esta função binária pode ser definida da seguinte forma:

$$count : G_m^P \times S_x \rightarrow |S_x|, \quad tal\ que : \quad S_x \in G_m^P \tag{4}$$

Assim, a qualidade de G_m^P é dada por:

$$Q_{G_m^P} = \sum_{i=1}^{Servs} \frac{V_P^{S_i}}{D_P^{S_i}} - P1 - P2 - P3, onde: \quad (5)$$

$$P1 = \left(\sum_{j=1}^{DServs^+} \frac{V_P^{S_j}}{D_P^{S_j}} \right) \times 2, \; P2 = \left(\sum_{l=1}^{DServs^-} \frac{D_P^{S_l}}{V_P^{S_l}} \right) \times 6 \text{ e } P3 = \left(\sum_{q=1}^{DServs^n} \frac{D_P^{S_q}}{100} \right) \times 10.$$

É importante estabelecer que $\forall S_i, S_j, S_l \in G_m^P$ e $\forall S_q \notin G_m^P$ temos que:
$Servs$ é a contagem dos serviços não repetidos de G_m^P,
$DServs^+$ é a contagem dos serviços alocados em G_m^P que estão acima de D,
$DServs^-$ é a contagem dos serviços alocados em G_m^P que estão abaixo de D e
$DServs^n$ é a contagem dos serviços NÃO alocados em G_m^P, mas que pertencem ao conjunto S associado a um profissional P.

Considerando que uma equipe poderá possuir diversas agendas diferentes então a qualidade de uma agenda mensal para esta equipe é calculada por:

$$Q_{G_m^E} = \frac{\sum_{x=1}^{o} Q_{G_m^{P_x}}}{|E|} \quad (6)$$

Dessa forma, a função de qualidade (Equação 5) avalia o grau de cumprimento dos respectivos serviços em relação às suas demandas. A primeira soma agrupa esse grau para todos os serviços alocados na agenda considerando a relação entre volume de alocação e demanda para os serviços. As somas $P1$, $P2$ e $P3$ funcionam como penalidades aos serviços desbalanceados em relação as suas demandas. O serviço cujo volume está acima de sua demanda ($P1$) é penalizado com peso dois, enquanto que o serviço abaixo da demanda ($P2$) sofre uma punição mais rígida com peso seis. O último somatório ($P3$) aplica uma penalidade ao serviço que deveria ter sido alocado na agenda mas não foi e, para isso é aplicado um peso dez.

Essas definições matemáticas estão contidas em heurísticas embutidas no Agente de Agendamento e as estruturas computacionais criadas para manipular o formalismo apresentado são utilizadas por todo o sistema inteligente proposto. A seção seguinte apresenta e especifica os dois agentes para planejamento e monitoramento, respectivamente.

5. Especificação dos agentes de Agendamento e de Indicadores

5.1. O Agente de Agendamento

O Agente de Agendamento é um agente que objetiva planejar atendimentos de saúde orientado por metas. O agente foi projetado usando a tecnologia de agentes Jason [Bordini and Hübner 2005] em uma arquitetura deliberativa baseada em estados mentais fundamentados em *crenças, desejos e intenções*, descrita em [Wooldridge 2008] e conhecida também como arquitetura BDI (*belief*, *desire* e *intention*). Sua concepção foi inspirada na proposta filosófica do BDI, pois o Agente crê que uma agenda dinâmica pode ser capaz de atender demandas de serviços ofertados se for otimizada para cumprir com metas preestabelecidas e, para isso, ele deseja planejar inteligentemente a alocação dos serviços

ofertados pela UBS. Desta forma, o Agente se compromete em realizar a otimização. Assim, sua intenção é planejar de maneira inteligente a cobertura para uma oferta dos serviços conforme as metas desejadas, materializando-as sob a forma dos serviços alocados em agendas a serem executadas pelas equipes de saúde. Com isso, o estado final do agente consiste de um grupo de agendas otimizadas. A Figura 2 ilustra os componentes internos do agente.

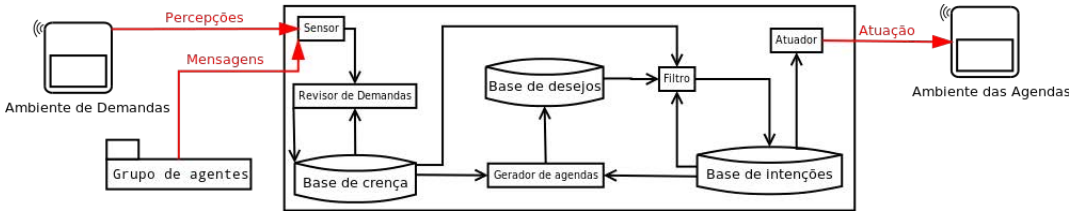


Figura 2. Estruturas internas do Agente de Agendamento

O agente possui sensores capazes de perceber mudanças no Ambiente de Demandas e de receber mensagens vindas de outros agentes com uma solicitação para otimizar agendas. A função Revisor de Demandas gerencia as crenças do agente calculando as demandas com base nas metas percebidas. O desejo de otimizar as agendas é realizado por meio da função Gerador de Agendas que utiliza ações internas específicas de Inteligência Computacional responsáveis por encontrar a máxima qualidade de uma agenda. Em seguida, a função Filtro faz com que o agente delibere em relação à agenda mais apropriada encontrada por meio do uso das heurísticas *Hill Climbing* e *Annealing Simulated* [Russell and Norvig 2003]. Por fim, o agente atua no Ambiente das Agendas inserindo o seu estado atual. O Algoritmo 1 fornece uma visão geral do comportamento do Agente sob a forma da linguagem lógica *AgentSpeak* [Bordini and Hübner 2005].

Algoritmo 1 agenteDeAgendamento

```
+agendasDaEquipe(null).
!encontraEMonitoraArtefatos.
+percebeNovasDemandas(Demandas).
+recebeSolicitacoesPorMensagens.
!revisorDeDemandas(Demandas).
!geradorDeAgendas (MelhoresAgendas).
-+agendasDaEquipe(MelhoresAgendas).
!apresentaRelatorio.
!registraAgendas.
```

O Algoritmo 2 especifica em pseudocódigo o processo executado pelo Agente por meio do plano *!geradorDeAgendas*. Este plano é responsável por gerar inúmeras agendas a fim de buscar a maximização do valor da função de qualidade das agendas da Equipe em relação ao cumprimento das metas planejadas, conforme formalizado na Equação 6.

5.2. O Agente de Indicadores

O Agente de Indicadores é um agente de monitoramento que percebe as agendas planejadas pelo Agente de Agendamento e calcula os indicadores PMAQ relacionados com o planejamento da oferta dos serviços prestados pela UBS. A concepção do agente foi

Algoritmo 2 Processo de Otimização por *HillClimbing* e *Annealing Simulated*

```
Inicializa Equipe E com N Profissionais
Atribui demandas D para Equipe E
for each Profissional P ∈ E.profissionais do
  criaAgenda(A)
  alocaServicosAleatoriamente(A)
  adicionaAgendaNaEquipe(A,E)
end for
Equipe HC ← geraOtimizacaoPorHillClibing(E)
Equipe AS ← geraOtimizacaoPorAnnealingSimulated(E)
retorne filtraMelhorAgenda (HC, AS)
```

inspirada em uma arquitetura reativa simples com estado interno [Wooldridge 2008] e a Figura 3 exibe as estruturas internas do agente.

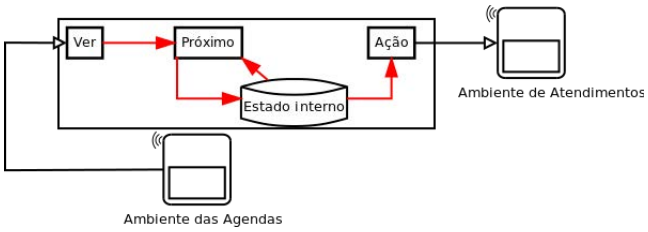


Figura 3. Estruturas internas do Agente de Indicadores

Neste trabalho, o agente possui como função Ação o registro os valores calculados no Ambiente de Atendimentos. Próximo significa o comportamento Agente para calcular os quinze indicadores selecionados pela abordagem. O Estado Interno representa o armazenamento dos valores calculados para cada um dos indicadores a partir das agendas percebidas pela função Ver. O Algoritmo 3 descreve informalmente o comportamento lógico do Agente usando comandos semelhantes aos utilizados pela linguagem *AgentSpeak*.

Algoritmo 3 agenteDeIndicadores

```
agendasDaEquipe(null).
!encontraEMonitoraArtefatos.
+percebeNovasAgendas(Agendas).
for each Agenda A ∈ Agendas do
  !calculaIndicadores(A).
end for
!registraIndicadores.
!apresentaIndicadores.
```

Partindo das definições apresentadas na Seção 4, este artigo buscou a experimentação de funcionamento dos agentes supracitados. Usando uma abordagem mono-objetiva desejou-se maximizar a qualidade de uma agenda mensal de uma equipe de saúde formada por grupos de três diferentes profissionais (enfermeiros, médicos e odontólogos).

6. Execução e simulação da abordagem

Esta seção objetiva simular a execução da abordagem considerando ambos os agentes apresentados na Seção 5, assim como, avaliar a formulação do problema orientado por metas, apresentado na Seção 4. Para tanto foi gerado um modelo de instância com dados empíricos, mas em conformidade com [Brasil 2002]. Por meio da utilização dos algoritmos de otimização citados, gerou-se agendas para coletar dados e verificar o cumprimento das metas em relação às demandas instanciadas. O intuito foi maximizar a qualidade das agendas da Equipe para demonstrar que a alocação inteligente de atendimentos possibilita um planejamento que priorize metas, assim como o PMAQ sugere que seja.

Para executar uma simulação com a abordagem proposta foi criado o Agente de Simulação. Este Agente gerencia a simulação criando os ambientes (artefatos Cartago), operando sobre as demandas e comunicando-se com o Agente de Agendamento. Sua missão é preparar os dados representativos para um município e suas Equipes de Saúde, instanciando-os a partir de um modelo textual para então atribuir os valores ao artefato Demandas. O modelo de instância foi concebido a partir das variáveis correlacionadas com os indicadores do PMAQ selecionados na abordagem. Também estão incorporadas ao modelo criado as metas associadas aos serviços ofertados, como podem ser visualizadas na Tabela 3.

6.1. Metodologia para a simulação

Em busca de confiabilidade para os resultados, idealizou-se uma instância empírica, porém com configurações semelhantes à de uma UBS do município de Viçosa do Ceará-CE. Para os testes foram produzidas três instâncias com quantidades diferentes para o número de profissionais de uma única equipe, no entanto, com os mesmos valores nas variáveis. As três instâncias estão sumarizadas conforme a Tabela 2 e a valoração das variáveis comuns para elas podem ser percebidas na Tabela 3.

Tabela 2. Sumário das instâncias de testes

Instância	Nº de Enfermeiros	Nº de Médicos	Nº de odontólogos
(1)	1	1	1
(2)	2	1	2
(3)	2	2	3

6.2. Resultados das execuções

Para cada uma das três instâncias foram gerados dados sobre o comportamento dos agentes propostos. A Figura 4 mostra os resultados para a execução considerando a instância 1. Nota-se que com essa configuração algumas metas foram atingidas, porém, demandas mais altas como a do serviço "Consulta_Enfermagem" não puderam ser atingidas pela insuficiência de profissional de enfermagem diante do volume desejado. O mesmo aconteceu com os serviços de odontologia e, por isso, na instância 2 foi acrescentado uma unidade ao número dos profissionais de enfermagem e de odontologia.

Percebe-se pela Figura 5 que foi possível atingir a maioria das metas. É viável concluir que o número de profissionais de enfermagem está adequado, porém, o número de médicos está aquém do mínimo e o número de odontólogos também não foi suficiente para atingir todas as demandas aplicáveis à área de odontologia. A instância 3 visa cobrir

Tabela 3. Valoração das variáveis da Equipe

Tipo	Descritor da variável PMAQ ou meta	Valor
PMAQ	*numeroGestantesCadastradasEquipe*	40
	populacaoFemininaCadastradaCom15AnosOuMais	321
	numeroMenoresDeDoisAnos	15
	numeroDeMenoresDeUmAnoAcompanhadas	09
	numeroDePessoasAcompanhadasNoDomicilio	32
	numeroDeMenoresDeCincoAnosCadastradas	28
	numeroDiabeticosCadastrados	34
	numeroHipertensosCadastrados	40
	populacaoCadastrada	3200
	numeroPessoasComTuberculoseCadastradas	10
	numeroPessoasComHanseniaseCadastradas	22
Meta	Média atendimentos durante o pré-natal	12
	Cobertura para prevenção do câncer ginecológico	30
	Cobertura para demanda de visitas domiciliares	70
	Média de consultas de puericultura	3
	Cobertura para demanda de consultas médica	18
	Média de consultas de Hipertensos e Diabéticos	2
	Cobertura para demanda de ação coletiva de escovação dental	18
	Cobertura para demanda de consultas odontológicas	25
	Cobertura para demanda de consultas odontológicas às gestantes	100
	Cobertura para demanda de consultas de enfermagem	6
	Média de consultas para vigilância (tuberculose e hanseníase)	2

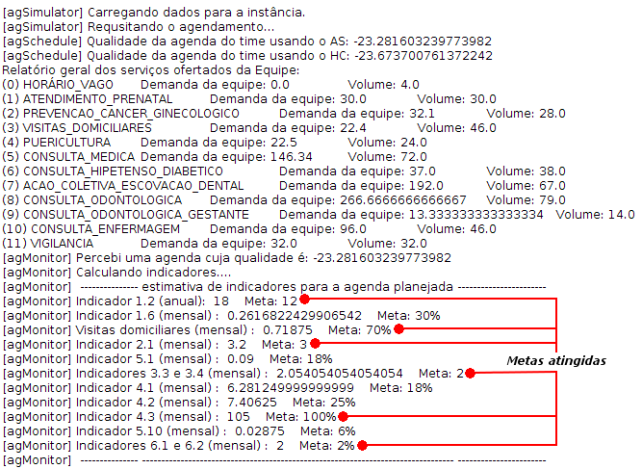


Figura 4. Execução com a instância 1

exatamente a lacuna detectada. A Figura 6 atesta que o número de profissionais foi suficiente para garantir o cumprimento das metas estimadas. O valor de qualidade da agenda gerado pela abordagem mostra o grau de satisfação que o Agente de Agendamento pode obter ao otimizar as agendas e, inclusive, subsidiar uma possível sugestão para o número de profissionais ideal de uma equipe de forma a tornar as metas factíveis.

Esse resultado ilustra a adequação da abordagem em relação à área de planejamento e monitoramento. Nas figuras 4, 5 e 6 é possível ver a atuação dos agentes especificados.

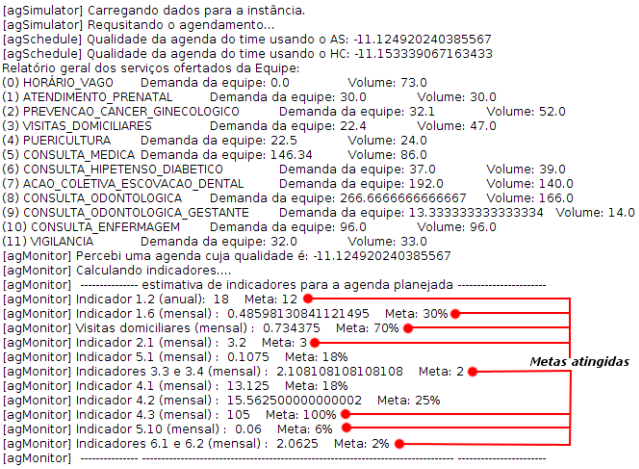


Figura 5. Execuções com a instância 2

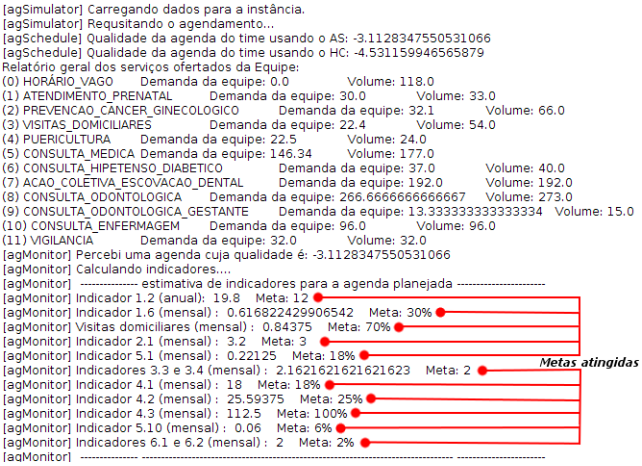


Figura 6. Execuções com a instância 3

7. Considerações finais

O PMAQ é um programa que objetiva a criação de um ciclo contínuo para o crescimento do acesso e a melhoria da qualidade dos serviços prestados, visando criar nas equipes uma cultura de aperfeiçoamento contínuo. A proposta de uma arquitetura de um sistema inteligente apresentada neste artigo tem a capacidade de atuar no planejamento e monitoramento dos indicadores das EAB, apoiando-as no processo de gerenciamento de suas ações. O modelo gerado possui como principal diferenciação dos trabalhos relacionados a capacidade de atuar na AB, especificamente no PMAQ, utilizando métodos inteligentes para auxiliar na implantação e manutenção do Programa. Por meio de uma formalização do problema foi possível codificar o Agente de Agendamento e os resultados obtidos mostram que é possível usar a orientação por metas para realizar um planejamento dos serviços ofertados priorizando-as para o cumprimento dessas demandas. Com o Agente de Indicadores foi possível ilustrar a utilidade da abordagem em relação às estimativas e alcance dos indicadores estabelecidos pelo Programa a partir do planejamento de demandas. O trabalho possui limitações em relação ao monitoramento dos indicadores estabelecidos pelo PMAQ que não foram contemplados na presente abordagem, pois seriam necessários dados específicos encontrados apenas em prontuários. Outra limitação iden-

tificada é a de construir instâncias a partir de dados reais e abertos, pois o Ministério da Saúde disponibiliza os dados associados ao município e não às UBS.

Como trabalhos posteriores, almeja-se realizar simulações de atendimentos com base nas agendas planejadas. A ideia é aplicar as agendas otimizadas de forma que sejam coletadas informações sobre o comportamento evolucionário dos indicadores ao longo das mudanças ocorridas no ambiente. Para tanto, é necessário criar um modelo evolucionário desse ambiente e incluir as diversas situações encontradas no dia a dia de unidades básicas de saúde, tais como: falta de médicos, demanda elevada de pacientes, influência política nas triagens para atendimentos etc. Diante disso é possível mostrar o que acontece com os indicadores em cada situação gerada e, dessa forma, a proposta pode tornar-se um importante instrumento de estudo para viabilização do PMAQ em diferentes cenários especificados. Além disso, visa-se também especificar os agentes não detalhados neste trabalho. Por fim, agradecemos os suportes financeiros fornecidos pela Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) por meio do edital PPSUS-11/2013 e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por meio de bolsa de estudo.

Referências

Barbosa, D. C. M. and Forster, A. C. (2010). Sistemas de informação em saúde: a perspectiva ea avaliação dos profissionais envolvidos na atenção primária à saúde de ribeirão preto, são paulo. *Cad. saude colet*, pages 424–33.

Bordini, R. H. and Hübner, J. F. (2005). A java-based agentspeak interpreter used with saci for multi-agent distribution over the net. http://jason.sourceforge.net.

Brasil, M. d. S. (2002). Portaria n. 1101, de 12 de junho de 2002: Estabelece parâmetros assistenciais do sus. *Diário Oficial da União, Brasília*, 139(112).

BRASIL, M. d. S. (2012). *Programa nacional de melhoria do acesso e da qualidade da atenção básica (PMAQ): manual instrutivo*. Ministério da Saúde: (Série A. Normas e Manuais Técnicos).

Braun, L., Wiesman, F., Herik, v. d. J., and Hasman, A. (2005). Agent support in medical information retrieval. In *Working notes of the IJCAI-05. Workshop on agents applied in health care*, pages 16–25.

Ladeira, F. (2011). *Acesso e Qualidade! Atenção básica ajusta foco em sua missão*. Revista Brasileira de Saúde da Família, Brasília, n.29, p. 31-37.

Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009). Environment programming in cartago. In *Multi-Agent Programming:*, pages 259–288. Springer.

Russell, S. and Norvig, P. (2003). Inteligência artificial: uma abordagem moderna. ed. *Campus, 2ª Edição. São Paulo, Brazil*.

Silva, R. C., Forster, A. C., Alves, D., Ferreira, J. B., and Sant'Anna, S. C. (2012). Ferramenta computacional para programa de melhoria da atenção básica (pmaq-ab). In *Atas do XIII Congresso Brasileiro de Informática em Saúde*.

Vermeulen, I. B., Bohte, S. M., Elkhuizen, S. G., Lameris, H., Bakker, P. J., and Poutré, H. L. (2009). Adaptive resource allocation for efficient patient scheduling. *Artificial intelligence in medicine*, 46(1):67–80.

Wooldridge, M. (2008). *An introduction to multiagent systems*. Wiley. com.

Auxiliando os Agentes de Software a Selecionar Melhor seus Parceiros

Elane Cristina da R. C. Saraiva[1], Viviane Torres da Silva[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Pauí (IFPI)
R. Francisco Urquiza 462 - 64800-000 – Floriano-PI - Brazil

[2]Universidade Federal Fluminense (UFF)
R. Passo da Pátria 156(E) - 24210-240 – Niterói – Brazil

elanecrc@gmail.com, viviane.silva@ic.uff.br

Abstract. *This paper proposes three mechanisms able (i) to help an agent to identify how agent evaluates the behavior of its partners; (ii) to estimate the reputation that the agent using the approach will probably receive after interacting with such partner and (iii) to calculate the final reputation of agent joining the reputation as a service provider and the reputation that the agent will receive if interact with this evaluator*.

Resumo. *Este trabalho propõe três mecanismos capazes de (i) ajudar um agente a identificar como outro agente avalia o comportamento de seus parceiros; (ii) estimar o valor da reputação que o agente usando a abordagem provavelmente irá receber depois de interagir com o avaliador e; (iii) calcular a reputação final do agente unindo a reputação como provedor de serviço e a reputação que o agente ao usar o mecanismo receberá se interagir com tal agente*.

1. Introdução

Agentes heterogêneos interagem em Sistemas Multiagentes para alcançar alguns objetivos em comum. Por serem entidades autônomas e orientadas a objetivos, os agentes são capazes de tomar suas próprias decisões e suas ações individuais podem ser prejudiciais aos seus parceiros. Considerando que os agentes precisam colaborar uns com os outros para alcançar seus próprios objetivos e os objetivos dos Sistemas Multiagentes onde estão inseridos, há uma necessidade de mecanismos que os ajudem a decidir com quem interagir e evitar parceiros mal-intencionados.

Sistemas de Reputação coletam, distribuem e agregam feedbacks sobre o comportamento passado dos participantes. Os agentes são capazes de avaliar o comportamento de seus parceiros, armazenar tais informações da reputação, tornar a sua experiência à disposição de outros agentes e usar essa informação para decidir sobre seus futuros parceiros. Os valores de reputação ajudam os agentes não só a decidir em quem confiar, mas também a dissuadir os participantes desonestos.

Diversos Sistemas de Reputação têm sido propostos e alguns deles como [Liu and Sun, 2010][Liu et al 2011][Noorian, Marsh and Fleming, 2011][Yang et al, 2009] e [Zhang and Cohen, 2008] se preocupam em identificar testemunhas confiáveis/consultores e detectar avaliações colaborativas injustas. No entanto, eles não se preocupam com o valor da reputação que o agente pode receber se ele interagir com o melhor parceiro através da avaliação dessas testemunhas/consultores. Atualmente, nos Sistemas de Reputação, os futuros parceiros são selecionados com base somente em

seus valores de reputação que são avaliados de acordo com a qualidade dos serviços prestados nas interações passadas. No entanto, esta não é a única métrica importante para ser utilizada quando se seleciona um futuro parceiro.

Desde que Sistemas de Reputação estimulem a disseminação de valores de reputação entre os agentes, a avaliação feita pelos parceiros sobre o comportamento de um agente pode ser usada por um terceiro para selecioná-lo ou descartá-lo em futuras interações. É importante ter um bom valor de reputação para ser capaz de participar de futuras interações e colaborar com outros agentes. Portanto, um agente pode levar em consideração não só sobre a qualidade dos serviços prestados, mas também sobre como os seus parceiros avaliam o seu comportamento (considerando que as avaliações dos agentes são transmitidas a outros agentes). Unindo o comportamento estimado do agente como avaliador e como prestador de serviço, um agente pode tomar uma decisão mais precisa ao escolher os seus futuros parceiros.

Neste contexto, este trabalho propõe três mecanismos capazes de: (i) ajudar o agente a identificar como outro agente avalia o comportamento de seus parceiros; (ii) estimar o valor de reputação que o agente receberá se ele interagir com tal avaliador; (iii) calcular a reputação final do agente utilizando duas informações, prestação de serviço e a forma de avaliar seus parceiros. O primeiro mecanismo analisa um conjunto de valores de reputação fornecido pelo avaliador aos seus parceiros e estima o padrão de comportamento na avaliação de seus parceiros. O segundo mecanismo, com base no padrão de comportamento identificado, calcula o valor de reputação que o agente receberá e o grau de certeza do valor calculado baseado na diferença entre os valores de reputação. O terceiro mecanismo calcula a reputação final do agente utilizando a reputação como provedor de serviço e a reputação que o agente poderá receber se interagir com tal avaliador. Ao utilizar o nosso mecanismo, o agente poderá selecionar o melhor parceiro de acordo com o seu critério de avaliação, podendo priorizar a reputação como provedor de serviço ou a reputação que ele poderá receber se interagir com o futuro parceiro.

Observe que quanto mais relacionados estão os valores de reputação considerados, mais preciso será o comportamento estimado. Consideramos que os valores de reputação estão relacionados quando tais valores (valores de avaliações fornecidos aos agentes) participam do mesmo tipo de interação. Diferentes interações pressupõem comportamentos diferentes e, consequentemente, os valores de reputação avaliados com base nos comportamentos que não estão relacionados também são diferentes.

Este artigo está estruturado da seguinte forma. A próxima seção apresenta a definição da situação a ser utilizada pelo agente ao desempenhar um papel. A seção 3.0 descreve os mecanismos propostos neste trabalho. Utilizamos um exemplo simples de *e-commerce* para exemplificar a nossa abordagem. Na seção 4.0, apresentamos uma discussão sobre os benefícios do uso desta proposta. A seção 5.0 descreve brevemente alguns trabalhos relacionados e a seção 6.0 conclui e apresenta alguns trabalhos futuros.

2. Definindo Situação

Os mecanismos propostos neste artigo assumem que os valores de reputação considerados foram fornecidos pelos avaliadores aos seus parceiros em uma situação particular. É importante considerar que os valores de reputação são fornecidos aos

agentes que participam do mesmo tipo de situação, uma vez que só é possível prever um futuro valor de reputação quando comportamentos similares são realizados pelos parceiros. Consideramos que um agente tende a avaliar comportamentos semelhantes de forma semelhantes e, portanto, fornecer valores de reputação semelhantes e coerentes aos seus parceiros. A Def. 01 apresenta a definição de situação utilizada nesta abordagem e anteriormente proposta em [Silva and Hermoso and Centeno, 2009].

Definição 1. Temos uma situação, $S_s^{Ai} = (A_i, R_j, I_k, t)$, quando $A_i \in A$ é um agente pertencente a um conjunto de agentes; $R_j \in R$ representa um papel desempenhado por A_i e; $I_k \in I$ representa uma interação (i.e, uma mensagem enviada ou recebida) realizada por A_i desempenhando um papel R_j em um determinado momento t.

Ao restringir os valores de reputação a uma mesma situação os mecanismos são capazes de identificar o padrão de comportamento a ser seguido pelo avaliador, prever o valor de reputação que um agente receberá e calcular a reputação final do avaliador se ele participar do mesmo tipo de situação S_s. No entanto, note que comportamentos diferentes no mesmo tipo de situação provavelmente irão gerar diferentes valores de reputação.

Por exemplo, considerando dois agentes desempenhando papel de vendedores e que participaram de uma interação relacionada à venda de um bem e não entregaram o bem com 03 dias após ter recebido o pagamento, como esperado. Supomos que o agente A entregou o bem com 04 dias de atraso e agente B com 07 dias. Em tal cenário, os dois agentes participaram do mesmo tipo de situação, mas perceba que um agente entregou o bem com mais atraso que o outro. Sendo assim, os valores de reputação que eles receberão provavelmente serão distintos uma vez que os seus comportamentos foram diferentes. Portanto, se os valores de reputação também são baseados pelo mesmo comportamento, mais preciso será o comportamento estimado do avaliador.

Note que o mecanismo proposto não compara as situações, uma vez que considera os valores de reputação. Ao utilizar o mecanismo, deve ser objetivo do agente coletar os valores de reputação relacionados à mesma situação em que os parceiros tenham executados (ou não executados) fatos similares. Quanto maior a diferença entre as situações e os valores de reputação relacionados aos comportamentos, menor a probabilidade de o comportamento estimado ser apresentado pelo avaliador.

3. Nossa Abordagem

Nesta seção, apresentamos o mecanismo para estimar o padrão de comportamento do agente avaliador (AV) e o mecanismo para estimar o valor da reputação que o agente usando o mecanismo (AM) receberá se ele interagir com o agente avaliador (AV). A fim de fazer tais estimativas, os mecanismos usam fórmulas das medidas de tendência central e medidas de dispersão [Brase and Brase, 2012]. Estas fórmulas são bem conhecidas na estatística descritiva e se preocupa com a descrição das características de distribuição de freqüência. A média (média aritmética), a mediana (pontuação média) e a moda (pontuação mais frequente) são exemplos de medidas de tendência central. Uma vez que diferentes conjuntos de dados podem ter a mesma média, o objetivo das medidas de dispersão é descobrir o quanto os números em um conjunto diferem da média de tal conjunto [Brase and Brase, 2012]. Tais medidas são capazes de avaliar a variabilidade apresentada em um conjunto de dados. Exemplos de medidas de dispersão são desvio padrão e coeficiente de variação.

Desvio Padrão: O desvio padrão indica a variação dos elementos de um conjunto a partir da média. Um desvio padrão baixo indica que os pontos de dados tendem a ser muito próximos da média, enquanto um desvio padrão elevado indica que os dados estão dispersos em uma ampla gama de valores. O desvio padrão **(DP)** é calculado de acordo com a equação 01, onde x_i representa cada elemento do conjunto, o símbolo m representa a média e n é o número de elementos do conjunto. O desvio padrão individual **(DPI_i)** indica a variação de cada elemento do conjunto em torno da média, como representado na equação 02.

$$DP = \sqrt{\frac{\sum(x_i - m)^2}{n-1}} \qquad (01) \qquad\qquad DPI_i = x_i - m \qquad (02)$$

Coeficiente de Variação: O coeficiente de variação (CV) é uma medida normalizada de distribuição de probabilidades. Esta medida mostra o grau de variabilidade em relação à média da população. O coeficiente de variação (CV) é calculado pela equação 03. O coeficiente de variação individual (CVI_i) é uma medida individual de dispersão, indicando o grau de variabilidade de cada elemento em relação à média, conforme ilustrado na equação 04:

$$CV = \frac{DP}{m} \qquad (03) \qquad\qquad CVI_i = \frac{DPI_i}{m} \qquad (04)$$

3.1 Alguns Padrões Comportamentais

Mesmo que dois agentes participem de uma mesma situação com os mesmos parceiros desempenhando os mesmos fatos, eles podem avaliar o comportamento de seus parceiros de diferentes maneiras e fornecer diferentes valores de reputação. Nesta seção, apresentamos cinco exemplos de padrões de comportamento que indicam diferentes estratégias utilizadas pelos avaliadores no cálculo do comportamento de seus parceiros. Nota-se que não é nossa intenção fornecer todos os possíveis padrões de comportamento, mas exemplificar alguns deles. Tais padrões de comportamento foram encontrados após investigar o comportamento dos avaliadores em diferentes sistemas de reputação.

Normal: todos os valores de reputação são semelhantes. O avaliador avalia todos os seus parceiros de forma semelhante. Um caso particular desse padrão ocorre quando todos os valores de reputação são exatamente os mesmos.

Ruído: o valor de reputação fornecido a um dos agentes que participou da mesma situação dos outros agentes é completamente diferente dos demais valores de reputação. Tal reputação caracteriza um erro na avaliação do comportamento do agente e erros podem ocorrer outras vezes.

Tendência: os valores de reputação fornecidos a um dado agente em todas as situações que ele tenha participado são completamente diferentes dos valores previstos aos outros agentes e esses valores são reputações semelhantes. Esse agente está sendo prejudicado ou beneficiado pelo avaliador.

Mudança de Comportamento: os valores de reputação fornecidos seguem o mesmo padrão até um determinado período no tempo *t*, mas após esse período os valores de reputação seguem outro padrão. Nesse caso, considera-se que depois de um conjunto de situações, seguindo o mesmo padrão de comportamento, o agente muda o comportamento e começa a avaliar os seus parceiros de uma forma bem diferente.

Aleatório: quando uma estratégia aleatória é utilizada, cada agente recebe um valor de reputação diferente, mesmo quando um agente participa mais de uma vez da mesma situação. Nesse caso, o agente fornece valores de reputação aleatórios.

3.2 Mecanismo para Estimar os Padrões Comportamentais

Este mecanismo utiliza o coeficiente geral de variação (CV) e o coeficiente de variação individual (CVI) para descobrir o padrão de comportamento do avaliador. Abaixo detalhamos como é usada cada métrica ao analisar os valores de reputação e descobrir o padrão comportamental do avaliador. Consideramos neste trabalho que um CV abaixo de 25% caracteriza que os valores de reputação são semelhantes e um CV acima de 25% indica que os valores são diferentes. Uma vez que assumimos que os valores de reputação foram fornecidos aos agentes que executaram fatos semelhantes em situações semelhantes e que os agentes tendem a avaliar comportamentos semelhantes da mesma forma.

Para exemplificar a nossa abordagem, usamos um aplicativo de *e-commerce*, onde agentes desempenhando papéis de compradores e vendedores interagem comprando e vendendo livros usados. A situação a ser considerada é a entrega do livro ao comprador após efetuar o pagamento do bem. Nesta aplicação, antes de um comprador interagir com um vendedor, ele solicita ao vendedor os valores de reputação que ele forneceu aos compradores com quem interagiu. Vamos demonstrar a aplicabilidade dos mecanismos concentrando-se em um agente chamado comprador E que solicitou os valores de reputação fornecidos a diferentes compradores. Cada vendedor (ou avaliador) fornece os valores de reputação dos 10 compradores com quem ele tenha interagido em tal situação. A Tabela 1 apresenta os valores de reputação fornecidos por cinco vendedores que utilizaram diferentes padrões de comportamento ao avaliar seus parceiros. A Tabela 2 apresenta os CVs do conjunto dos valores de reputação fornecidos por cada vendedor/avaliador e o CVIs de cada valor de reputação.

Tabela 1. Valores de Reputações fornecidos por cinco vendedores aos seus parceiros

Avaliadores	Avaliações dos Parceiros									
Vendedor 01 (moda = 1,5)	A	B	C	D	E	A	F	G	H	A
	1,8	1,5	1,5	2,5	2,4	1,9	2,0	2,3	2,0	1,5
Vendedor 02 (moda = 9,1)	B	C	A	D	E	A	B	E	G	H
	9,1	8,8	9,2	9,3	1,0	8,9	9,1	9,0	9,1	9,2
Vendedor 03 (moda = 3,0)	E	A	H	G	C	E	B	F	A	E
	8,0	1,5	1,5	2,0	3,0	7,9	3,0	3,0	2,0	8,1
Vendedor 04 (moda = 2,0)	H	F	A	D	H	C	G	B	C	A
	2,1	1,9	2,0	1,8	2,0	9,0	4,0	7,0	8,0	3,0
Vendedor 05 (moda = 3,0)	A	B	C	D	E	B	G	H	A	B
	1,0	3,0	7,0	8,0	4,0	3,0	9,0	3,5	9,5	0,0

Normal: se o CV é baixo significa que os valores do conjunto estão próximos uns dos outros. Portanto, se o CV e todos os CVIs (ou quase todos os CVIs) relacionados a cada reputação são baixos, significa que os valores de reputação são homogêneos (próximos da média). Assim, podemos concluir que o avaliador está seguindo o padrão de comportamento normal. Ao analisar o CV e os CVIs na Tabela 2 relacionados ao vendedor 01, o mecanismo conclui que tal agente segue o padrão normal, pois o CV é baixo e a maioria dos CVIs também são baixos.

Tabela 2. CVs e CVIs dos valores das reputações fornecidas pelos compradores aos seus parceiros

Avaliadores	Percentual dos CVIs das Reputações dos Parceiros									
Vendedor 01 (CV=19%)	A	B	C	D	E	A	F	G	H	A
	7	23	23	29	24	2	3	19	3	23
Vendedor 02 (CV=31%)	B	C	A	D	E	A	B	E	G	H
	10	6	11	12	88	8	10	9	10	11
Vendedor 03 (CV=70%)	E	A	H	G	C	E	B	F	A	E
	100	65	63	50	25	98	25	25	50	103
Vendedor 04 (CV=69%)	H	F	A	D	H	C	G	B	C	A
	49	53	51	56	51	121	2	72	96	26
Vendedor 05 (CV=70%)	A	B	C	D	E	B	G	H	A	B
	79	38	46	67	17	38	88	27	98	100

Ruído: Se o CVI de apenas um único valor de reputação é alto, isso significa que tal reputação é completamente diferente das outras e podemos concluir que é um erro. Pela análise da Tabela 2, é possível detectar um ruído na avaliação do comprador E pelo vendedor 02, pois o seu CVI é elevado em relação aos demais (i.e., distante dos outros valores). Tal agente interagiu com o comprador 02 outra vez e recebeu um valor de reputação semelhante às recebidas pelos outros agentes.

Tendência: se (i) o CV é elevado e; (ii) todos os CVIs dos valores de reputação de um dado agente são muito semelhantes e os mais elevados e; (iii) o valor de reputação mais baixo ou o mais alto é um dos valores de reputação de tal agente, indica que este agente está recebendo valores de reputação que são completamente diferentes dos outros. Tais valores de reputação são os menores ou os mais elevados e nenhum outro agente tem uma reputação similar. Podemos concluir que há uma tendência ao avaliar o comportamento desse agente de uma maneira diferente. No nosso exemplo, o vendedor 03 interagiu 03 vezes com o comprador E que recebeu valores de reputação muito semelhantes e tais valores são muito diferentes dos valores de reputação recebidos pelos outros agentes.

Podemos concluir que o vendedor 03 está priorizando o comprador E, uma vez que (i) os CVIs relacionados aos valores de reputação do vendedor 03 são muito semelhantes; (ii) o CV também é elevado, indicando que há uma discrepância nos valores de reputação fornecidos pelo vendedor 03 e; (iii) um dos valores de reputação do vendedor é a mais elevada.

Mudança de Comportamento: antes de considerar que é um caso de uma mudança de padrão de comportamento, é importante eliminar a hipótese de ser um ruído, um comportamento de tendência ou um padrão aleatório. Então, se não é nenhuma dessas situações e o CV não é baixo, o mecanismo inicia à avaliação do CV considerando apenas os três primeiros valores de reputação, a fim de descobrir um padrão. Depois disso, ele inclui outro valor reputação no conjunto e recalcula o CV. Se o valor do CV mudar drasticamente, podemos concluir que o agente mudou a forma como ele avalia o comportamento de seus parceiros. Senão, o mecanismo continua analisando os próximos valores. Em nosso cenário, podemos concluir que o vendedor 04 mudou o seu comportamento ao avaliar agente C. Antes dessa avaliação, considerando os cinco primeiros valores de reputação, os CVIs são semelhantes e o CV é baixo. Ao considerar os últimos 05 valores de reputação, o mecanismo detecta que o

padrão de comportamento real do vendedor 04 é aleatório, uma vez que estes valores são muito diferentes e o CV é elevado.

Aleatório: se o CV é elevado e (i) não há ruído; (ii) não há comportamento de tendência e; (iii) não há comportamento aleatório, pode-se concluir que os valores de reputação são dispersos e, portanto, completamente diferentes uns dos outros. Conclui-se então, que estamos lidando com uma estratégia aleatória. Ao avaliar CV e os CVIs dos valores de reputação dos parceiros do vendedor 05, nota-se que os valores estão dispersos. O vendedor 05 está seguindo o padrão aleatório e fornecendo valores de reputação aleatórios sem se preocupar com o comportamento dos agentes.

3.3 Mecanismo para Estimar o Futuro Valor de Reputação

Após definir o padrão comportamental do agente avaliador **(AV)**, é possível a utilização de tais informações juntamente com os valores de reputação para estimar a reputação que o agente que utiliza o mecanismo **(AM)** receberá se ele interagir com o AV. Além de fornecer o futuro valor de reputação, o mecanismo também fornece um grau de certeza de tal valor. Para estimar a reputação, o mecanismo adota diferentes estratégias que variam de acordo com o padrão de comportamento identificado.

Normal: se o AM ainda não interagiu com o AV, a reputação é a média estimada considerando todos os valores de reputação ou a moda (se houver uma pontuação que ocorre com mais frequência). Se existir moda, acreditamos que existe uma tendência em fornecer o valor relacionado com a moda. O grau de certeza da Equação 05 é avaliado de acordo com o CV. Quanto mais baixo for o CV maior é o grau de certeza.

$$\textbf{GrauDeCerteza} = 100\ \% - \textbf{CV} \qquad (05).$$

No caso do AM já ter interagido com o AV anteriormente, a reputação estimada é a média dos valores de reputação que o agente recebeu em situações anteriores. Uma vez que assumimos que o AM irá interagir com o AV em situação semelhante, acreditamos que o AV irá atribuir, provavelmente, valores de reputação semelhantes. O grau de certeza é avaliado através dos CVIs relacionados com os valores de reputação que o agente tem recebido, conforme a Equação 06.

$$\textbf{GrauDeCerteza} = 100\ \% - (CVI_1 + \ldots CVI_n) / n \qquad \textbf{(06).}$$

Em nosso exemplo, o valor estimado da reputação que o comprador E receberá se ele interagir com o vendedor 01 um é 2,4, uma vez que tal reputação é fornecida ao agente E pelo vendedor 01 na interação passada. O grau de certeza é 76 % pois o CVI=24%.

Ruído: neste caso, é importante avaliar três alternativas diferentes. Em primeiro lugar, se o valor do ruído não tiver sido atribuído ao AM, a reputação estimada e o grau de certeza são avaliados depois de descartar o ruído e deve utilizar o mesmo método indicado no caso do padrão de comportamento normal. O ruído é descartado porque caracteriza um erro. Em segundo lugar, se o valor da reputação do ruído foi atribuído ao AM e tal agente tem interagido com o AV em outras situações e tem recebido os valores de reputação semelhantes aos valores recebidos por outros agentes, também consideramos a reputação ruído como um erro. Nota-se que isto não é uma tendência, já o AM interagiu com o AV em outras situações e não recebeu valores reputação semelhantes ao ruído. Assim, o ruído deve ser descartado e deve ser utilizada a mesma

abordagem descrita no caso do padrão normal. Por outro lado, se a reputação do ruído foi atribuída ao AM e esse agente não interagiu com o AV em qualquer outra situação, não podemos afirmar que o valor é um ruído ou será uma tendência. Optamos por considerar como uma tendência e estimar a reputação e o grau certeza seguindo o padrão tendência.

Em nosso exemplo, nos concentramos sobre o valor de reputação que o comprador E receberá após interagir com o vendedor 02. O valor de reputação estimado após o descarte do ruído é 9,0 (valor de reputação que o comprador E recebeu numa interação passada com o vendedor 02) e o grau de certeza é de 91% (já que o CVI de tal agente é 9%).

Tendência: se o AM é quem tem recebido diferentes valores de reputação (valores mais baixos ou valores mais altos), o valor estimado da reputação é calculado usando apenas os valores tendenciosos e seguindo a estratégia do padrão normal. Uma vez que existe uma tendência em avaliar o agente usando os valores distantes dos demais, o valor estimado da reputação deve seguir essa tendência. No entanto, se o AM não é quem está sendo prejudicado ou beneficiado, os valores tendenciosos são descartados e a estratégia do padrão normal é utilizada sobre demais valores.

Portanto, o valor da reputação que o comprador E receberá após a interação com o vendedor 03 é 8,0 (a média das reputações que tal agente recebeu) e o grau de certeza é 99% (o CV dos três valores de reputação que o agente E é 1%).

Mudança de Comportamento: ao identificar mudança de comportamento no padrão das avaliações, é necessário descobrir o último padrão de comportamento a ser seguido pelo avaliador antes de estimar a reputação. Uma vez que tal padrão for encontrado, a reputação estimada é avaliada seguindo a estratégia definida por esse padrão de comportamento. No nosso exemplo, o valor de reputação estimado que o agente E receberá após a interação com o vendedor 04 é 6,2, o que representa a média dos últimos 05 valores. O grau de certeza é 58%, uma vez que só consideram o CVIs dos últimos 05 valores.

Aleatório: no caso de o AV fornecer valores aleatórios, a nossa proposta é a utilização da mesma estratégia utilizada pelo padrão normal, em que a média (ou a moda) é usada para avaliar a reputação estimada. Em caso de padrão aleatório, o CV é sempre elevado e o grau certeza é sempre baixo. No nosso exemplo, o valor da reputação estimada do comprador E após a interação com o vendedor 05 é, portanto, 3,0 (a moda) e o grau certeza é 30%.

3.4 Mecanismo para Calcular a Reputação Final de um Agente

Como comentado anteriormente, ao escolher interagir ou não com um parceiro, um agente pode considerar não só a qualidade dos serviços prestados por tal parceiro, mas também sobre como ele irá avaliar o seu comportamento após a interação. Assim, propomos a utilização da equação 07 em que une tais informações e calcula a reputação final do futuro parceiro.

$$\textbf{ReputacaoFinal} = (\alpha * \textbf{ReputacaoProvedorServico}) + (\beta * \textbf{ReputacaoEstimada} * \textbf{GrauDeCerteza}) \quad (07)$$

A variável *ReputacaoProvedorServico* é a reputação referente à prestação de serviço do avaliador (AV) podendo ser utilizado qualquer modelo para calcular tal

reputação, a *ReputacaoEstimada* é o valor de reputação que o agente usando o mecanismo deve receber do AV se ele interagir com esse parceiro e o *GrauDeCerteza* está associado ao referido valor. As variáveis α e β estão relacionadas com a importância que o AM considera a cada valor de reputação. Por exemplo, se o AM está mais interessado em receber uma reputação muito boa após interagir com o AV ao invés de um serviço de alta qualidade, este agente deve fornecer um valor $\beta > \alpha$. O agente usando a abordagem pode ponderar tais valores de acordo com o seu critério de seleção de seus parceiros.

4. Discussão

Os mecanismos são independentes do Sistema de Reputação utilizado pela aplicação, uma vez que depende apenas de (i) valores de reputação (numéricos e positivos) ordenados por tempo; (ii) identificação dos agentes que receberam tal valor e; (iii) identificação do agente que está utilizando os mecanismos (a fim de descobrir se ele deve ou não interagir com o avaliador). Ambos os mecanismos foram implementados em Java, tal como ilustrado na Figura 1.

Cada subclasse da classe PadraoComportamento é responsável por (i) informar se o avaliador está seguindo o padrão ou não (através da execução de método de execute()); (ii) estimar o valor da reputação estimada (através do método estimadaReputacao()) e; (iii) calcular o grau de certeza do valor (através do método grauDeCerteza()). A classe RAVPAC é a principal classe e é responsável pelo cálculo da reputacao final conforme o mecanismo proposto neste trabalho.

Note que outros padrões de comportamentos podem ser facilmente implementados. Para definir outro padrão de comportamento é necessário (i) estender a classe PadraoComportamento e implementar os seus métodos abstratos e; (ii) estender a classe RAVPAC e implementar o método incluirComportamento() para incluir todos os comportamentos que o agente é capaz de analisar. Ao calcular a reputação como provedor de serviço através do método estimaReputacaoParceiro() poderá ser implementada qualquer proposta disponível na literatura. No entanto, perceba que o cálculo da reputação final não poderá ser alterado.

A fim de avaliar a aplicabilidade da nossa abordagem implementamos uma simulação usando Jason [Bordini and Hubner and Wooldridge, 2007]. Na simulação, 20 agentes compram livros e 10 agentes vendem livros: 05 dos vendedores são bons prestadores de serviços e 05 são prestadores de serviços ruins (eles atrasam a entrega dos livros). Todos os 20 compradores são bons prestadores de serviços. Além disso, cada vendedor avalia o comportamento de seus parceiros, seguindo um dos padrões comportamentais descritos na seção 3.1 e 10 compradores usam a nossa abordagem para ajudar na seleção de seus futuros parceiros.

Cada comprador que utiliza a nossa abordagem calcula dos vendedores (i) a reputação como prestador de serviço (RPS); (ii) o valor da reputação estimada que o comprador receberá se interagir com o vendedor (RE); (iii) o valor da reputação final do vendedor considerando $\alpha = \beta = 50$ % (RF), (iv) $\alpha = 30$ % e $\beta = 70$ % (RFB) e (v) $\alpha = 70\%$ e $\beta = 30$ % (RFA). O Gráfico 01 mostra essas informações sobre o ponto de vista do comprador E. Devido ao limite de espaço estamos mostrando dados relativos de apenas 05 vendedores. Observe que os vendedores 01 e 02 possuem a RPS elevada e a RE associada também é elevada (RF e FRB estão bem próximas). Isso significa que eles

são bons prestadores de serviços e, provavelmente, irão fornecer valores elevados de reputação ao comprador E. No entanto, o vendedor 04 possui a RPS elevada, mas apresenta a RE baixa (RFA é maior do que RFB). Por outro lado, o vendedor 05 possui uma RPS média, mas, provavelmente, fornecerá uma reputação elevada (RFB é superior a FR e RFA). Assim, se os agentes 01 e 02 estiverem ocupados, o comprador que utiliza a nossa abordagem deve preferir selecionar agente 05 já que além de receber um bom serviço e ele será bem avaliado.

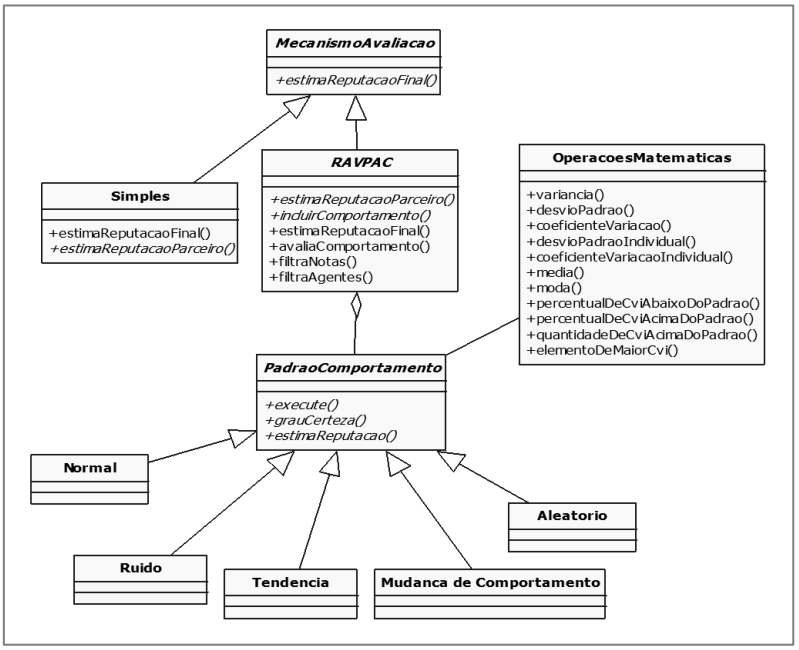


Figura 1 Diagrama de Classe

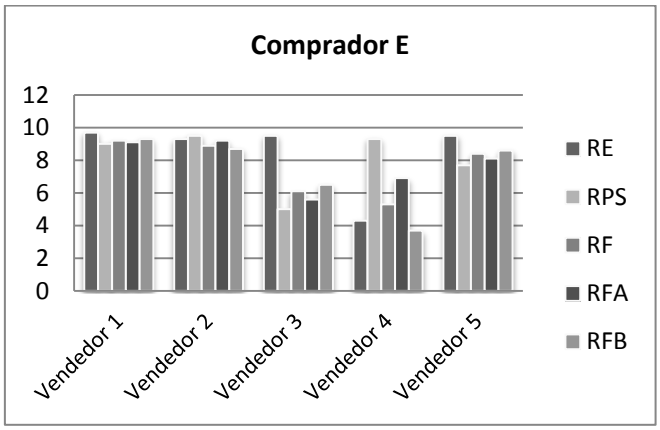


Gráfico 1. Reputação como Provedor de Serviço e Reputações Finais.

Na nossa simulação os vendedores avaliam o comportamento de seus parceiros baseado apenas em suas experiências diretas. Além disso, não consideramos falsos testemunhos fornecidos por vendedores sobre o comportamento dos compradores com os quais interagiram. Estamos supondo que os valores de reputação fornecidos aos compradores representam os valores de reputação que vendedores têm realmente fornecido aos seus parceiros. Podemos considerar que os compradores estão recebendo os valores de reputação certificados fornecidos pelos vendedores como proposto em [Huynh, Jennings and Shadbolt, 2004].

5. Trabalhos Relacionados

Em Medic (2012) é apresentado um estudo onde comparam 19 modelos de confiança e reputação. Em todos esses modelos os agentes são capazes de avaliar o comportamento de seus parceiros e utilizar essas informações ao escolher seus futuros parceiros. No entanto, nenhum desses modelos se preocupa sobre como os agentes avaliam seus parceiros e como tal avaliação pode beneficiar ou prejudicar os agentes que interagem com eles. Ao usar essas abordagens, os agentes escolhem os seus parceiros com base unicamente nas reputações de acordo com a prestação de serviços e não consideram como os agentes avaliam os seus parceiros.

Apesar das abordagens como [Liu and Sun, 2010][Liu et al 2011][Noorian, Marsh and Fleming, 2011][Yang et al, 2009] e [Zhang and Cohen, 2008] se preocuparem com a identificação de testemunhos confiáveis/consultores e detectar avaliações colaborativas injustas, elas não se preocupam com o valor da reputação que o agente pode receber se ele interagir com o melhor parceiro. Essas abordagens são capazes de identificar os agentes que estão fornecendo falsos testemunhos sobre o comportamento de seus parceiros e de excluir esses agentes ao pedir novos depoimentos. No entanto, em caso de testemunhos confiáveis, os valores de reputação que essas testemunhas prestam aos seus parceiros não são usados na escolha de um futuro avaliador. Nota-se que mesmo sendo de confiança, dois agentes podem apresentar diferentes valores de reputação sobre comportamentos semelhantes no mesmo tipo de situação. Por exemplo, do ponto de vista de um vendedor que recebe o pagamento com 02 dias após a entrega de um bem, isto pode ser imperdoável. Este vendedor irá fornecer um valor muito baixo para reputação como prestação de serviço do comprador. Mas outro vendedor pode fornecer um valor de reputação razoável ao considerar que o atraso de 02 dias, no pagamento do bem, não é tão grave.

Não devemos esquecer que o comportamento de um agente será avaliado de acordo com o ponto de vista do seu parceiro. Se considerarmos que o parceiro é um parceiro de confiança e que o nosso agente apresentará um bom comportamento, o nosso agente não deverá receber um valor de reputação ruim. Portanto, ao escolher o melhor parceiro para interação, é importante considerar o valor de reputação do agente como prestador de serviços, mas também é importante considerar o valor de reputação que tal agente pode fornecer ao nosso agente na próxima interação.

6. Conclusões e Trabalhos Futuros

Mecanismos de reputação atuais suportam apenas a avaliação do comportamento de parceiros como prestadores de serviços. O histórico de tais informações, bem como a propagação de tais valores de reputação pode ser utilizado para selecionar os futuros parceiros. Em sistemas onde os agentes devem interagir uns com os outros para atingir seus objetivos e parceiros são selecionados com base em seus valores de reputação, os agentes podem se preocupar com o seu comportamento em tais interações e também sobre as avaliações que os seus parceiros vão fazer sobre o seu comportamento. Portanto, há uma necessidade de um mecanismo que permita estimar o comportamento de um agente quando avaliam os seus parceiros e de um mecanismo para estimar a reputação que um agente utilizando o mecanismo irá receber após a interação com o avaliador e uma maneira de usar essa informação ao selecionar futuros parceiros. Para

responder a essas necessidades foram definidas as abordagens apresentada neste trabalho.

Como trabalhos futuros estamos em processo de definição de mecanismos de incentivos para estimular os agentes a fornecer informações sobre seus parceiros. Por exemplo, os agentes que não oferecem tais informações podem ser automaticamente descartados como futuros parceiros. Supondo que é importante interagir com outros agentes para atingir os objetivos, um agente será estimulado a cooperar. É também nossa intenção fornecer mecanismos para descobrir os valores de reputação não verdadeiros. Esses valores podem ser facilmente detectados se a abordagem também pede reputações certificadas. Finalmente, mas não menos importante, atualmente, os mecanismos propostos supõe que as situações em que os agentes têm participado são do mesmo tipo. O nosso objetivo é estender o mecanismo que estima o valor de reputação de um agente para ser capaz de considerar a diferença entre as situações e usar essa informação ao estimar as reputações.

REFERÊNCIAS

Brase, C.H. and Brase, C.P. (2012) *Understanding Basic Statistics*, Brooks Cole, 6th Edition.

Bordini, R. and Hubner, J. and Wooldridge, M. (2007) *Programming Multi-agent Systems in Agentspeak using Jason*. John Wiley & Sons, vol. 01. p. 273, 2007.

Huynh, T. D., Jennings, N. R., Shadbolt, N. R. (2004) FIRE: An Integrated Trust and Reputation Model for Open Multi-Agent Systems, In *Proceeding of ECAI*, pp.18–22.

Liu, Y. and Sun, Y. (2010) *Anomaly Detection in Feedback-based Reputation Systems through Temporal and Correlation Analysis*, in Proc. of 2nd IEEE International Conference on Social Computing.

Liu, S et al. (2011) *iCLUB: An Integrated Clustering-based Approach to Improve the Robustness of Reputation Systems*. In Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems, vol 3, pp. 1151–1152.

Medic, A. *Survey of Computer Trust and Reputation Models – The literature Overview*. (2012) International Journal of Information and Communication Technology Research, vol. 2, nº 3, pp. 254-275.

Noorian, Z. and Marsh, S. and Fleming, M. (2011) *Multi-layer Cognitive Filtering by Behavioral Modeling*, in Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 871–878.

Silva, V; Hermoso, R.; Centeno, R. (2009) A Hybrid Reputation Model Based on the Use of Organization, In *Coordination Organization, Interaction and Norms III*, LNAI 5428.

Yang, Y et al. (2009) *Defending Online Reputation Systems against Collaborative Unfair Raters through Signal Modeling and Trust*. In Proceedings Annual Symposium on Applied Computing, Honolulu, ACM Press, pp. 1308–1315.

Zhang, J. and Cohen, R. (2008) *Evaluating the Trustworthiness of Advice about Seller Agents in E-Marketplaces: A Personalized Approach*. Electronic Commerce Research and Applications, 7(3), pp. 330–340.

Um Agente Inteligente para Simulação de Voo Usando Jason e X-Plane

Tielle da Silva Alexandre, Carlos Eduardo Pantoja

[1] CEFET/RJ - Campus Nova Friburgo
Av. Gov. Roberto da Silveira, 1900 – Prado – 22.635-000 –
Nova Friburgo – RJ – Brasil

tiellesa@gmail.com, pantoja@cefet-rj.br

Abstract. *This paper presents an intelligent agent which connected to a professional flight simulator, performs the pilot function in the task of controlling an airplane to achieve the missions of takeoff; reach determined point based on the latitude longitude and altitude; and remain stable during the flight. This work also presents an integration between Jason framework and the X-Plane flight simulator.*

Resumo. *Este trabalho apresenta um agente inteligente que, conectado a um simulador de voo profissional, realizará a função do piloto na tarefa de controlar uma aeronave na missão de levantar voo; atingir determinado ponto levando em consideração a latitude, longitude e altitude; e se manter estável durante o voo. O trabalho também apresentará uma integração entre o framework Jason e o simulador de voo X-Plane.*

1. Introdução

Um agente inteligente pode ser tanto físico como virtual e tem a capacidade de atuar em um ambiente, que pode ser físico, como um robô atuando através de efetuadores, ou simulado, atuando através de execução de comandos de software. Além disso, possui objetivos de satisfação e subsistência individuais, pode se comunicar com outros agentes, disponibilizar serviços e tem pró-atitude por satisfazer seus objetivos [Wooldridge 2009].

Um agente pode possuir crenças, que são as ideias que este tem do ambiente e sua compreensão da atmosfera em que ele está presente. Os desejos são os eventuais planos que o agente realiza a partir de suas crenças, entretanto, o agente só irá atuar sobre algum plano se estiver comprometido em atingir determinado objetivo, apesar de desejá-lo. Já as intenções se dão quando um agente se compromete em seguir um plano para atingir determinado objetivo [Bratman 1987].

Os Veículos Aéreos Não Tripulados (VANT) têm sido cada vez mais utilizados em diversas áreas de conhecimento, com o objetivo de auxiliar em tarefas onde haja riscos a vidas humanas. Primeiramente idealizados para realizarem missões de estratégia militar, hoje os VANT realizam também operações de monitoramento agrícola, ambiental, segurança e defesa civil. Um VANT não necessita de pilotos embarcados e pode ser guiado à distância, por meio eletrônico ou computacional, manipulado especificamente por pilotos treinados ou serem completamente autônomos.

Neste processo, visto que a pilotagem da aeronave é feita a distância, pode-se detectar problemas como o de comunicação ou de falha humana que podem comprometer o voo e a missão imposta à aeronave. Além disso, a decolagem e aterrissagem são ações críticas que demandam experiência do piloto. Atualmente, mesmo em modelos com piloto automático acionado por controle remoto a determinada altura, a decolagem e aterrissagem ainda é um processo unicamente manual e dependente de um piloto qualificado, como o AG PLANE da AGX Tecnologia.

O VANT necessita ainda de uma estação de controle, podendo essa ser portátil ou móvel. Englobando todos os equipamentos para se operar um VANT, um defeito ou até mesmo a falta de energia na estação de controle compromete toda uma operação. Dessa forma, um VANT inteligente seria uma solução para se contornar tais problemas. Contudo, devido o alto custo do projeto de construção de aeronaves, faz-se necessária uma fase de simulação e testes.

Diversos trabalhos utilizam o paradigma orientado a agentes e visam a autonomia de um VANT, como o caso do [Wallis et al. 2002], que utiliza a linguagem orientada a agentes JACK [Winikoff 2005] para prover um ambiente de programação simples para comportamentos táticos de voo; e [Huff et al. 2003], onde um simulador representa um VANT como um conjunto de agentes que podem simular diferentes abordagens de planejamento de voo.

O objetivo deste trabalho é desenvolver um agente de software capaz de realizar um voo de maneira autônoma em um simulador profissional, utilizando a comunicação de pacotes UDP para: receber informações acerca do ambiente através de crenças; e poder executar ações no simulador através de comandos para a aeronave simulada. Para implementação do agente serão utilizados: o simulador de voo profissional X-Plane, para simulação do agente inteligente; uma API [Cantoni 2010] em Java para comunicação entre o simulador de voo e o ambiente simulado do agente; e, por fim, o framework Jason para a codificação do agente inteligente.

Este artigo está estruturado da seguinte forma: na seção 2 apresenta-se a metodologia utilizada na integração das tecnologias utilizadas no trabalho; na seção 3 é apresentada a implementação do ambiente e do agente inteligente e também serão apresentados os resultados obtidos através da integração com o simulador de voo; na seção 4 serão apresentados alguns trabalhos relacionados; e por fim, na seção 5 é apresentada a conclusão e alguns trabalhos futuros.

2. A Metodologia

Nesta seção serão apresentados os conceitos básicos sobre as tecnologias utilizadas no desenvolvimento do projeto, onde, o ambiente simulado utilizado é o simulador X-Plane integrado a linguagem de programação Java e para a implementação do agente é utilizado o framework Jason. Além disso, será apresentada a metodologia de integração entre o agente e o simulador de voo.

O X-Plane é um simulador de voo de alta realidade, que pode ser usado para prever as qualidades de voo de aeronaves de asa fixa e rotativa com precisão. O simulador prevê o desempenho e manuseio de quase qualquer aeronave, dessa forma é uma ferramenta útil para pilotos treinarem e se aperfeiçoarem, para engenheiros testarem o comportamento

aéreo de novas aeronaves, bem como a exploração da dinâmica de voo de aeronaves. O X-Plane foi escolhido, pois é um simulador de alta precisão de aeronaves comercias, assim como é utilizada para testes de comportamento aéreo e das dinâmicas de voos de protótipos em fase de estudo.

O Jason é um framework baseado em Java e AgentSpeak para desenvolvimento de sistemas multiagentes onde a linguagem AgentSpeak representa uma tentativa de refinar as características principais da arquitetura BDI. O Jason foi escolhido por ser um framework orientado a agentes que utiliza uma arquitetura cognitiva, além de ser uma linguagem de licença livre [Bordini et al. 2007].

O agente inteligente usando o framework Jason deve ser capaz de se comunicar com o simulador de voo através de comunicação de pacotes UDP em uma rede de computadores. A comunicação UDP permite tanto que o agente inteligente tenha acesso às variáveis relacionadas ao ambiente e a aeronave como crenças, que serão usadas na deliberação das ações de voo, quanto no envio de instruções para o simulador, com a finalidade de controlar as funções da aeronave. A figura 1 ilustra a metodologia de comunicação entre o agente e o simulador.

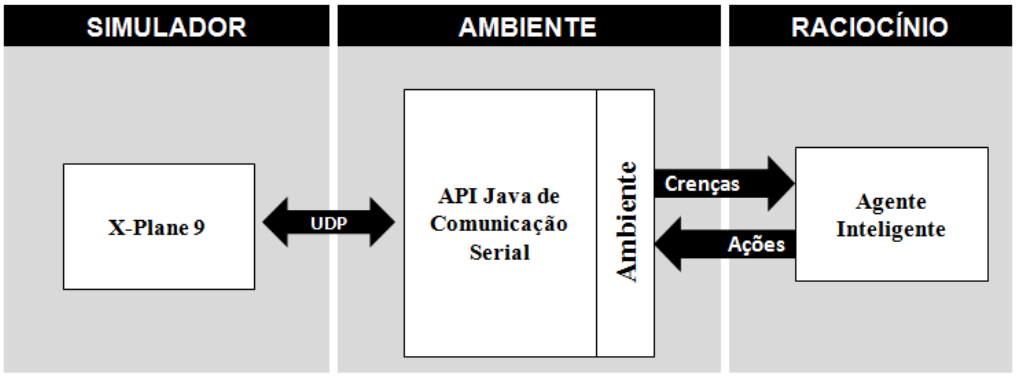


Figure 1. A metodologia de integração entre as tecnologias.

A integração entre o framework Jason e o X-Plane funciona através da importação de uma API em Java para comunicação Serial na classe de ambiente do Jason, que é responsável por receber os pacotes UDP vindos do simulador, interpretá-los e enviá-los ao ambiente do agente, onde serão transformados em crenças. Da mesma forma, o ambiente identifica as ações externas do agente transformando-as em comandos, que são enviados ao simulador de voo através de pacotes UDP.

3. A Implementação

Nesta seção será apresentada a implementação do agente inteligente em Jason, que é capaz de pilotar uma aeronave modelo P180 Avanti Ferrari e possui um plano de voo que é responsável por identificar variáveis de aceleração, posicionamento global, velocidade do vento, para que seja possível um posicionamento adequado na pista para realizar a decolagem até uma altitude pré-estabelecida. Após atingir tal altitude, o agente inteligente estabiliza a aeronave.

Foi elaborada a conexão entre os aplicativos Jason e X-Plane, de forma que a partir da classe Java utilizada como ambiente fosse possível acessar os dados do simulador,

obtendo informações como velocidade, latitude, longitude, altura e as angulações dos vetores do plano cartesiano transmitindo-as como crenças para o agente.

Para que seja possivel realizar uma integração com o ambiente utilizado pelo Jason é necessário instanciar a classe *XPlaneJasonComm*, que é responsável pela comunicação com o simulador através da interpretação dos pacotes UDP, dentro da classe do ambiente do agente. A classe *XPlaneJasonComm* foi adaptada para ser uma *Thread*, para que sempre que forem requisitadas as informações do simulador, esta esteja com os dados mais recentes. Nessa mesma classe são identificadas quais são as informações que serão transmitidas do simulador ao ambiente como percepções. A figura 2 exibe o código relativo ao ambiente do agente.

```
public class XPlaneEnv extends Environment {
    private float lat, lon, alt, speed, vY, joy_elev, joy_ailrn, joy_ruddr, joy_vect;
    private XPlaneJasonComm xData;

    public XPlaneEnv(){
        super();
        xData = new XPlaneJasonComm();
        xData.start();
    }

    public void init(String[] args){
        try {
            updatePercept();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

Figure 2. O ambiente do agente com a integração a API de comunicação UDP.

Além da modificação no ambiente do agente e na classe *XPlaneJasonComm*, é necessário configurar o simulador identificando o IP da máquina que hospeda o agente e quais as informações estão habilitadas para serem transferidas pelo protocolo UDP. O simulador permite a conexão de diversas máquinas com diferentes IPs ao mesmo tempo, possibilitando voos em missão conjunta. As informações que representam as crenças são criadas como atributos na classe do ambiente do agente e são atualizadas pelo simulador de voo. Em seguida o método *updatePercept* adiciona as informações como crenças ao agente especificado. O codigo referente ao método pode ser visto na figura 3.

```
public void updatePercept() throws InterruptedException {
    clearPercepts("plane");
    this.lat = xData.getLat();
    this.lon = xData.getLon();
    this.alt = xData.getAlt();
    this.speed = xData.getSpeed();
    this.vY = xData.getvY();
    this.vX = xData.getvX();
    this.joy_elev = xData.getElev();
    this.joy_ailrn = xData.getAilrn();
    this.joy_ruddr = xData.getRuddr();
    this.joy_vect = xData.getVect();
    addPercept("plane",Literal.parseLiteral("lat("+this.lat+")"));
    addPercept("plane",Literal.parseLiteral("lon("+this.lon+")"));
    addPercept("plane",Literal.parseLiteral("alt("+this.alt+")"));
}
```

Figure 3. As crenças do agente baseadas nas informações do simulador de voo.

Com isso, a integração é possível ser executada através da programação das ações externas do agente no ambiente do Jason e em seguida programar a ação relativa que o agente deseja realizar no simulador. Para isso é necessário invocar o método *setValue* da classe *XPlaneJasonComm* que enviará, através pacotes UDP, a informação e o valor a ser modificado no simulador.

Inicialmente o avião começa acelerando exponencialmente mantendo as coordenadas de longitude de forma a permanecer na pista, ao aproximar-se do fim da pista é levantado voo em uma angulação de 20° e mantendo até atingir 1500 pés de altitude. Dada esta altura, a aeronave estabiliza-se com o plano horizontal.

As intenções iniciais do agente são *getInitialPosition* e *startMission*. A intenção *startMission* só é executada com a condição da crença *missionStarted* não estiver presente na base de crenças do agente. Após serem obtidas a crenças da posição inicial é dado o início a missão soltando os freios e acelerando a aeronave como se pode observar na figura 4.

```
/* Initial goals */
!getInitialPosition.
!startMission.

/* Plans */
+!getInitialPosition: not missionStarted <-
    .print("Pegando a Posição Inicial.");
    getPosition.

+!startMission: not missionStared <-
    +missionStarted;
    .print("Missão Iniciada.");
    throttle;
    .print("Acelerando.");
    !atPosition.
```

Figure 4. Objetivos iniciais do agente.

A partir de então é mantida a ação *throttle*, acelerando a aeronave e movendo-se somente na latitude para que a aeronave não saia da pista. A ação *getPosition* atualiza as crenças do agente com a posição atual da aeronave no simulador. Quando é percebida a latitude que representa a proximidade do final da pista, é acionado o plano *flying* que atua alterando o eixo do avião ao plano horizontal em 20° e mantendo a aceleração. Já o plano *!atPosition* verifica se o ponto de decolagem foi atingido, enquanto isso o agente continua acelerando. A figura 5 mostra os planos *!atPosition* do agente.

```
!atPosition: missionStarted & lat(Y) & lon(X) & Y <= 41.334 <-
   getPosition;
   centralize;
   !atPosition.

!atPosition: missionStarted & lat(Y) & lon(X) & Y > 41.334 <-
   .print("Levantando Voo.");
   !flying.

!atPosition: true <-
   !!atPosition.
```

Figure 5. O plano atPosition.

O plano *flying* é responsável por verificar se a aeronave chegou ao ponto de se estabilizar. Para isso o plano *flying* analisa se a posição da aeronave está entre 1000 e 1500 pés e enquanto essa valoração não for ultrapassada, o plano continua executando e atualizando a posição para o agente. Após atingir a altura desejada, o plano *stabilizing* é acionado com o objetivo de estabilizar a aeronave, centralizar e acionar o piloto automático da aeronave. A figura 6 exibe o plano *flying* e *stabilizing*.

```
+!flying: missionStarted & alt(A) & A < 1000<-
    .print("Subindo até 1500. Altitude atual:", A);
    getPosition;
    getup;
    !flying.

+!flying: missionStarted & alt(A) & A >= 1000 & A < 1500 <-
    .print("Valor atual:", A);
    getPosition;
    !flying.

+!flying: missionStarted & alt(A) & A >= 1500 <-
    !stabilizing.

+!stabilizing: missionStarted  <-
    .print("Piloto Automático Ativado.");
    centralize;
    stabilize;
    .wait(60000).
```

Figure 6. Os planos flying e stabilizing

3.1. Resultados

Esta seção apresenta os resultados obtidos na simulação do voo realizada pelo agente *plane* no simulador X-Plane. O agente pode controlar adequadamente a aeronave na maioria das simulações. Em algumas simulações a aeronave não seguiu um percurso retilínio na decolagem e em outras simulações a aeronave inclinava-se horizontalmente e não retornava a posição de estabilidade. Tais comportamentos se deram devido a simplicidade do raciocínio do agente e a alta realidade do simulador, que provê condições climáticas aleatórias que podem interferir no controle de voo. Contudo, uma codificação de planos e o acesso a outras informações do simulador como crenças do agente permitem um melhor controle da aeronave. A figura 7 exibe o agente *Plane* controlando o simulador de voo.

Foi fixado um aeroporto para que o agente pudesse realizar a decolagem, visto que era necessário determinar um ponto limite para que a aeronave não saísse da pista. Dessa forma, também é necessário a criação de planos para que o agente possa decolar de qualquer aeroporto sem uma prévia configuração.

A integração do Jason com o simulador X-Plane ocorreu conforme o esperado, mantendo uma comunicação através de pacotes UDP sem interferências e satisfatória para o raciocínio do agente. Sobre a performance e o controle do simulador, o ciclo de racioncínio do agente é mais rápido do que o tempo de chegada das informações vindo do simulador, facilitando o processamento das informações pelo agente. Contudo, o agente contém planos simples, dessa forma, testes com planos mais complexos devem ser realizados. As ações realizadas pelo agente são executadas com diferenças mínimas entre a deliberação do agente e a efetiva execução da ação no simulador, não comprometendo a

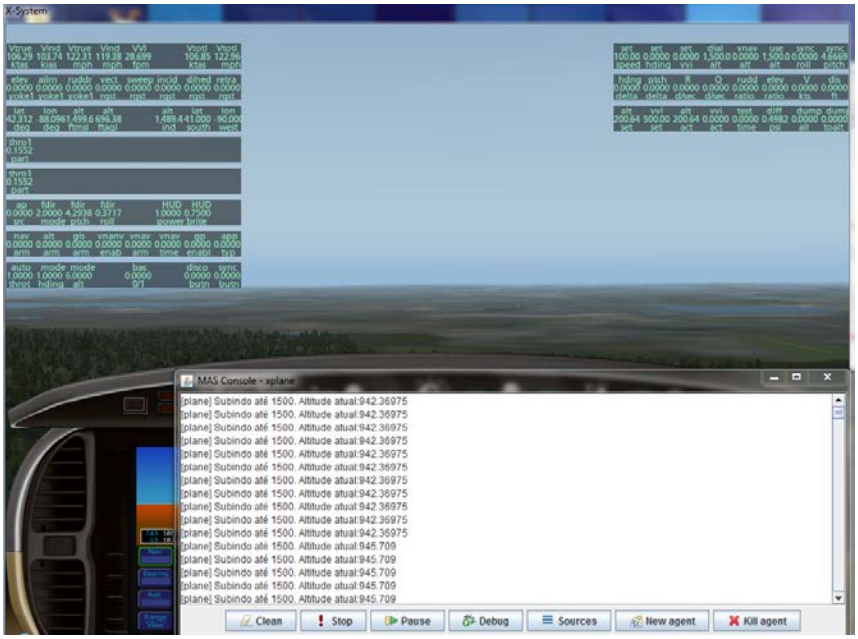


Figure 7. O agente controlando o simulador de voo.

ação do agente. Contudo, nenhuma simulação que exigisse um tempo de resposta quase que instântanea foi realizado.

4. Trabalhos Relacionados

Esta seção apresenta brevemente alguns trabalhos relacionados que utilizam linguagens de programação a agentes em simulações como [Wallis et al. 2002], que apresenta um conjunto de comportamentos táticos em um ambiente simulado de voo. Contudo o trabalho foi desenvolvido utilizando JACK, que é uma linguagem orientada agentes proprietária e reativa. Em [Huff et al. 2003] foi desenvolvido um ambiente de simulação para diferentes abordagens de voo usando o Java em conjunto com outras arquiteturas de softwares. Ambos os trabalhos não utilizam um simulador de voo profissional, sendo desenvolvidas rotinas próprias de simulação de voos, o que impossibilita uma análise para testes em voos em protótipos e ambientes reais.

Este trabalho apresenta um agente inteligente usando o Jason, que é uma linguagem orientada a agentes gratuíta e cognitiva que utiliza a aquitetura BDI. A arquitetura BDI permite que o agente seja programado baseados em crenças, desejos e intenções com a utilização de planos e ações aproximando a programação de ações cognitivas humanas. O trabalho integra o ambiente simulado do agente em Jason a um simulador de voo profissional utilizado para treinamentos que contabiliza hora de voos a pilotos. A utilização do simulador de voo garante que as ações realizadas pelo agente são seguras em um ambiente real de execução.

5. Conclusão

Este artigo apresentou uma integração entre um simulador de alta realidade chamado X-Plane e *framework* orientado a agentes Jason através de uma adaptação de uma biblioteca para comunicação através de pacotes UDP. Foi apresentado também um agente inteligente

que é capaz de realizar uma decolagem usando suas percepções de seu posicionamento global atual e, após atingir determinada altitude, estabilizar o voo e ativar o piloto automático da aeronave.

A integração permitiu que o controle da aeronave através do envio e recebimento dos comandos e informações através de pactoes UDP. O pacote em Java de comunicação com o simulador foi integrado ao ambiente do agente, que por vez passa as informações ao agente inteligente codificado em Jason. As simulações em condições climáticas normais e levando em consideração apenas as informações passadas ao agente foi satisfatória.

Como trabalhos futuros, é necessário ampliar os planos do agente para que ele possa ter um maior controle da aeronave em diversas situações. Também é possivel integrar outras aeronaves para atuarem em rede no mesmo simulador, permitindo a realização de missões em conjunto ou realizar voos em formação, integrando os agentes a alguma plataforma organizacional como o Moise+.

References

Bordini, R. H., Hubner, J. F., and Wooldridge, W. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Jonh Wiley and Sons, London.

Bratman, M. (1987). *Intentions, Plans, and Practical Reason*. Harvard University Press.

Cantoni, L. (2010). Avaliação do uso da linguagem pddl no planejamento de missões para robôs aéreos. Dissertação, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil.

Huff, N., Kamel, A., and Nygard, K. (2003). An agent based framework for modeling uavs. In *Computer Applications in Industry and Engineering*, page 139–144. Springer-Verlag.

Wallis, P., Ronnquist, R., and Lucas, A. (2002). The automated wingman: Using jack intelligent agents for unmanned autonomous vehicles. In *Aerospace Conference*, page 2615–2622. IEEE.

Winikoff, M. (2005). Jack intelligent agents: An industrial strength platform. In Bordini, R., Dastani, M., Dix, J., Fallah Seghrouchni, A., and Weiss, G., editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer US.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons.

Institutional Situatedness in Multi-Agent Systems

Maiquel de Brito[1], **Jomi Fred Hübner**[1]

[1]PPGEAS – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

maiquel.b@posgrad.ufsc.br, jomi.hubner@ufsc.br

Abstract. *Institutions regulating multi-agent systems are not autonomous. Their state changes as the result of facts occurring in the environment where agents act. The modelling of institutional dynamics allowing the institutional regulation based on environmental facts is not a trivial matter. Different institutional abstractions represent different social aspects, having different natures, semantics, life cycles, etc. This work reviews the institutional situatedness analysing how it is currently addressed. We propose a classification for the existing approaches and point some open issues to be tackled in such subject.*

1. Introduction

Institutional abstractions such as norms, roles, goals, missions, scenes, etc, have been proposed to conciliate the goals of the system and the issues raised from system openness and agents autonomy [Artikis et al. 2002, Boissier et al. 2007, Piunti 2009, Esteva et al. 2004]. A regulated MAS may be viewed not just as a set of agents acting in the environment. Besides, a set of mechanisms implementing institutional abstractions composes the institutional dimension (or simply the *institution*) of the multi-agent system (MAS), as shown in the Figure 1.

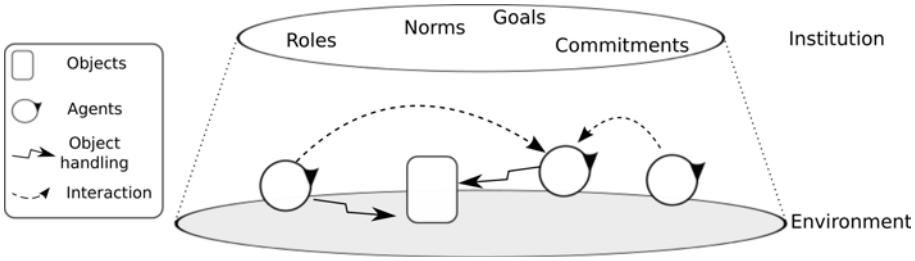


Figure 1. The institution regulates the agents acting in the environment.

Different from the agents, the institution is not autonomous and its state does not change spontaneously. Rather, its state changes as result of facts occurring in the environment, composed of agents and other objects [Searle 1995, Searle 2009]. We consider an institution as *situated* in the environment when its regulation tasks are performed based on environmental facts so that it does not depend on agents informing norm violations, goal achievements, role adoptions, etc. We refer as *situatedness* to the property of an institution being situated in the environment. While specifying the institution requires models and languages to represent institutional abstractions, specifying the institutional situatedness requires models and languages to represent how environmental facts affect the different components of the institution.

Some works have proposed models and languages to specify institutional situatedness. But, as far as we know, there is not any work explicitly reviewing such proposals, comparing them and pointing open issues in the field. To fill this gap, this work analyses how current works deal with situatedness. We are not interested in understanding how specific institutional abstractions are situated. Rather, we aim to have a wide view of the subject, understanding how it is addressed regardless institutional abstractions that are situated.

The main contributions of this analysis are (i) a classification for the current approaches in which other approaches (current or further) can fit (ii) a better understanding about different approaches, according to the proposed classification and (iii) some directions about open issues in this subject. The rest of this paper is organized as follows: Section 2 describes some relevant points to understand the situatedness problem, Section 3 describes how the problem is currently addressed, Section 4 discusses the proposed solutions and, finally, Section 5 points some final remarks about the work.

2. Background

In MAS, several abstractions are proposed to represent social aspects needed to regulate and direct the autonomous acting of the agents towards the achievement of system goals. Some examples of these abstractions are norms, goals, roles, missions, interaction scenes, etc. Each abstraction has a specific nature, representing a specific social aspect of MAS. For example, norms represent what the agents must achieve or avoid, roles represent coherent behaviour expected from an agent, missions represent sets of coherent goals to be achieved by an agent, etc [1]. These abstractions compose the institution that regulates the system while agents act in the environment.

Institutional abstractions, in general, do not represent how the environment, where agents effectively act, influences the regulating tasks. As consequence, implementations of the abstractions do not consider the institution evolving as result of facts external to it. Some of them leave to some external element (including the agents) the responsibility of informing norms violations, role adoptions, goal achievements, etc through institutional interfaces [Gutknecht and Ferber 2001, Campos et al. 2009, Hübner et al. 2006, Hübner et al. 2009]. This is illustrated in the Figure 2: agents act in the environment and use an interface to handle the institutional platform, adopting roles and committing to missions[2].

There are several issues related to the approach of agents handling the institutional platform [Brito et al. 2013]. First, agents must be aware of institutional abstractions and about its implementation. For example, to adopt a role, an agent must be aware of the semantics of the *role* concept and must also know how a particular platform implements the role adoption. Besides, agents can avoid institutional consequences of their actions. For example, it is reasonable that an agent running through a red traffic light does not inform the institution about the norm violation. Thus, it is worth that institutions are situated in the environment, performing the institutional tasks based on environmental

[1] A comprehensive and well organized analysis of institutional abstractions can be found in [Coutinho et al. 2009]

[2] *adoptRole()* and *commitMission()* are primitives provided by interfaces to handle institutions according to the $\mathcal{M}$oise model. Descriptions of these interfaces are found in [Hübner et al. 2006, Hübner et al. 2009].

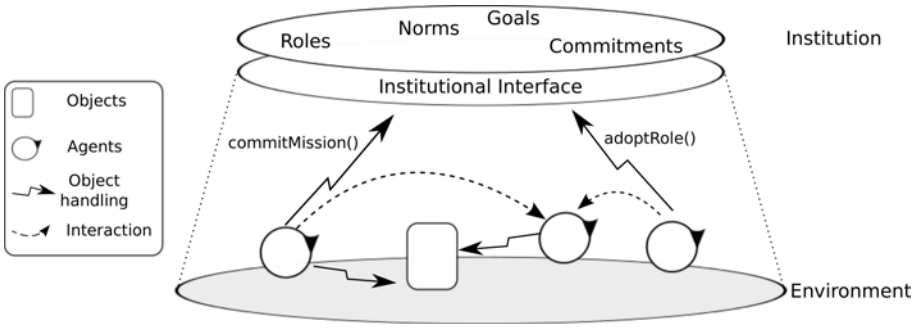


Figure 2. Agents act in the environment and handle the institutional platform.

	Ontological	Functional	
		Concept	Interface
[Dastani et al. 2013]		✓	
[Dastani et al. 2008]		✓	
[Campos et al. 2009]			✓
[Piunti et al. 2010]			✓
[Brito et al. 2013]			✓
[Aldewereld et al.]	✓		

Table 1. Approaches for situatedness

facts rather than on agents handling [Campos et al. 2009].

In human societies, the relation between concrete facts and the institutional state raises naturally and is analysed by the social sciences [Searle 1995]. But in MAS, which can be considered artificial societies [Castelfranchi 2000], such relation is not natural and must be modelled, defining how environmental facts affect the different components of the institution. This modelling, i.e. the conception of institutional situatedness, is not a trivial matter: different institutional abstractions represent different social aspects, having different natures, semantics, life cycles, etc. These differences must be taken into account in situatedness models. The next section describes some works that face this issue.

3. Approaches for situatedness

Some works have addressed situatedness in MAS. They are usually inspired in the *count as* theory of John Searle [Searle 1995, Searle 2009]. According to that theory, human institutions are based on constitutive rules stating that concrete elements *count as* institutional ones. For example, a piece of paper *counts as* a five dollar bill, the eldest son of the deceased king *counts as* the new king, etc.

A contribution of this paper is a classification of the approaches for situatedness. This classification is proposed based on the analysis of some related works. It divides the works into two groups: ontological, described in the Section 3.1 and functional, described in the Section 3.2. The Table 1 summarises the related works according to their approaches.

3.1. Ontological approach for situatedness

Situatedness can be viewed as an ontological matter. Once institutions are specified through abstract concepts, ontological situatedness relates such concepts to concrete elements from the environment. For instance, when a norm states that "authors are forbidden to submit papers having over than 15 pages", ontological situatedness defines what is an author, what is a submission, how to detect the number of pages of the paper, etc.

In the literature, the work of [Aldewereld et al.], in line with [Grossi et al. 2006a, Grossi et al. 2006b, Aldewereld et al. 2009] proposes situatedness according to the ontological approach. The work deals with situatedness of norms and proposes a way to define the meaning of abstract concepts used in norm specifications. This is done through rules in a production system. If a norm states that a is obliged to do b, the rules define that j counts as a and k counts as b, where j and k are environmental elements.

3.2. Functional approach for situatedness

The different institutional abstractions have their specific life cycles. By *life cycle* we mean the set of states where an instance of an institutional abstraction can be found and how the abstraction changes its state. A norm, for example, can be *active*, *fulfilled*, *violated* and *disabled*. A possible life cycle for a norm is *active* and then *fulfilled*. The functional approach for *situatedness* specifies how the life cycle of institutional abstractions evolves as result of facts occurring in the environment.

To exemplify the functional approach and state a clear difference from the ontological one we can again use the norm "authors are forbidden to submit papers having over than 15 pages". While the ontological approach allows to specify what is an author, the functional approach specifies what must happen in the environment so that the norm is active, violated, satisfied, etc.

There are two kinds of functional approaches for situatedness, that we refer as *concept* situatedness and *interface* situatedness.

3.2.1. Concept situatedness

As previously described, different institutional abstractions have specific life cycles. For instance, (i) a *role* may be adopted and then leaved; (ii) a norm may become active and then satisfied or violated; (iii) a goal may be pending and then fulfilled. Taking these differences into account, concept situatedness defines how the life cycle of *specific* abstractions, which represent institutional concepts, are affected by facts from the environment[3].

The proposal of [Dastani et al. 2013] is an example of concept situatedness. The situated abstraction is *commitment*. The work deals with the changes in the state of commitments (according to the life cycle shown the Figure 3) as consequence of environmental facts. The code excerpt (cex1) below shows an example of specification according to such approach. The line 1 states that an agent x proposing to an agent y to achieve q before the instant d_2 if y achieve p before d_1 counts as a new commitment whose initial state is *conditional* (C^{c}). The line 2 states that, from the commitment defined in the line 1, the

[3] An institutional abstraction represents an institutional concept.

agent x informing to y that has done q before d_1 counts as satisfying of the commitment, whose state changes from *conditional* to *satisfied* (C^s).

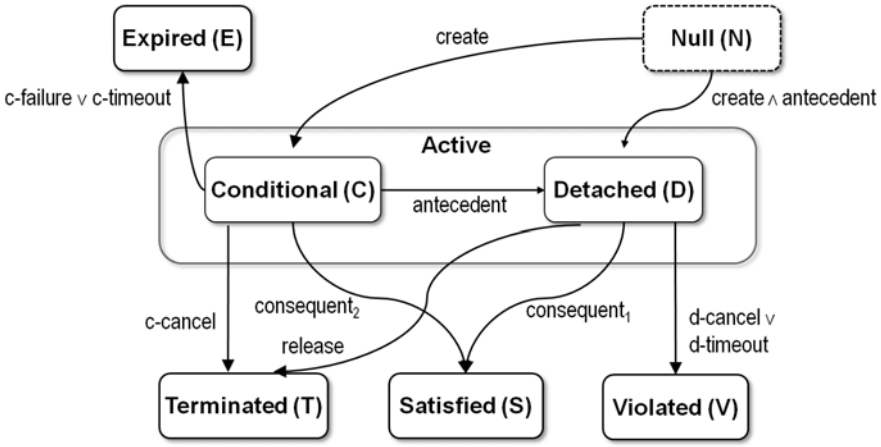


Figure 3. Life cycle of a commitment [Dastani et al. 2013]

$$
\begin{aligned}
1:&\ \ offer(x,y,p,q,d_1,d_2) \Longrightarrow_{cr} \mathsf{C}^c(x,y,p,q,d_1,d_2)\\
2:&\ \ tell(x,y,q) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d1 \wedge q \Longrightarrow_{cr} \mathsf{C}^s(x,y,p,q,d_1,d_2)
\end{aligned}
\qquad \text{(cex1)}
$$

Concept situatedness is also the approach in the set of works related to the 2OPL model, composed by [Dastani et al. 2009, Dastani et al. 2008, Tinnemeier et al. 2009, Tinnemeier et al. a, Tinnemeier et al. b]. In this case, the abstraction considered is *norm*. The works propose a way to represent how the life cycle of a norm, illustrated in Figure 4, is affected by the environmental state. The code excerpt (cex2) below shows a rule according to this approach. The rule defines that if the environment has properties pointing that a paper having more that 15 pages has been submitted to a conference, then there is a norm violation.

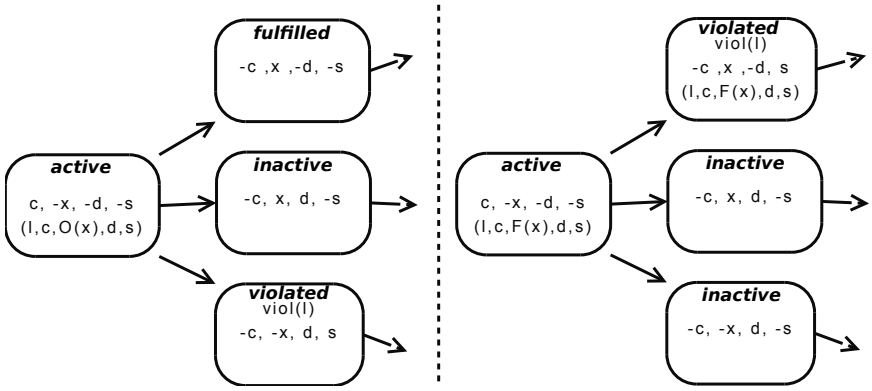


Figure 4. Life cycle of obligations (left) and prohibitions (right) (adapted from [Tinnemeier et al. 2009])

.

$$received(As) \wedge member((A,Id),As) \wedge pages(Id) > 15 \Longrightarrow viol\_size(A) \qquad \text{(cex2)}$$

3.2.2. Interface situatedness

Different from concept situatedness, interface situatedness is not concerned about which are the institutional abstractions affected by environmental facts. Institutional concepts are not included in the models. At a glance, in this approach, a situatedness interface observes the environment and, by interpreting the situatedness specification, produces informations about *what should happen* in the institution. These informations do not have, themselves, institutional meaning. It is assumed that the institution takes such informations and changes its own state accordingly (Figure 5).

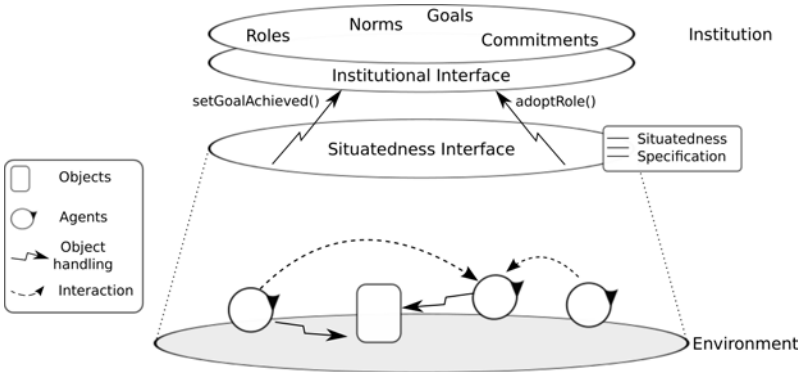


Figure 5. The situatedness interface observes the environment and informs the institution when necessary.

The work of Piunti et al. [Piunti 2009, Piunti et al. 2010] deals with situatedness of the $\mathcal{M}$oise organisational model. A language is proposed to specify that events from the environment trigger operations in ORA4MAS artifacts, that implement the organizational model [Hübner et al. 2009, Ricci et al. 2011]. The code excerpt cex3, extracted from [Piunti et al. 2010], shows a specification according to this approach. It specifies that the event *pay*, produced by the environmental object *BillingMachine*, triggers the operation *setGoalAchieved* in the artifact named *visitSchBoard*. In the application scenario, the rule states that when a billing machine produces the event *toPay*, the organisational goal *pay_visit* is achieved.

$$+\texttt{op\_completed}(\text{"}\texttt{BillingMachine}\text{"}, \texttt{Ag}, \texttt{pay}) \\ -> \texttt{apply}(\text{"}\texttt{visitorSchBoard}\text{"}, \texttt{setGoalAchieved}(\texttt{Ag}, \texttt{pay\_visit})). \qquad \text{(cex3)}$$

Interface situatedness is also the approach in the work of Brito et al. [Brito 2012, Brito et al. 2013], where a programming language is proposed to specify properties that the institution should have as consequence of both events or states of the environment. The code excerpt cex4, extracted from [Brito et al. 2013], is an example of this approach. It states that if the environmental object *Art* has the property *auctionStatus(closed)*, then the institution should have the property *play(Winner,Role,g1)*. In the application scenario, the property *play(Winner,Role,g1)* holds when the agent *Winner* plays the role *Role* in the group *g1*.

$$
\begin{array}{l}
* \ \texttt{auctionStatus(closed)[source(Art)]} \\
\quad \textbf{count} - \textbf{as} \ \texttt{play(Winner, Role, g1)[source(g1)]} \\
\quad \textbf{in} \ \texttt{currentWinner(Winner)[source(Art)]} \ \ \& \\
\qquad \texttt{not(Winner} == \texttt{no\_winner)} \ \ \& \ \ \texttt{auction\_role(Art, Role).}
\end{array}
\tag{cex4}
$$

A third case of interface situatedness is the *Situated Electronic Institutions* model (SEI) [Campos et al. 2009], that provides situatedness to *Electronic Institutions* (EI) [Esteva et al. 2001]. In SEI, special agents named *modeller* and *staff agents* monitor the environment checking facts that may be relevant to the institution. Those facts are informed to special agents named *governors* that are aware about institutional specification and determine the institutional meaning of the facts.

4. Discussion

The Section 3 presented some works related to institutional situatedness proposing a classification to them. While that section describes how different approaches propose to situate an institution, this section discusses two important points observed in the analysed works. The first point, discussed in the Section 4.1 is the comprehensiveness of the approaches regarding to institutional aspects that can be situated. The second point, discussed in the Section 4.2, is the institutional semantic of specifications following the different approaches.

4.1. Institutional coverage

In concept situatedness, each model situates a specific abstraction (Figure 6). This implies a limited coverage to the institutional abstractions that can be situated. The analysed works propose conceptual situatedness for commitments and norms. The proposed models cannot be used to situate additional abstractions. Different concepts require different models and languages. Besides, the same institutional abstraction may have many different conceptions. For example, the life cycle of commitments in [Dastani et al. 2013] is different of the one proposed in [Fornara and Colombetti 2006] e [Chesani et al. 2013]. Thus, different conceptions for the same abstraction require different models of concept situatedness. To provide conceptual situatedness for all institutional abstractions possibly present in an MAS, it would be necessary to provide languages for every abstraction. Admitting that some institutional aspects of MAS may have not been identified yet, new institutional abstractions may be still proposed and to situate them, it is needed additional work to propose models and languages.

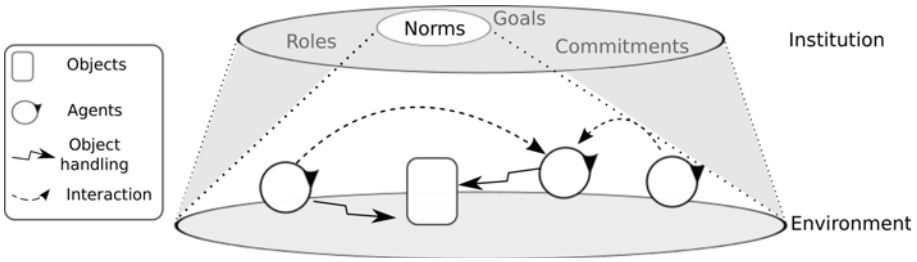


Figure 6. Concept situatedness: an approach situates just one abstraction.

Interface situatedness and ontological situatedness do not have the issue of limited institutional coverage. But it does not mean completeness regarding to the situated institutional abstractions. Rather, such approaches do not take into account specific institutional abstractions. Although the approaches consider that the environmental facts affect the institution, it is not taken into account whether they are affecting norms or roles or goals or anything else. While this feature provides a wider institutional coverage, it raises some issues related to the institutional semantics of the models. These issues are discussed in the next section.

4.2. Institutional semantics

Conceptual situatedness is concerned with the environment affecting the life cycle of specific abstractions. The semantics of the abstractions is taken into account in the semantics of languages for situatedness specification. For example, the code excerpt cex1 shows the specification of institutional effects for messages exchanged by the agents. Such specification uses the syntactic elements C^{c} e C^{s}, that have institutional semantics. The rule shown in the Figure 7 belongs to the operational semantics of the approach and states that from a communication act $com(\alpha)$, the institutional state of the system is updated according to the operator $\oplus$.

$$\frac{com(\alpha) \wedge \sigma_i' = \oplus(\sigma_i \cup \sigma_b \cup \{\alpha\})}{\langle \sigma_b, \sigma_i \rangle \rightarrow \langle \sigma_b, \sigma_i' \rangle}$$

$$where \quad \oplus\,(\sigma_i \cup \sigma_b \cup offer(x, y, p, q, d_1, d_2)) = \sigma_i \cup \mathsf{C}^{\mathsf{c}}(x, y, p, q, p1, d2)$$

Figure 7. Semantic rule for commitment situatedness [Dastani et al. 2013].

Interface situatedness, on its turn, does not make any assumption about specific institutional abstractions. Institutional concepts are not part of the models and related languages do not use institutional primitives. As a consequence, languages for interface situatedness cannot express the institutional meaning of environmental facts. The languages define how the facts from the environment are handled to produce some information that is provided to the institution. But this information does not have itself institutional meaning. The institution is in charge of handling that information, updating its institutional abstractions accordingly. That is, the institutional meaning of the produced information is given by the very institution instead of being given by the situatedness model. For example, (i) in the model of [Piunti 2009, Piunti et al. 2010]., is specified that events from the environment trigger operations in artifacts and (ii) in the model of Brito et al., is specified that the institution should have some properties as consequence of events and states from the environment. In both cases, however, the institutional meaning of the environmental facts is placed into the designer's mind rather than be placed in the language semantics. For example, the code excerpt cex3, from [Piunti et al. 2010], specifies that the event *pay*, produced by an environmental object named *BillingMachine*, triggers the operation *setGoalAchieved* in an artifact named *visitorSchBoard*. In its specific application, the rule means that the institutional consequence of the event *pay* produced by an electronic billing machine is the achievement of the institutional goal *pay_visit*. But such institutional meaning is not explicit in the semantics of the language. The semantics just

$$\frac{\langle x, y, c\rangle \in R_s \quad \mathcal{N} \cup \mathcal{I} \cup D \models x \quad \mathcal{N} \cup \mathcal{I} \cup D \models c \quad y \notin \mathcal{I}}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \cup y\rangle}$$

Figure 8. Semantic rule for count-as rule evaluation [Brito et al. 2013].

states that the operation *apply*, with two parameters, is triggered in an artifact. The institutional semantics, on its turn, is defined in the artifact operation: *setGoalAchieved(a,g)* means that the goal *g* has been achieved by the agent *a* [Hübner et al. 2009].

The approach of [Brito et al. 2013, Brito 2012] has a similar issue. The code excerpt cex4 states that when the environment has the property *auctionStatus(Closed)*, then institution should have the property *play(Winner,Role,g1)*. But the property *play(Winner,Role,g1)* does not have, itself, any institutional meaning. In its specific application, the property is truth when the agent *Winner* plays the role *Role* in the group *g1*. But the institutional platform is in charge of interpreting the property giving it an institutional meaning. Notice that *role*, that is an institutional concept, does not even belong to the situatedness model. We can clearly see the lack of institutional semantics in this proposal through the operational semantics of the language. The semantic rule shown in the Figure 8 states that when a rule $\langle x, y, c\rangle$ is evaluated, the element y is added to a queue $\mathcal{T}$ to be consumed by the institutional platform. The element y does not have institutional meaning.

Although SEI model does not have a specification language, this approach for situatedness also lacks of institutional semantics. Special agents observe the environment and just inform the *governors* about relevant facts. The *governors* are in charge of give institutional semantics for that information.

Finally, the approach of [Aldewereld et al.], that addresses situatedness as an ontological problem, also lacks of institutional semantics. Resuming a previous example, for a norm stating that "*a* is obliged to *b*", the implementation of ontological situatedness allows to specify that "*j* counts as *a*" and "*k* counts as *b*". But it allows also to specify that "*j* counts as *b*" and "*k* counts as *a*", that is wrong as *a* is an agent while *b* has a different nature. This is possible because the environmental elements are related to the norm elements but are not related to institutional concepts. For example, the rule states that "*j* counts as *a*" but do not takes into account that *j* is an agent. The designer is in charge of to consider this semantics when writes the rules.

The use of institutional primitives to situatedness programming, as allowed by the concept situatedness, is an advantage as the programs have an explicit institutional meaning. This allows the agents (both human and artificial) to reason about the institutional effects of environmental facts and of their own actions. For example, observing the code excerpt cex1, through the semantics of Figure 7, an agent knows that when it utters an *offer*, it produces a commitment. On other hand, when situatedness language lacks of institutional primitives, agents may not be able to use the situatedness program to reason about the institutional effects of environmental facts. By observing the code excerpt cex3, for example, an agent knows that when the object *BillingMachine* produces the event *pay*, the operation *setGoalAchieved(Ag,pay_visit)* in the artifact *visitorSchBoard*. But to know that such operation means the achievement of the goal *pay_visit*, the agent should to know

also the ORA4MAS infrastructure. Thus, in this case, it is not sufficient to the agents to reason about the situatedness specification to know how to achieve the goal *pay_visit*.

As another issue, the lack of institutional semantics in situatedness languages, observed in both ontological and interface situatedness, allows specifications that do not produce any institutional effect. For example, although the replacement of the code excerpt cex3 by the code excerpt below(cex5) keeps the program semantic and syntactically correct, it does not specify any institutional consequence for the environmental fact. Concept situatedness, on its turn, induces a consistent programming as the consequences of environmental facts are defined in terms of institutional concepts.

$$+\texttt{op\_completed}(\text{"}\texttt{BillingMachine}\text{"}, \texttt{Ag}, \texttt{pay}) \\ -> \texttt{apply}(\text{"}\texttt{visitorSchBoard}\text{"}, \texttt{meaninglessParam}(\texttt{Ag}, \texttt{pay\_visit})). \quad \text{(cex5)}$$

5. Final remarks

This work analysed institutional situatedness in MAS. Current approaches can be divided in two approaches: ontological and functional. We observed that (i) some approaches have a limited coverage regarding to situated institutional abstractions, (ii) some approaches lack of institutional semantics and (ii) there is no approach that situates all institutional abstractions and allows situatedness programming using institutional primitives.

Acknowledgements

The authors are grateful for the support given by CAPES and CNPq (grant number 140261/2013-3).

References

Aldewereld, H., Álvarez-Napagao, S., Dignum, F., and Vázquez-Salceda, J. Making norms concrete. In van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M., and Sen, S., editors, *Proc. 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 807–814.

Aldewereld, H., Alvarez-Napagao, S., Dignum, F., and Vázquez-Salceda, J. (2009). Engineering Social Reality with Inheritance Relations. In *Proc. 10th International Workshop on Engineering Societies in the Agents World X*.

Artikis, A., Pitt, J., and Sergot, M. (2002). Animated specifications of computational societies. In *Proc. 1st international joint conference on Autonomous agents and multiagent systems*.

Boissier, O., Hübner, J. F., and Sichman, J. S. (2007). Organization Oriented Programming: From Closed to Open Organizations. In O'Hare, G. M. P., Ricci, A., O'Grady, M. J., and Dikenelli, O., editors, *Engineering Societies in the Agents World VII*, volume 4457 of *LNCS*, pages 86–105. Springer Berlin Heidelberg.

Brito, M. (2012). Uma linguagem para especificação da dinâmica dos fatos institucionais em sistemas multiagentes. Master's thesis, Universidade Federal de Santa Catarina,

Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Brito, M., Hübner, J. F., and Bordini, R. H. (2013). Programming institutional facts in multi-agent systems. In Aldewereld, H. and Sichman, J. S. a., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VIII*, volume 7756 of *LNCS*, pages 158–173. Springer Berlin Heidelberg.

Campos, J., López-Sánchez, M., Rodríguez-Aguilar, J., and Esteva, M. (2009). Formalising Situatedness and Adaptation in Electronic Institutions. In Hübner, J., Matson, E., Boissier, O., and Dignum, V., editors, *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, volume 5428 of *LNCS*, pages 126–139. Springer Berlin / Heidelberg.

Castelfranchi, C. (2000). Engineering social order. 1972:1–18.

Chesani, F., Mello, P., Montali, M., and Torroni, P. (2013). Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems*, 27(1):85–130.

Coutinho, L. R., Sichman, J. S., and Boissier, O. (2009). *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, chapter Modelling Dimensions for Agent Organizations. Information Science Reference.

Dastani, M., Grossi, D., Meyer, J.-J. C., and Tinnemeier, N. (2009). Knowledge Representation for Agents and Multi-Agent Systems. chapter Normative Multi-agent Programs and Their Logics, pages 16–31. Springer-Verlag, Berlin, Heidelberg.

Dastani, M., Tinnemeier, N., and Meyer, J.-J. (2008). *A programming language for normative multi-agent systems*, chapter XVI, pages 397–417. Information Science Reference, Hershey, PA, USA.

Dastani, M., Torre, L., and Yorke-Smith, N. (2013). Monitoring interaction in organisations. In Aldewereld, H. and Sichman, J. S., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VIII*, volume 7756 of *LNCS*, pages 17–34. Springer Berlin Heidelberg.

Esteva, M., Rodríguez-Aguilar, J. A., Sierra, C., Garcia, P., and Arcos, J. L. (2001). On the Formal Specifications of Electronic Institutions. In Dignum, F. and Sierra, C., editors, *AgentLink*, volume 1991 of *LNCS*, pages 126–147. Springer.

Esteva, M., Rosell, B., Rodríguez-Aguilar, J. A., and Arcos, J. L. (2004). AMELI: An Agent-Based Middleware for Electronic Institutions. pages 236–243.

Fornara, N. and Colombetti, M. (2006). Specifying and Enforcing Norms in Artificial Institutions. In Dunin-Keplicz, B., Omicini, A., and Padget, J. A., editors, *EUMAS*, volume 223 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Grossi, D., Aldewereld, H., Vázquez-Salceda, J., and Dignum, F. (2006a). Ontological aspects of the implementation of norms in agent-based electronic institutions. *Computational & Mathematical Organization Theory*, 12(2-3):251–275.

Grossi, D., Meyer, J.-J. C., and Dignum, F. (2006b). Counts-as: Classification or Constitution? An Answer Using Modal Logic. In Goble, L. and Meyer, J.-J. C., editors, *DEON*, volume 4048 of *LNCS*, pages 115–130. Springer.

Gutknecht, O. and Ferber, J. (2001). The madkit agent platform architecture. In Wagner, T. and Rana, O. F., editors, *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, volume 1887 of *Lecture Notes in Computer Science*, pages 48–55. Springer Berlin Heidelberg.

Hübner, J. F., Boissier, O., Kitio, R., and Ricci, A. (2009). Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, 20(3):369–400.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2006). $\mathcal{S} - \mathcal{M}$oise$^+$: A middleware for developing organised multi-agent systems. In Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J., and VÃ¡zquez-Salceda, J., editors, *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *Lecture Notes in Computer Science*, pages 64–77. Springer Berlin Heidelberg.

Piunti, M. (2009). *Designing and Programming Organizational Infrastructures for Agents situated in Artifact-based Environments*. PhD thesis, Universit‘ di Bologna.

Piunti, M., Boissier, O., Hübner, J. F., and Ricci, A. (2010). Embodied organizations: a unifying perspective in programming agents, organizations and environments. In *MALLOW*.

Ricci, A., Piunti, M., and Viroli, M. (2011). Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192.

Searle, J. (1995). *The Construction of Social Reality*. Free Press.

Searle, J. (2009). *Making the Social World:The Structure of Human Civilization*. Oxford University Press.

Tinnemeier, N., Dastani, M., and Meyer, J.-J. In *Proc. of 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 121–128, Richland, SC.

Tinnemeier, N., Dastani, M., and Meyer, J.-J. In *Proc. of 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, pages 957–964, Richland, SC.

Tinnemeier, N. A. M., Dastani, M. M., Meyer, J.-J. C., and van der Torre, L. (2009). Programming Normative Artifacts with Declarative Obligations and Prohibitions. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 2:145–152.

Reading minds using classification algorithms on fMRI data

Caroline Froehlich[1] **, Alexandre R. Franco**[2] **, Felipe Meneguzzi**[1]

[1]Faculdade de Informática , [2]Faculdade de Engenharia
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

carol.frohlich@gmail.com, {alexandre.franco,felipe.meneguzzi}@pucrs.com.br

Abstract. *Functional Magnetic Resonance Imaging (fMRI) is a non-invasive method to obtain brain images that indirectly shows neuronal activation. With fMRI scans, we are able to measure areas of the brain that are active in time during extension of the exam, which are often transformed into a time-sequence of images. These images are then analyzed by human experts to infer information of interest. Recent work has used machine learning algorithms to extract more complex information from fMRI scans. In this paper we propose to use a classification based algorithm to differentiate, at each time point during the scan, whether a single patient is performing a task or not. We process the data to generate examples when the patient is performing a task or resting, and experiment different parameters for the classification algorithm to achieve a high success rate.*

1. Introduction

Functional Magnetic Resonance Imaging (fMRI) is an imaging test that indirectly measures neural activity over time. Some neural activity patterns are known to indicate a person's cognitive state or indicate if a patient has a neuropsychological disorder. Cognitive disorders such as autism [Just et al. 2012], alzheimer's disease [Woodard et al. 2009] and dementia [Woodard and Sugarman 2012] are known to have been successfully identified using fMRI data. Moreover, it has been shown that using fMRI, we are able to predict the cognitive state of patients (predict in what they are thinking or seeing) [Mitchell et al. 2003]. Additionally given its non-intrusiveness to patients, fMRI is widely recognized as a powerful diagnostic tool for conditions with a neurological basis. In this context, learning disabilities such as dyslexia are known to have a strong neurological basis [Wolf and Bowers 1999]. However, no standard test used today is able to detect or predict dyslexia precisely [Hoeft et al. 2011]. Thus, new tests are needed in order to fill this gap and our overarching objective is to reach this goal by using fMRI, an imaging test able to indirectly detect neural activity while people are resting or performing cognitive tasks.

There are a number of problems that need to be overcome when analyzing fMRI data to discover such patterns. For instance, there is a very large amount of information to analyze at the same time, since a regular test contains about 30,000 voxels. Furthermore, the difference between an activated portion of the brain and a deactivated portion is often minimal. Thus, there is significant potential for using machine learning algorithms to find complex brain activation patterns. Machine learning can work with a huge amount of portions of the brain and they are sensitive to minimal changes in the fMRI test.

In this paper, we report on our initial efforts to use classification techniques on fMRI data as a diagnostic tool for dyslexia sufferers by calibrating such algorithms to

distinguish when a patient is engaged with a mental task or not. Namely, we use a classification technique to predict whether a single patient is performing a task or resting at any given time during the fMRI scan. Specifically, we train a Support Vector Machine (SVM) learning algorithm with fMRI data from a single patient, in which we have partitioned the training data as time points of the fMRI scan. We show that the resulting classifier has an average of 75% accuracy on the final test partition of the dataset.

This paper is organized as follows. Section 2 reviews the background in fMRI and machine learning, describing the SVM algorithm in detail. Section 3 presents the data we use and how we transform it to be the input of the SVM algorithm. Section 4 describes our model of the experiment with the fMRI data, while Section 5 shows the results of these experiments. Finally, in Section 6 we conclude with final considerations and point the direction of future work.

2. Background

2.1. fMRI

Neuroimaging involves different techniques to acquire images of the brain, each of which have distinct purposes, measuring and showing views of how a subject's brain is or how it works. Each exam has a distinct spatial and temporal resolution, and detect different tissues.

For example, MRI (Magnetic Resonance Imaging) is a neuroimaging test that shows a static view of the brain anatomy in detail. MRI data have a good spatial resolution, showing with millimeter accuracy brain tissue of white matter and gray matter. With the MRI scanner we can acquire a high resolution 3D image of the brain showing these tissues.

We are interested in fMRI (functional Magnetic Resonance Imaging) [Huettel et al. 2004] , that is a 4D functional neuroimage that consists of a time series of 3D images of the brain. While MRI acquires one high resolution image showing the brain structure, fMRI acquires many low resolution images in order to detect the activation of brain regions over time, since its purpose is to map brain activity [Biswal et al. 1995]. Compared to EEG, fMRI data has a high spatial resolution (it shows the brain details in millimeters, as MRI), and a low temporal resolution (because the MRI scanner typically takes a couple of seconds to adquire a single volume).

In MRI, we can observe in high detail how the brain structure is organized. In fMRI, we observe the level of activation of the brain, that means which brain region is activated when performing a specific task and how brain regions work together. For example, we can test which parts of the brain increase their activity when a subject is performing a task while inside the MRI scanner, such as moving the left hand or checking if two words rhyme. When more than one region is activated during a task, fMRI data can indicate how these brain parts interact: if they work synchronously (both regions are activated at the same time), when one is activated, the other deactivate, or if they are not related (the activation of one is not dependent on the other).

While the purpose of fMRI is to detect neural activity during experiments, it does not measure neural activity directly. Instead, it measures the fluctuation of oxygenated and deoxygenated hemoglobin in the blood flow. As oxygenated and deoxy-

genated hemoglobin have different magnetic properties, fMRI can detect changes in the hemoglobin molecule. Local neural activity increases oxygen consumption in cells, that increases local oxygenated hemoglobin level, so fMRI detects neural activity by measuring the change of deoxygenated to oxygenated hemoglobin, called BOLD (Blood Oxygen Level Dependent). Approximately two seconds after neural activity, there is change in the BOLD signal, that lasts 4 seconds, and we can see this changes in neuroimage.

We can obtain much information from fMRI data. First, we can see the brain function when someone is performing a cognitive or motor task, mapping the regions where there is more neural activity. Second, we can identify the cognitive state of a subject in a given time using fMRI data.

We can find brain activation patterns in subjects that are performing the same task by looking for brain regions that are activating or communicating in the same way. Also, we can identify when a single subject is performing a task by repeating the task several times and look for the brain regions that are always activated during the task. These patterns form biomarkers, that in this context are a set of characteristics (brain region activations) that identify a cognitive state.

In this paper, we are concerned with a specific type of fMRI test called task-based fMRI (used as a synonym of fMRI). In the task based-fMRI, the subject is required to perform a cognitive task, such as checking if 2 words rhyme. Stimuli (words) are presented on a screen for a few seconds while the subject is inside the scanner. Then, the subject has a few seconds to answer if the words rhyme, and a few seconds to rest. Each stimulus is repeated a number of times in order to identify clearly the brain regions that the task activates. In this type of experiment, we look for brain areas that are more activated during a trial. But knowing which brain areas are activated during a trial is not enough. Because the brain never ceases activity, data from this experiment shows voxels that are always activated when subjects are not performing any task in particular, and this information is not important to task-based experiments, where we want to see voxels that are activated in a specific task. Therefore, we acquire subjects baseline data, that is when the subject is not performing any task in particular and subtract the baseline data from task-based data to see just voxels involved in the task.

We divide the fMRI test in blocks. A block is a set of trials that a subject performs in row. For example, perform the rhyme task 6 times in row and then rest for a long period. Beside the task block, there are rest blocks, where the subject looks at a screen showing a fixation-cross (a black screen with a white cross in the middle) and not thinking in anything in particular. The baseline data is acquired in this block.

2.2. Support Vector Machine

Machine learning is an area of AI that studies techniques and algorithms to learn unknown functions from data [Russell and Norvig 2010], [Mitchell 1997]. There are many machine learning techniques, each of which is suitable for particular combinations of data and target functions. In this paper, we work with a class of machine learning algorithms called supervised learning. This type of algorithm solves the classification problem, namely, given examples E with labels Y, we want to learn a function that maps the examples to the labels (also called classes). Once this function is learned, it can generalize what it has learned from the previous examples and predict the label of the new example that the

algorithm itself has never seen before.

More specifically, we work with a machine learning algorithm called Support Vector Machine (SVM) [Vapnik 2000]. SVM solves the binary classification problem, which is when each example is classified as being in one of two classes: positive and negative. SVM converts each example into a point in a Cartesian plane, and tries to find a hyperplane that divides the Cartesian plan in two: on one side there should be all positive examples while on the other side there should be all the negative examples. The type of hyperplane that divides the examples is determined by the kernel we use along with the SVM algorithm. A kernel is the type of function that we use to create the hyperplane, for example, a linear kernel creates a linear hyperplane that divides the examples. However, since not all examples can be linearly divided we sometimes need a non-linear kernel.

In this paper, we work with a non-linear kernel called RBF (Radial basis function) kernel [Powell 1987]. An RBF kernel has 2 parameters we can adjust: C and γ. The C parameter determines how the algorithm treats misclassified examples; if the value of C is low, the SVM training algorithm tries to find a smooth hyperplane, even if some examples are misclassified. Conversely, if the value of C is high, the SVM training algorithm tries to find a hyperplane that correctly classifies all examples. The γ parameter determines the level of influence a single example has on the boundary of the resulting hyperplane. Thus, if γ is set to a low value, each example makes the hyperplane that represents the class much bigger, whereas if γ is high, each example does not increase the hyperplane that represents the class so much.

3. Data

The data we used in our experiments was obtained in a series of fMRI exams aimed at evaluating the proficiency in reading and comprehension of subjects diagnosed with dyslexia. Such tests are collectively known as language tests, since their aim is to activate the brain regions associated with language skills. The particular language test performed by the patients during an imaging session involves what is known as a pseudo-word tasks. In this task, a visual stimulus in the form of a written word is presented on a screen and patients have a few seconds to answer whether the word presented is a real word or not. The pseudo-word task aims to activate brain regions related to language processing, such regions are different than the areas activated when patients are resting. Thus, our goal is to differentiate the neural activity pattern when the subject is performing the pseudo-word task and when the subject is resting (i.e. not thinking in anything in particular).

Therefore, the fMRI data generated by this test can be classified according to the stimulus being presented to the subject, namely: pseudo-word task (when a word is being shown to the subject) and rest (when no word is being presented to the subject). In the pseudo-word part, patients have to decided if a certain written word displayed in a screen is real (e.g. dream) or not (e.g. cra), indicating if the word is real by pressing a button. Each time a word is displayed to the subject is called a *trial*. A trial lasts 8 seconds, in which the word was presented for 7 seconds followed by a 1 second fixation-cross. Each block has 48 seconds and contains 6 trials. In the resting block, a fixation-cross was presented for 30 seconds. The duration of the entire scanning session was 272 seconds, composed of 4 pseudo-word blocks and 2 rest blocks. 2 extra rest blocks lasting 10 seconds were added between pseudo-word blocks.

We acquired 1 image for each two seconds of the test, which was subsequently upsampled by a factor of 2, yielding a total of 272 images and performed 2 transformations in the data in order to obtain the training examples. The fMRI scanner used to acquire data generates 3D images with 3x3x3 mm voxels. First, we transformed the images from a voxel size of 3x3x3mm into images of voxels with 2x2x2m using the AFNI neuroimaging tool[1] [Cox 1996] in order to apply an image segmentation technique based on image templates. Thus, because the template is 2x2x2mm, we need to adjust the test data to fit the template voxel size. Second, we used a template that divides the brain into 116 distinct regions [Alemán-Gómez et al. 2006], grouping voxels that belong to the same region. Basically, this template is a bit mask, which, for each region, turns off all voxels that does not belong to that region, and then extracts the time series of that region by calculating the average brain activation of the voxels at each second. By using this mask, we obtain 116 time series data representing the average activation of each region. After processing the data, we have a 272x116 matrix, where the 116 columns are the brain regions from the template and each line is the average brain activation in 1 second. Thus, each cell of this matrix is the average brain activation in one region in a given time, represented by a real number.

We now want to create 2 types of examples with the resulting matrix: when the subject is performing a task; and when the subject is resting. For the task examples, we have to look at the 7 seconds in which the word is presented, calculating the average activation of each brain region in that time. As we have seen in Section 2, the BOLD signal takes 2 seconds to start and lasts only 4 seconds. Then, from the total 7 seconds of the task, we have to skip 2 seconds in order to wait for the BOLD signal to occur and look at the next 4 seconds, which is the amount of time that the bold signal lasts. Thus, from the 7 seconds of a task, we calculate the average activation on seconds 3,4,5 and 6.

After applying such preprocessing on the task blocks, we are left with 24 task examples (4 blocks with 6 tasks each). For the remaining examples, we use the 2 rest blocks. For each rest block, we remove the first 4 seconds, waiting for the BOLD signal clear after the task block. Consequently since each rest block lasts 30 seconds (4 of which were removed to wait for the BOLD signal to clear), we are able to create 7 examples in sequence of the remaining 26-second block, where each example is the average brain activation in 4 seconds.

Finally, we have the examples to use with the classification algorithm: 24 task examples and 14 rest examples (7 examples for each block), for a total of 38 examples. Each example is a vector of 116 elements, where each element is the average brain activation in one region for 4 seconds.

4. Experiments

Our goal is to train an SVM classifier to differentiate between task and rest examples. For this purpose, we search for parameter values that we have seen in 2.2 in order to obtain a high prediction rate. We used the SVM algorithm from LIBSVM [Chang and Lin 2011] and performed the 6 steps below in our experiments.

1. We transformed the fMRI data in examples for the SVM algorithm, as we have seen in Section 3.

[1] AFNI is a popular neuroimaging software available at: `http://afni.nimh.nih.gov`

2. We scaled the features in data, so all features in all examples are real values between [-1,1]. Scaling data prevents overfitting, it reduces the differences between the features, minimizing the problem of one feature having much more importance than the others [Pereira et al. 2009].
3. In order to maximize the prediction accuracy, we have to find the best parameters for the SVM algorithm with the data, train the algorithm with these parameters and test the algorithm success rate. However, we cannot use the same examples for finding good parameters, training and testing. Using the same data for all of these steps can overfit the results. Therefore, we separated the data in 3 sets: the tuning set, which we use to find good parameters for the algorithm; the training set, which we use to train the algorithm using the parameters we have found and perform cross-validation and the test set, which we use to validate the classifier by measuring its expected accuracy.
4. After splitting the data, we need to find good parameters for the SVM algorithm using the tuning set. For this purpose, we use the RBF kernel along with the SVM algorithm, which maps the examples in a high dimensional space and fits well non-linear data [Keerthi and Lin 2003]. We use cross-validation several times with predefined parameters in order to optimize these parameters using Grid Search [Nelder and Mead 1965]. The Grid Search procedure gives us the parameters that generated the best prediction rate results.
5. We trained the SVM algorithm again setting the new parameters we found with Grid Search using the data from the test set, so the results are not overfitted. We use the training set to measure the SVM accuracy with leave-one-out cross-validation.
6. We validated the results by measuring the trained SVM true-accuracy with the test set, that the algorithm has never seen before.

5. Results

We now evaluate the accuracy of the experiments described in Section 4 using the data from 3 patients (referred to as P001, P002 and P003). In our experiments, each patient data was used to train a separate classifier.

As described in step 3, the data of each patients is split into 3 sets: the tuning set, the training set and the testing set. Table 1 shows the amount of examples of task and rest we use in each set, as the number of examples is unbalanced (there are more examples of task than rest).

set	task	rest	total
tuning	10	5	15
train	12	7	19
test	2	2	4

Table 1. Data of a single patient split in 3 sets

Table 2 shows the result of the experiment using the data of each patient. First, we present the RBF kernel parameter values, C and γ, that we find when tuning the SVM algorithm. Second, the cross-validation accuracy using the training data and the new

Patient	C	γ	cross-validation accuracy	true-accuracy
P001	2	0.0078125	78.94%	75%
P002	2	0.0078125	73.68%	100%
P003	128	0.00048828125	42.1%	50%

Table 2. Results of the experiments described in section 4

parameter values. Finally, we show the expected accuracy of the trained SVM in the test set.

When looking at the results of Table 2, we note that we work with a limited number of examples. To measure the expected accuracy there are only 4 examples, so that if the SVM algorithm misclassifies one example we lose 25% of the expected accuracy. Notice that the accuracy for the SVM classifier generated with P001 and P002 data is high, and their respected tuning parameters are the same. We observe that the selected parameters are not distant from the the starting parameters ($C = 8$, $\gamma = 0.5$). On the other hand, the SVM algorithm accuracy for P003 data is equal to random guessing, and the parameter values are far from the other patients data and and the starting parameters from the tuning algorithm.

In order to verify if all the steps in section 4 contribute to increase the prediction rate of the SVM algorithm, we created other 3 experiments by removing some important steps. In this first experiments, we do not scale the data before tuning the algorithm parameters, training and testing. In the second experiment, we do not boost the algorithm parameters. And in the third experiment, we do not scale the data nor boost the parameters. The true accuracy of each experiment is in table 3.

Patient	no scale	no parameter boosting	no scale and no parameter boosting
P001	75%	50%	50%
P002	100%	100%	50%
P003	75%	50%	50%

Table 3. True accuracy when we do not scale the data or boost the parameters

The results of table 3 show two important facts. First, scaling is not very helpful on this data. Although the true accuracy for P001 and P002 data when scaling or not is the same, the true accuracy of P003 data increases when not scaling. Looking at the not scaled data, we notice that it is between [-4, 3]. Thus, the difference between the data values is not significant enough to perform scaling, and the true accuracy gets better when not scaling. Second, when not boosting the parameters of the SVM algorithm, only the true accuracy of P001 data decreases. As the difference between the default parameter values and the parameter values found from the tuning algorithm are close, not tuning the parameter does not change dramatically the true accuracy from P001 and P002 data. The tuning algorithm seems not to have found good parameters for P003 data, as they are too different from P001 and P002 parameter values and the resulting accuracy is the same as random guessing. Thus, changing the parameter values from P003 data does not change the accuracy.

We have seen that our approach for obtaining a high prediction rate and differ-

entiate task examples for rest examples need improvements, as we do not obtained the expected results from all patients data. Moreover, we learn which are the relevant steps for finding good results, and why some steps are unnecessary.

6. Conclusion

In this paper we described an application of machine learning algorithms to discover what a patient is doing at any given time in the fMRI scan session. To apply the machine learning algorithm to our problem, we transformed the fMRI test in readable data for the SVM algorithm and created experiments to test the true accuracy of the SVM algorithm using the data. in order to evaluate which steps of our main experiments are relevant, we created other smaller experiments and compared the results.

As future work, we aim to generalize the classifier so that we can train using the data from multiple patients and predict the classification of examples of data from an unknown patient. Moreover, we want generate an SVM classifier to differentiate when a patient is seeing a word or a pseudo-word.

Acknowledgement

We would like to thank Augusto Buchweitz from The Brain Institute of Rio Grande do Sul (InsCer) for making available the data collected under the ACERTA project. We would also like to thank CAPES for financial support under the OBEDUC project number 14385.

References

Alemán-Gómez, Y., Melie-García, L., and Valdés-Hernandez, P. (2006). Ibaspm: toolbox for automatic parcellation of brain structures. In *12th Annual Meeting of the Organization for Human Brain Mapping*, volume 27.

Biswal, B., Zerrin Yetkin, F., Haughton, V. M., and Hyde, J. S. (1995). Functional connectivity in the motor cortex of resting human brain using echo-planar mri. *Magnetic resonance in medicine*, 34(4):537–541.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cox, R. W. (1996). Afni: software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical research*, 29(3):162–173.

Hoeft, F., McCandliss, B. D., Black, J. M., Gantman, A., Zakerani, N., Hulme, C., Lyytinen, H., Whitfield-Gabrieli, S., Glover, G. H., Reiss, A. L., et al. (2011). Neural systems predicting long-term outcome in dyslexia. *Proceedings of the National Academy of Sciences*, 108(1):361–366.

Huettel, S. A., Song, A. W., and McCarthy, G. (2004). *Functional magnetic resonance imaging*, volume 1. Sinauer Associates Sunderland.

Just, M. A., Keller, T. A., Malave, V. L., Kana, R. K., and Varma, S. (2012). Autism as a neural systems disorder: a theory of frontal-posterior underconnectivity. *Neuroscience & Biobehavioral Reviews*, 36(4):1292–1313.

Keerthi, S. S. and Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689.

Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45.

Mitchell, T. M., Hutchinson, R., Just, M. A., Niculescu, R. S., Pereira, F., and Wang, X. (2003). Classifying instantaneous cognitive states from fmri data. In *AMIA Annual Symposium Proceedings*, volume 2003, page 465. American Medical Informatics Association.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Computer journal*, 7(4):308–313.

Pereira, F., Mitchell, T., and Botvinick, M. (2009). Machine learning classifiers and fMRI: A tutorial overview. *NeuroImage*, 45(1, Supplement 1):S199–S209. Mathematics in Brain Imaging.

Powell, M. J. (1987). Radial basis functions for multivariable interpolation: a review. In *Algorithms for approximation*, pages 143–167. Clarendon Press.

Russell, S. J. and Norvig, P. (2010). *Artificial intelligence: a modern approach*, volume 2. Prentice hall.

Vapnik, V. (2000). *The nature of statistical learning theory*. springer.

Wolf, M. and Bowers, P. G. (1999). The double-deficit hypothesis for the developmental dyslexias. *Journal of Educational Pscyhology*, 91(3):415–438.

Woodard, J., Seidenberg, M., Nielson, K., Antuono, P., Guidotti, L., Durgerian, S., Zhang, Q., Lancaster, M., Hantke, N., Butts, A., et al. (2009). Semantic memory activation in amnestic mild cognitive impairment. *Brain*, 132(8):2068–2078.

Woodard, J. L. and Sugarman, M. A. (2012). Functional magnetic resonance imaging in aging and dementia: Detection of age-related cognitive changes and prediction of cognitive decline. In *Behavioral Neurobiology of Aging*, pages 113–136. Springer.

Modelagem de Emoções utilizando Redes Bayesianas

Felipe Neves da Silva[1], Adriano V. Werhli[1], Diana F. Adamatti[1]

[1]Progama de Pós-Graduação em Modelagem Computacional – Universidade Federal do Rio Grande (FURG)

fnds9@yahoo.com.br,{werhli,dianaada}@gmail.com

Abstract. *Although there are theoretical models for the functioning of emotions, these alone are insufficient for an accurate simulation in computational means. Therefore, we developed a Bayesian Network inspired by a model of emotions, the OCC model, applied to a multiagent system with the aim of simulating emotions in agents. The combined use of Bayesian Networks and the OCC model structure seeks to add unpredictability to the originally deterministic model, as well as the application of the proposed model to a multiagent system provides the study of the influence of emotions on the actions and behavior of agents.*

Resumo. *Embora existam modelos teóricos para a modelagem de emoções, estes por si só não são suficientes para uma simulação precisa em meios computacionais. Deste modo, este trabalho propõe a construção de uma rede Bayesiana baseada em um modelo de emoções, o modelo OCC, a qual é aplicada a um ambiente multiagentes, de forma a simular emoções em agentes. O uso combinado de redes Bayesianas e da estrutura do modelo OCC procura adicionar imprevisibilidade ao modelo originalmente determinista, bem como a aplicação do modelo proposto para um sistema multiagentes propicia o estudo da influência das emoções sobre as ações e o comportamento de agentes.*

1. Introdução

O ser humano sempre foi alvo de diversos estudos e serviu inspiração nas mais diversas áreas do conhecimento, entre elas a computação. A Inteligência Artificial (IA), por exemplo, possui algumas técnicas inspiradas tanto na fisiologia como no comportamento individual e social do ser humano. Os Sistemas Multiagentes oferecem estruturas próprias para a simulação das mais diversas situações, denominados agentes, que são capazes de interagir entre si e com o ambiente ao qual estão inseridos. Com essas características, se tornaram a ferramenta preferida pelos desenvolvedores para realização da simulação das relações humanas em meios computacionais.

Entretanto o comportamento humano é determinado por diversas variáveis, muitas delas sem um método de simulação computacional definida, como as emoções, que interferem de forma decisiva no comportamento humano, influenciando a tomada de decisões, ações, memória, atenção, etc. (GRATCH; MARSELHA, 2001).

Existem modelos teóricos que tentam formalizar o funcionamento das emoções, dentre os quais se destaca o modelo OCC (ORTONY et al., 1988), que relaciona as emoções aos eventos que as geram. Composto de 22 emoções, o modelo admite três formas de estímulo, eventos do ambiente, ações de indivíduos e objetos. O modelo se baseia no princípio da diferenciação entre reações de valência positivas e negativas, ou

seja, a partir de um estímulo do ambiente, variáveis são atribuídas de forma a determinar se o evento proporciona sentimentos positivos ou negativos para o indivíduo modelado. Sendo assim, o modelo gera sempre as mesmas emoções a partir de um estímulo de mesma valência.

Este artigo apresentada a possibilidade de se transformar o modelo OCC em uma rede Bayesiana baseado em sua estrutura acrescentando, assim, diferentes características ao modelo, como variáveis e valores probabilísticos. Esta nova estrutura possibilita que a partir de modificações para os valores de inicialização para rede apresentada, seja possível o trabalho com emoções para diferentes perfis de indivíduos, bem como que seja possível avaliar a intensidade das emoções simuladas.

As redes Bayesianas são uma excelente ferramenta para modelagem de problemas reais por utilizarem o raciocínio probabilístico, que se diferencia do raciocínio lógico, utilizado pela maioria das ferramentas computacionais, por permitir o trabalho com informações incompletas do ambiente, situação comum neste tipo de problema, seja pela dificuldade, imprecisão ou, até mesmo, impossibilidade de coleta destas informações.

Para a utilização prática e avaliação dos resultados gerados da união do modelo OCC com as redes Bayesianas, propõe-se ainda que este modelo híbrido seja inserido em um sistema multiagentes, utilizando um exemplo para que se possam avaliar as mudanças de comportamento dos agentes devido a emoções geradas a partir de estímulos do ambiente, permitindo, desta forma, que os agentes tornem-se ainda mais próximos à realidade humana.

O artigo está estruturado em 5 seções. A seção 2 apresenta uma revisão sobre as três técnicas utilizadas na realização da modelagem. Na seção 3, é apresentada a rede Bayesiana de emoções desenvolvida, bem como sua aplicação a um sistema multiagentes e o estudo de um exemplo avaliando suas características. Os resultados obtidos a partir de exemplo são discutidos na seção 4, enquanto a seção 5 apresenta as conclusões, bem como os futuros trabalhos propostos.

2. Revisão Bibliográfica

2.1. Redes Bayesianas

Um grande problema para modelagem de problemas reais é a falta de informações completas sobre seu ambiente, seja pela dificuldade, imprecisão ou, até mesmo, impossibilidade de sua coleta. Nestes casos, a utilização do raciocínio probabilístico, bem como dos métodos que o utilizam se apresenta como uma boa alternativa (RUSSEL e NORVIG, 2003).

Uma importante ferramenta para modelagem de ambientes de incerteza são as chamadas redes Bayesianas que podem ser definidas como uma combinação da Teoria Probabilística e a Teoria de Grafos. Elas oferecem uma representação gráfica das relações probabilísticas existentes entre os diversos componentes do ambiente a ser modelado (WERHLI, 2007).

A Figura 1 apresenta a estrutura básica de uma rede Bayesiana, um grafo acíclico direcionado, ou seja, um grafo em que todas suas arestas são direcionadas e não existem ciclos.

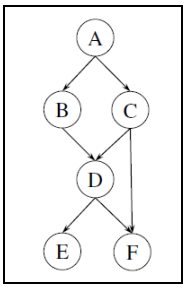


Figura 1. Rede Bayesiana composta pelos nós N={A, B, C, D, E, F} e pelas arestas €={(A,B),(A,C),(B,D),(C,D),(D,E),(D, F),(C, F)} (WERHLI, 2007).

Redes Bayesianas utilizam a Teoria Probabilística para modelar as incertezas existentes em um determinado ambiente. A Teoria Probabilística é um campo da matemática que estuda e analisa a ocorrência de fenômenos aleatórios, os quais são experimentos repetidos sob as mesmas condições produzem resultados que não se pode prever com certeza (MORGADO et al., 2001). Dois tipos de probabilidades devem ser destacados: condicional e incondicional. O segundo, mais simples, é a probabilidade que independe de acontecimentos anteriores, o contrário do primeiro, o qual é amplamente utilizado para modelagem de redes Bayesianas.

Representa-se uma probabilidade condicional por P(A | B), que significa a probabilidade de que o evento A ocorra dado a ocorrência de um evento B.

Considerando novamente a rede Bayesiana hipotética apresentada na Figura 1, sua estrutura é composta pelo grupo de nós N={A, B, C, D, E, F} e pelo conjunto de dependências entre nós representado pelo grupo de arestas €={(A, B), (A, C), (B, D), (C, D), (D, E), (D, F), (C, F)}. Existindo uma aresta direcionada do vértice A paro o nó B, então A é chamado pai de B, assim como B é dito filho ou descendente de A.

Uma rede Bayesiana é caracterizada por possuir uma regra simples e única capaz de expandir seu conjunto de probabilidades em termos de simples probabilidades condicionais. Esta regra segue a propriedade local de Markov, a qual diz que um nó é condicionalmente independente de todos seus não descendentes dados seus pais.

Após se definir uma rede Bayesiana, com suas variáveis e probabilidades, pode-se extrair conhecimento nela representado através de um processo de inferência. Segundo (HRUSCHKA JR., 2003), existem diversos métodos para realização de inferência, dentre os quais se podem destacar o método de inferência por eliminação de variáveis (utilizado neste trabalho), e o método do agrupamento.

2.2. Modelagem de Emoções

Existem diversas propostas para a modelagem de emoções, de forma a tentarem apresentar uma explicação melhor sobre como estas funcionam. Além disso, estas propostas oferecem o modelo básico para que emoções sejam simuladas em máquina (GRATCH; MARSELHA, 2001).

Porém, realizar a simulação de emoções em máquina não é uma tarefa fácil. Em tarefas onde as emoções exercem um papel fundamental, como processos de tomada de decisão, diversos fatores, tanto sociais quanto fisiológicos, tornam a realização da modelagem e simulação do processo bastante complexa.

Outro ponto que dificulta o trabalho com emoções é a dificuldade de compreensão do que são e como funcionam. Não existe um consenso sobre a definição de emoções. Del Nero (1997) afirma que emoção é um processo consciente e que em

conjunto com o pensamento e a vontade formam os protagonistas principais para o palco que é a mente. Para Damásio (2000), a emoção é um rótulo que designa um conjunto de fenômenos ou comportamentos. Ele divide as emoções em primárias (medo, alegria, tristeza, raiva, etc.) e as secundárias (ciúme, culpa, orgulho, etc.). Moffat e Frijda (2000) afirmam que as emoções são funcionais, isto é, elas possuem um valor adaptativo para o organismo, contradizendo a convenção existente de que as emoções não são racionais. Sloman (2001) também conclui que não há uma definição única de emoção, pois esta depende de como se analisa quais são as concepções individuais dos seres humanos ou outros animais em relação ao assunto.

Quando se trata sobre o funcionamento das emoções, devem-se levar em conta duas características: as emoções são processos fisiológicos de difícil mensuração, são as coisas que sentimos; emoções são geradas por estímulos, entretanto é impossível afirmar que um mesmo estímulo irá gerar a mesma emoção em dois indivíduos diferentes. Isto se deve a diversos fatores que, em resumo, definem cada pessoa como um ser diferente.

A fim de um melhor entendimento do funcionamento das emoções, foram propostos diversos modelos para a estruturação das emoções, cada uma visando aspectos diferentes do ser humano. Alguns de cunho psicológico, como percepção, sentimentos, experiências, cognição e comportamento (MOFFAT; FRIJDA, 2000) (CAÑAMERO; VAN DE VELDE, 1999) (ORTONY et al., 1988), e outros de cunho fisiológico, como batimentos cardíacos, pressão arterial e sudorese (SLOMAN, 1999) (PICARD, 1997).

Dentre os modelos propostos, destaca-se o proposto por Ortony, Clore e Collins (ORTONY et al., 1988). Conhecido como OCC, o modelo é capaz de identificar, a partir de estímulos gerados em um ambiente arbitrário, quais emoções seriam geradas, dentro de um conjunto predeterminado de emoções. Este é um dos modelos mais utilizados no ramo da computação, seja para adicionar emoções a agentes artificiais ou para se trabalhar com a tomada de decisão influenciada pelas emoções.

O modelo se baseia no princípio da diferenciação entre reações de valência positivas e negativas, ou seja, a partir de um estímulo do ambiente, variáveis são atribuídas de forma a determinar se o evento proporciona sentimentos positivos ou negativos para o indivíduo modelado.

No modelo OCC são considerados três geradores de estímulos para emoções: eventos, os quais têm suas consequências analisadas de forma a gerar emoções; agentes, onde se analisam suas ações; e objetos, em que se analisam seus aspectos e propriedades.

Toda emoção gerada no modelo é uma reação a um ou mais aspectos do ambiente. Entretanto, para indivíduos distintos, um determinado estímulo pode gerar emoções diferentes. Esta diferenciação ocorre no modelo a partir da atribuição de um valor positivo ou negativo como reação do indivíduo para uma determinada ocorrência. Este conceito se torna mais claro ao se observar a estrutura do modelo na Figura 2.

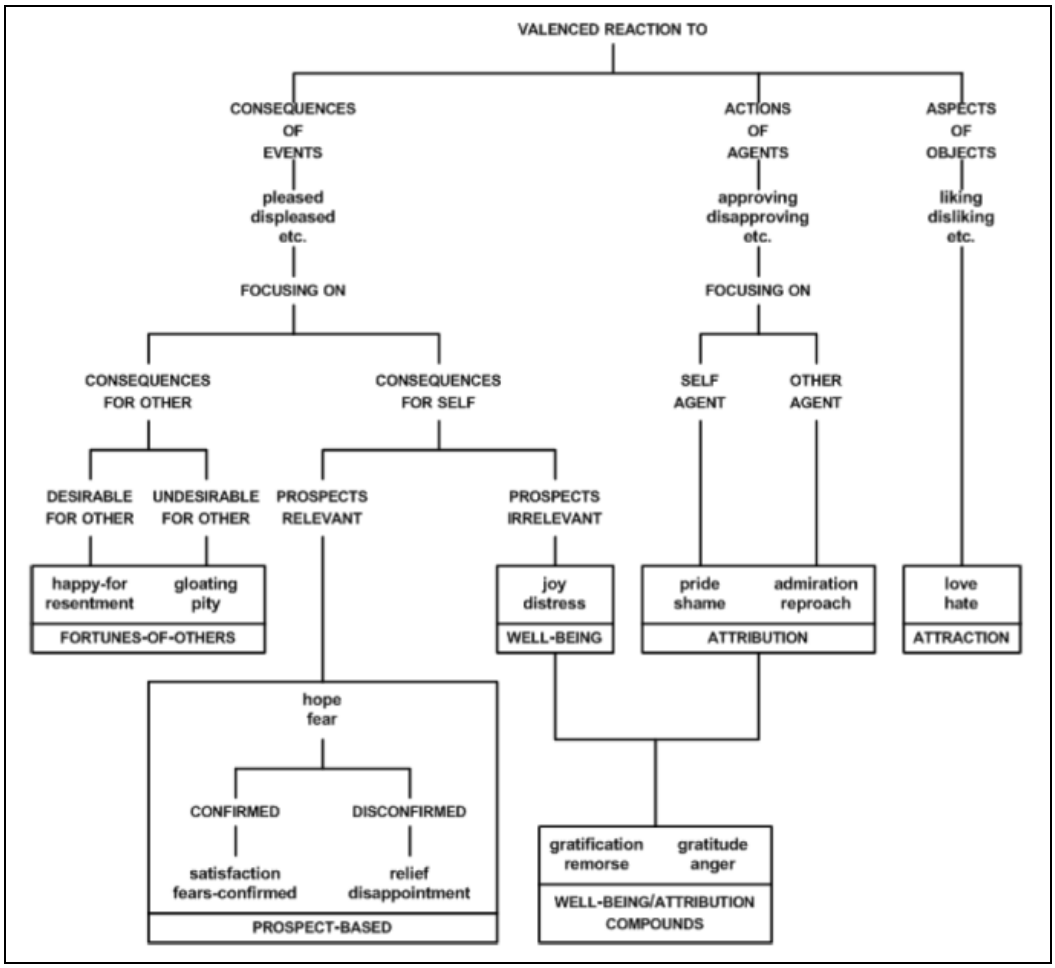


Figura 2. Estrutura do Modelo OCC (ORTONY et al., 1988).

A estrutura do modelo pode ser dividida em três ramos principais, cada um correspondendo a um tipo de estímulo gerado pelo ambiente. O ramo mais a esquerda refere-se ao desenvolvimento de emoções geradas a partir de eventos, o ramo central das emoções geradas por ações e o ramo a direita refere-se às emoções produzidas a partir de objetos. É importante ressaltar que a estrutura apresenta uma descrição lógica para a geração de emoções e não uma sequência temporal. A parte inferior do modelo apresenta o conjunto de emoções trabalhadas, as quais totalizam 22 emoções distintas.

2.3. Sistemas Multiagentes

De mesma forma como nas organizações humanas as atividades, muitas vezes, são realizadas por um grupo de pessoas que trabalham de modo cooperativo, onde existem decisões individuais que afetam o grupo, em sistemas multiagentes (SMA) as pessoas são representadas por agentes artificiais, os quais se relacionam em um ambiente de forma a buscar soluções para problemas de forma cooperativa, compartilhando informações, evitando conflitos e coordenando a execução de atividades (ADAMATTI, 2003).

De maneira geral, pode-se dizer que um agente artificial em um sistema mutiagentes é uma entidade dotada de certa autonomia e inteligência inserida em um ambiente virtual, sendo capaz de perceber e interagir com os componentes deste ambiente, incluindo os possíveis demais agentes que o habitam.

Na maior parte das vezes, uma aplicação desenvolvida sobre um SMA tem o objetivo de simular alguma situação da realidade. Em uma simulação baseada em agentes, o fenômeno real é decomposto em um conjunto de elementos e em suas interações. Para cada elemento é modelado como um agente, resultando em um modelo geral onde os agentes interagem entre si e com o ambiente (FROZZA, 1997).

Para Russel e Norwig (2003), a modelagem de sistemas multiagentes e sua simulação exigem algumas características: os agentes devem ser autônomos; o comportamento dos agentes deve ser representado em alto nível de abstração; agentes devem ser flexíveis, tendo características de comportamento pró-ativo e reativo; agentes podem executar tarefas que exijam desempenho de tempo real; agentes se encaixam em aplicações distribuídas; agentes devem possuir habilidade de trabalhar cooperativamente.

3. A Rede Bayesiana de Emoções

O modelo OCC de emoções, bastante abrangente e que possui uma estrutura de simples tradução computacional, foi a escolha como modelo base para rede Bayesiana de emoções proposta.

O modelo OCC por si só é um modelo previsível, isto é, um modelo que não considera a imprevisibilidade humana, determinando sempre o mesmo resultado emocional a partir de eventos ou ações de mesma valência para o indivíduo. Desta forma, propõe-se a construção de uma rede Bayesiana que possua uma estrutura inspirada no modelo OCC de emoções e que possa, através de suas características básicas, adicionar a imprevisibilidade necessária ao modelo de simulação.

A utilização de redes Bayesianas para adição de imprevisibilidade em modelos de emoções ocorre, de maneira geral, de forma diferente à proposta neste trabalho como pode ser visto na Figura 3. Nesta figura é possível comparar os fluxogramas de funcionamento do trabalho proposto, Figura 3b, com o de um trabalho semelhante, o de Conati et al. (2010), Figura 3a.

Enquanto no trabalho de Conati et al. (2010) a rede Bayesiana é utilizada para realizar um pré-processamento para os estímulos do ambiente de forma a propiciar alimentação para o modelo OCC clássico, estático e definido por uma sequência de desvios condicionais, o modelo proposto apresenta uma rede Bayesiana que redefine o modelo OCC, possuindo uma estrutura semelhante ao original mas acrescentando uma série de características que o transformam em um modelo dinâmico que apresenta diferentes resultados emocionais para os mesmos estímulos do ambiente em diferentes momentos.

Estas características propiciam, entre outras possibilidades, o trabalho com perfis comportamentais onde dependendo das características do indivíduo simulado a rede pode ser inicializada com valores diferente, fazendo com que apresente emoções diferentes as de um indivíduo com outra inicialização para as mesmas circunstâncias ambientais.

Para sua construção e análise foi utilizado um software livre auxiliar chamado JavaBayes (COZMAN, 2001). O software permite a construção, visualização gráfica e análise de redes Bayesianas através de uma interface simples e de fácil acesso. O software possibilita a construção de redes com qualquer estrutura, com um número indefinido de nós, arestas e variáveis. Além disso, oferece diversas funções, como exemplo, a seleção do algoritmo para o cálculo de probabilidades entre os métodos

agrupamento e eliminação de variáveis e a possibilidade de exibir as probabilidades de todas as variáveis da rede a partir da observação de um de seus nós.

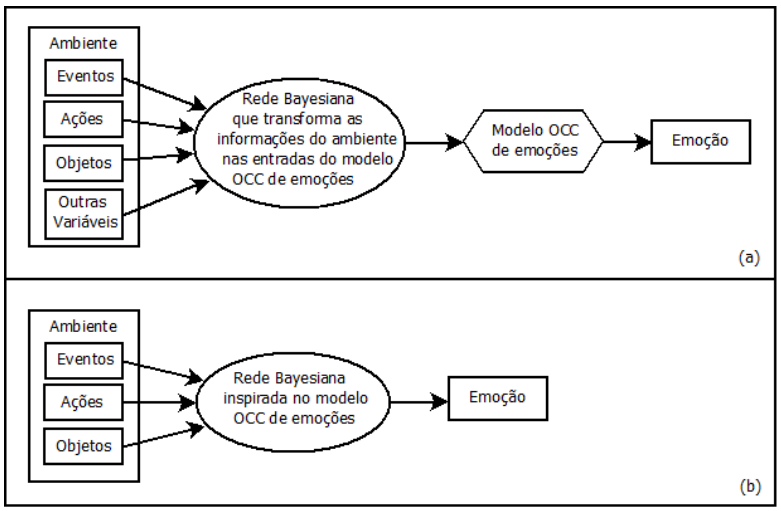


Figura 3. Comparação entre as diferentes combinações entre redes Bayesianas e o modelo OCC. (a) Método mais utilizado onde a rede Bayesiana processa as informações do ambiente e cria as informações de entrada do modelo OCC. (b) Modelo proposto, onde o modelo OCC é substituído por uma rede Bayesiana de estrutura inspirada no modelo de emoções.

Como o modelo OCC de emoções é constituído basicamente de uma sequencia de desvios condicionais que possibilitam a determinação de qual emoção é ativada a partir de determinadas condições, a rede Bayesiana baseada em sua estrutura foi construída traduzindo cada um destes desvios em um conjunto de nós, variáveis e arestas, onde cada nó representa um desvio, cada variável representa um dos estados que satisfazem o desvio e cada aresta liga cada desvio a próxima condição a ser analisada. O resultado desta tradução é uma rede constituída de 34 nós e 49 arestas que pode ser visualizada na Figura 4.

Ao observar a estrutura da rede é possível verificar que, assim com ocorre com o modelo OCC de emoções, ela pode ser dividida em três áreas distintas de acordo com a fonte de estímulo do ambiente. Os nós na área à esquerda representam os estímulos gerados a partir de eventos ocorridos no ambiente, a área central compreende os nós relativos a estímulos gerados a partir das ações de outros indivíduos, enquanto a menor área, localizada a direita, refere-se às emoções geradas a partir do contado do indivíduo com diferentes objetos. Na parte inferior da rede encontram-se os nós que representam cada uma das 22 emoções modeladas pelo sistema: *Happy-for, Resentment, Gloating, Pity, Hope, Fear, Joy, Distress, Satisfaction, Disappointment, Fears-confirmed, Relief, Pride, Shame, Admiration, Reproach, Love, Hate Gratification, Remorse, Gratitude* e *Anger*.

Como é característico das redes Bayesianas, cada nó da rede depende diretamente de seus nós pais, indicados pelas arestas, construídas na direção pai-filho, sendo permitido tanto um nó pai ter múltiplos nós filhos, como também um nó filho possuir múltiplos nós pai. Seguindo esta dinâmica, bem como as relações existentes entre emoções e estímulos existentes no modelo OCC, foram definidas tanto as variáveis de cada nó como também as probabilidades iniciais da rede.

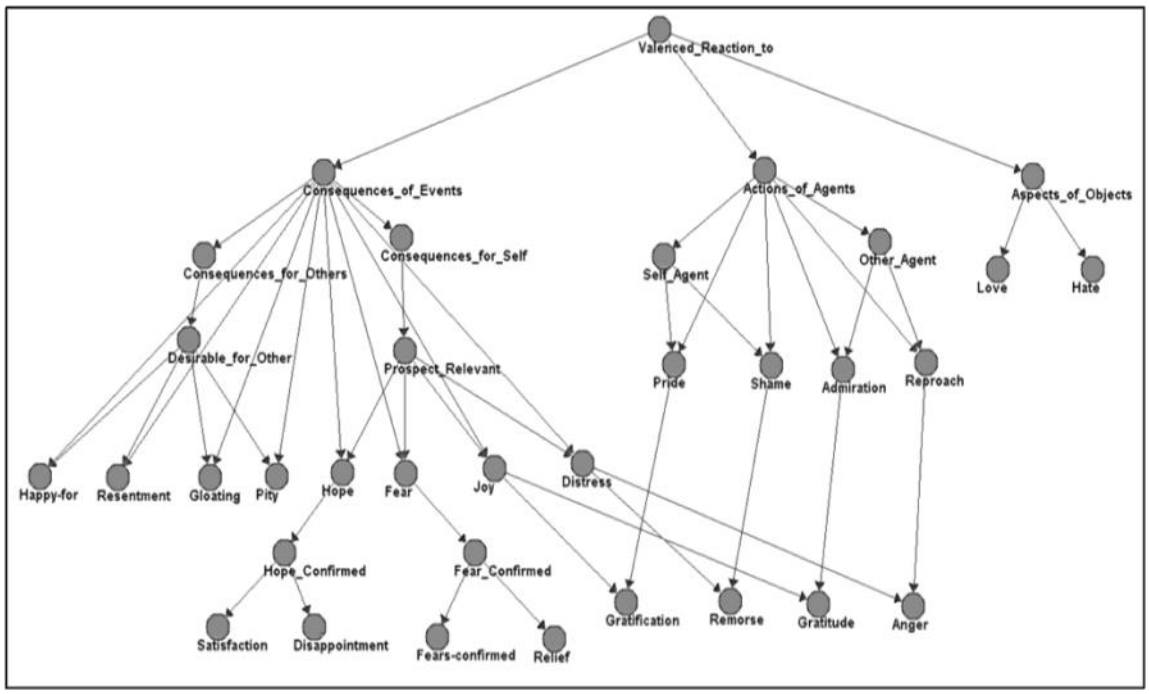


Figura 4. Rede Bayesiana desenvolvida, inspirada no modelo OCC de emoções, onde cada nó representa um desvio condicional no modelo original e as arestas representam as dependências pai-filho entre estes nós.

Cada nó possui uma variável que pode assumir dois estados, sendo em geral, inclusive para os nós que compreendem as emoções, estes estados "*true*" e "*false*", onde "*true*" representa um estado ativo, a confirmação de uma condição ou a efetividade de uma emoção; enquanto "*false*" significa o contrário. A exceção a esta regra são os três nós superiores a cada uma das áreas anteriormente citadas: *Consequences_of_Events, Actions_of_Agents* e *Aspects_of_Objects.*

Para estes três nós, suas variáveis possuem estados e funções diferentes dos demais: a variável do nó *Consequences_of_Events* pode assumir os estados "*Pleased*" e "*Displeased*" que, respectivamente, indicam se ocorreu um evento desejável ou indesejável para o indivíduo no ambiente. O nó *Actions_of_Agents*, a variável pode assumir os valores "*Approving*" e "*Disapproving*", que de maneira semelhante aos estados do nó *Consequences_of_Events* indicam se o individuo esta de acordo ou não com a ação de um individuo que ativou o nó. Por fim, a variável do nó *Aspects_of_Objects* pode assumir os estados "*Liking*" e "*Disliking*" que indicam se o indivíduo gosta ou não, respectivamente, do objeto que estimulou a rede Bayesiana.

Após definir os estados possíveis para as variáveis de cada um dos nós, foi necessário definir as probabilidades de ativação de cada um dos estados seguindo as dependências entre nós pais e filhos. A rede apresenta duas situações: nós que possuem um ou dois pais. Para o primeiro caso, os estados da variável ainda podem ou não depender dos estados do nó pai. Quando eles não dependem, suas probabilidades foram definidas como 50% de ocorrer qualquer um dos estados independente dos estados do pai. Já quando ocorre dependência, um dos estados do pai faz com que o filho tenha 95% de chance de ter um determinado estado, enquanto seu estado contrário faz com que seu nó filho possua também 95% de chances de ter o estado contrário. Quando um nó possui dois nós pais ele sempre depende dos estados destes, sendo definido que o filho possui 95% de possuir um dos seus estados quando um determinado estado para

cada um de seus pais. Quando os estados dos pais se invertem, o nó filho tem 95% de chance de também ter seu estado invertido. No caso de um dos pais terem seu estado invertido, o nó filho tem 50% de chance de ter cada um dos seus estados.

Após criar a estrutura da rede e definir todas as probabilidades é necessário adicionar a rede Bayesiana de emoções a um ambiente multiagentes de forma a validá-la e propiciar a avaliação de sua eficácia. Deste modo, Jason foi o ambiente multiagentes escolhido. Ele é desenvolvido na linguagem Java e possibilita de forma simples a execução de simulações baseadas em sistemas multiagentes (BORDINI et al., 2007).

Em sua base de dados padrão, Jason oferece uma série de exemplos de modelos multiagentes que simulam as mais diversas situações. Dentre eles se encontra o chamado *cleaning_robots*, o qual foi utilizado como base para o estudo da funcionalidade e efetividade da rede Bayesiana de emoções em um ambiente multiagentes.

Neste exemplo, dois robôs R1 e R2 coletam e eliminam lixo no planeta Marte. O robô R1 anda sobre o solo do planeta procurando unidades de lixo. Ao encontrar uma unidade, o agente a recolhe e leva até o ponto onde está R2, em seguida retornando ao ponto onde encontrou a unidade para continuar a busca. O robô R2, por sua vez, está posicionado junto a um incinerador e ao receber uma unidade de lixo imediatamente a queima.

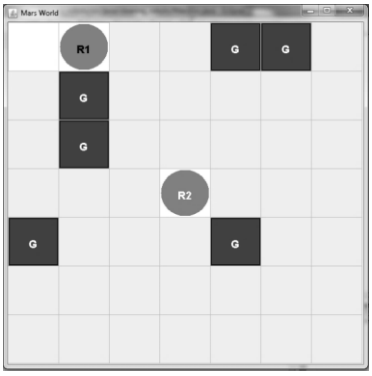


Figura 5. Interface Gráfica do exemplo cleaning_robots na ferramenta Jason, mostrando os agentes R1 e R2.

A Figura 5 apresenta uma visão geral do ambiente de simulação. As unidades de lixo, representadas por G no mapa, são colocados em posições randômicas na grade de posições assim que a simulação se inicia. O agente R1 inicia sempre na posição superior mais a esquerda, realizando sua busca percorrendo todas as posições do mapa, linha a linha e sempre da esquerda para direita. O agente R2 inicia sempre na posição central do mapa e fica fixo todo o tempo de simulação nesta posição.

Neste trabalho, apenas o agente R1 trabalha sobre o efeito de emoções. Deste modo, para adicionar a rede Bayesiana de emoções a este exemplo foi necessário que se desenvolvessem suposições sobre como o ambiente influenciaria a rede e como a rede influenciaria as ações do agente. Três suposições foram definidas:

• Quando R1 realiza cinco passos consecutivos sobre o mapa e não encontra nenhuma unidade de lixo ele passa a pensar que o ambiente pode estar limpo e que, portanto, seu trabalho está sendo realizado em vão o que o desmotiva;

• Quando R1 encontra uma unidade de lixo o efeito é o contrário da proposta na suposição anterior, por estar realizando a função a qual completa seu objetivo o agente aumenta sua motivação;

• Quando R1 deposita a unidade de lixo recolhida na posição onde está R2 existe a possibilidade de este agradecer R1 pelo bom trabalho, o que também o motiva.

5. Resultados

Para avaliar os efeitos da rede Bayesiana de emoções sobre o exemplo trabalhado é interessante que se obtenha uma forma de mensurar o desempenho de R1 ao realizar a limpeza do mapa. Assim, serão observados quantos ciclos de execução são necessários para o agente realizar a busca por unidades de lixo em todo o mapa. Um ciclo de execução compreende o período de tempo necessário para que o agente realize uma ação no ambiente.

Para este exemplo, serão considerados apenas os ciclos utilizados para realização de uma ação chamada *nextSlot*, função responsável por fazer o agente R1 andar sobre o mapa e a única função a sofrer influência direta das emoções segundo as suposições realizadas.

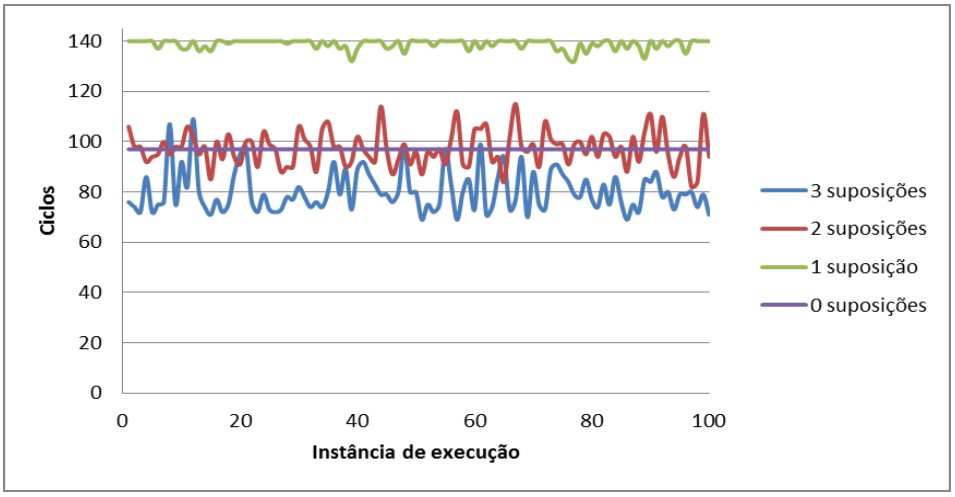


Figura 6. Tempo de execução para 100 instancias do exemplo com quatro configurações distintas: sem suposições, com uma suposição, com duas suposições e com três suposições.

Para demonstrar o aumento da complexidade do exemplo com o acréscimo de emoções, são apresentados os resultados obtidos a partir da execução de 100 instâncias do exemplo para quatro situações distintas: sem emoções; com emoções geradas apenas a partir da primeira suposição; com emoções geradas com a ativação das duas primeiras suposições; e o exemplo com todas as suposições ativadas. Os resultados obtidos são apresentados na Figura 6.

Primeiramente, nota-se que à medida que são adicionadas novas suposições ao exemplo, aumentando sua complexidade, o resultado se torna mais imprevisível. O exemplo sem emoçoes não variou seus resultados para as 100 execuções. Com uma suposição, o exemplo variou apenas oito ciclos entre seus resultados mais e menos eficientes. Já o exemplo com duas suposições, possuiu uma variação de 33 ciclos entre seus extremos. Por fim, o exemplo com três suposições possuiu a maior variação com 40 ciclos entre seu ápice de desempenho e seu pior resultado.

Outro ponto a se destacar é que os valores variam em torno de suas médias de desempenho, definidas pelas características de cada exemplo. O teste com apenas uma suposição ruim para o desempenho do agente obteve o pior resultado, com uma média de 138,77 ciclos por execução. O teste com duas suposições circunda uma média de 97,23 ciclos por execução, enquanto a média do teste com três suposições, sendo estas numa proporção de duas motivacionais e uma de desmotivação para o agente, foi de 80,27 ciclos por execução. Estes dois últimos casos, com duas e três suposições apresentam uma característica importante: apesar de suas médias se distanciarem por aproximadamente 17 ciclos, distância bastante grande para um universo de execuções que em várias delas não supera 100 ciclos, têm seus resultados se cruzando em alguns pontos. Isto indica que embora as suposições indicassem que a versão com três suposições possuísse sempre um agente mais eficiente do que o exemplo com duas, a imprevisibilidade da rede da rede Bayesiana de emoções, bem como a aleatoriedade do ambiente fazem com que seja possível o agente no ambiente com três suposições ser pior do que o do exemplo com duas.

6. Conclusão

Observando o funcionamento da rede Bayesiana de emoções é possível realizar algumas afirmações. Primeiramente, que a utilização de uma rede Bayesiana para a aplicação computacional do modelo OCC é possível, permitindo que através da manipulação de probabilidades em suas variáveis aumente-se a similaridade do modelo com a realidade, tendo em vista que o modelo oferece um método de decisão determinístico quanto à quais emoções ocorrem de acordo com determinado evento, o que pode ser modificado em uma rede Bayesiana.

A utilização do modelo OCC sob a forma de uma rede Bayesiana apresenta outra característica importante, permitir uma visualização das relações existentes entre as diferentes emoções que compõe o modelo, relações estas que muitas vezes ficam escondidas em sua estrutura. Por exemplo, ao determinar-se na rede criada a existência da emoção "*Anger*", observa-se um aumento na probabilidade de que a emoção "*Fear*" também ocorra, o contrário de "*Happy-for*", que se torna menos provável.

Embora o exemplo estudado seja simples, foi possível visualizar as características principais da rede Bayesiana de emoções, bem como sua influência em um ambiente multiagentes. Assim, se propõe como trabalhos futuros a aplicação da rede em um exemplo mais elaborado, com diversos agentes possuindo suas próprias redes Bayesianas de emoções. Este exemplo propiciaria mais interações entre os agentes e o ambiente a qual estiverem inseridos e, também, entre si, possibilitando, assim, uma maior quantidade de estímulos à rede e a consequente maior variação nas emoções dos indivíduos, influenciando diretamente seus comportamentos.

Outra possibilidade é que o modelo trabalhado trata apenas estados emocionais, reduzindo os valores individuais das emoções a dois valores gerais, um compreendendo as probabilidades de todas as emoções consideradas boas e outro compreendendo as das emoções ruins. Este modelo pode ser expandido, por exemplo, ao se trabalhar com níveis de estados emocionais. Assim, quando o valor total para o somatório entre os valores positivos e negativos de probabilidade for muito alto, o agente deve ter um comportamento diferente de quando este valor não for tão alto. Ao se definirem faixas de valores para o somatório das emoções é possível que, ao invés de haver apenas três tipos de reação à rede, estável, boa e ruim, possa existir diversas reações para o agente, tantas quantas forem as faixas de valores determinadas.

Também pode-se trabalhar com os valores individuais das emoções, seja avaliando-as apenas individualmente ou em conjunto a um sistema de estados emocionais. Embora bastante complexo, por exigir um conhecimento dos efeitos de cada emoção sobre a maneira como o agente encara as ações que deve realizar, permite que o modelo trabalhado se torne bastante completo e imprevisível, permitindo ao agente possuir diversas reações aos eventos ocorridos no ambiente a qual está inserido.

Referências

Adamatti, D. F. AFRODITE - Ambiente de Simulação Baseado em Agentes com Emoções. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2003.

Bordini, R. H.; Hübner, J. F.; Wooldridge, M. programming multi-agent systems in AgentSpeak using Jason. England: Wiley, 2007.

Cañamero, D.; Van de Velde, W. Emotically Grounded Social Interaction. Human Cognition and Social Agent Tecnology, Amsterdam, 1999.

Conati, C.; Maclaren, H. Empirically Building and Evaluating a Probabilistic Model of User Affect. University of British Columbia. Vancouver. 2010.

Cozman, F. G. Bayesian Networks in Java: User manual and download. JavaBayes, 2001. Disponivel em: <http://www.cs.cmu.edu/~javabayes/Home/index.html>. Acesso em: 9 Abril 2013.

Damásio, A. R. O Mistério da Consciência: do Corpo e das Emoções ao Conhecimento de Si. São Paulo: Companhia das Letras, 2000. 474 p.

Del Nero, H. S. O Sítio da Mente: Pensamento, Emoção e Vontade no Cérebro Humano. São Paulo: Collegium Cognito, 1997. 510 p.

Frozza, R. SIMULA: Ambiente para Desenvolvimento de Sistemas Multiagentes Reativos. Universidade Federal do Rio Grande do Sul. Porto Alegre. 1997.

Gratch, J.; Marselha, S. Modeling Emotions in the Mission Rehearsal Exercice. Conference on computer generated forces and behavioral representation, 2001.

Hruschka Jr., E. R. Imputação Bayesiana no Contexto da Mineração de Dados. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2003.

Moffat, D. C.; Frijda, N. Functional Models of Emotions. Affective Minds, Amsterdam, 2000.

Morgado, A. C. Análise Combinatória e Probabilidade. Rio de Janeiro: [s.n.], 2001.

Ortony, A.; Clore, G.; Collins, A. The Cognitive Structure of Emotions. Cambridge: Cambridge University Press, 1988.

Picard, R. Affective Computing. Cambridge: MIT Press, 1997. 292 p.

Russel, S.; Norvig, P. Artificial Intelligence A Modern Approach. 2a. ed. Upper Saddle River, New Jersey: Prentice Hall, 2003.

Sloman, A. Architectural Requirements for Human-like Agents Both Natural and Artificial (What sorts of machines ca love?). Human Cognition and Social Agent Tecnology, Amsterdam, 1999.

Sloman, A. Beyond Shallow Models of Emotions. COGNITIVE PROCESSING, 2001. 177-198.

Werhli, A. V. Reconstruction of Gene Regulatory Networks from Postgenomic Data. University of Edinburgh. Edinburgh, p. 230. 2007.

Balanceamento de Carga em Redes de Sensores Sem Fio baseado em Sistemas Multiagentes: DSA vs. LA-DCOP vs. Swarm-GAP

Paulo R. Ferreira Jr. [1], **Alexandre Lemke** [1], **Marcelo Giesel** [1], **Paulo A. Afonso** [1] **e Lisane B. Brisolara** [1]

[1]Centro de Desenvolvimento Tecnológico
Universidade Federal de Pelotas (UFPEL)
Pelotas – RS – Brasil

{paulo, alemke, mgiesel, paafonso, lisane}@inf.ufpel.edu.br

Abstract. *Several approaches have been applied in order to extend the useful life time of Wireless Sensor Networks since batteries limit its real application. One of these approaches is based on balancing the load of services ran by the network among its nodes. This paper evaluates the applicability of novel task-assignment algorithms, proposeds by Multiagent Systems community, to achieve that load balancing. We compared DSA, LA-DCOP, Swarm-GAP, and the network without load balancing. Our results show that, despite the high battery consumption due to agents' communication overheads, the general efficiency of the network stills the same as not using load balancing. Thus, the use of these algorithms is promising, but for its greater efficiency is mandatory a lower communication cost.*

Resumo. *Várias técnicas têm sido empregadas para buscar a extensão da vida útil de uma Rede de Sensores Sem Fio dado que a alimentação por baterias limita sua aplicabilidade. Uma das formas de conseguir isso é o emprego do balanceamento de carga dos serviços a serem realizados pela rede entre os nodos que a compõe. Este artigo avalia o emprego de algoritmos de alocação de tarefas em Sistemas Multiagentes, apresentados na literatura recente, para obter o balanceamento de carga mencionado. Foram experimentados os algoritmos DSA, LA-DCOP e Swarm-GAP, sempre comparados com a não utilização de balanceamento. Os resultados obtidos mostraram que, apesar de um alto consumo de bateria para a comunicação entre os agentes, a eficiência da rede permaneceu igual a não utilização do balanceamento de carga. Assim, o emprego dos algoritmos experimentados é promissor, mas para sua maior eficiência é preciso que o custo de comunicação seja bastante reduzido.*

1. Introdução

O projeto e aplicação de Redes de Sensores Sem Fio (RSSF) sofrem restrições dada a limitação no consumo de energia dos nodos que compõem tais redes, os quais são alimentados por baterias. Pesquisas tem buscado metodologias para aumentar a vida útil destas redes, diminuindo o consumo de energia dos nodos, sem perder a qualidade do serviço de sensoriamento prestado por eles. Permitir que os nodos comportem-se de maneira adaptativa para coordenarem-se na execução dos serviços de forma autônoma e *on*

line, de acordo com a dinâmica do ambiente, pode contribuir para o aumento da vida útil da rede.

Uma das abordagens estudadas nessa linha é o balanceamento de carga entre os nodos sensores. Nesta abordagem, um serviço é visto como um conjunto de tarefas com relações de dependência entre elas. As tarefas de um serviço devem ser distribuídas entre os nodos da rede de forma a evitar que uma mesma tarefa seja feita por mais de um nodo e a buscar que as tarefas percebidas sejam distribuídas uniformemente entre estes nodos. Estas estratégias objetivam mapear as tarefas de um serviço aos nodos da rede de maneira que os serviços de sensoriamento tenham sua carga balanceada entre os nodos.

As RSSF têm muitas semelhanças com os SMA (Sistemas Multiagentes). Pode-se enxergar um nodo como um agente e o conjunto de agentes organizados em um sistema como uma rede de nodos. Os agentes, tais quais os nodos, precisam fazer escolhas quanto a realização de tarefas na busca dos objetivos do sistema. Vários trabalhos já se valeram desta semelhança para utilizar técnicas empregadas na solução de problemas em SMA na solução de problemas semelhantes em RSSF [Lesser et al. 2003].

Considerando-se tais estudos, que se valem das semelhanças entre SMA e RSSF para o desenvolvimento de metodologias e técnicas aplicáveis a problemas encontrados em RSSF, acredita-se que seja relevante o estudo do emprego das técnicas de alocação de tarefas mais recentemente propostas na área de SMA para o balanceamento de carga em RSSF.

O Extended Generalized Assignment Problem (E-GAP) [Scerri et al. 2005] vem sendo utilizado para modelar muitos problemas de alocação de tarefas em SMA. O objetivo nesse problema é realizar alocações que maximizem o somatório da eficiência dos agentes em cada instante. A eficiência é a medida da competência do agente de realizar a tarefa que ele está realizando no momento.

Vários algoritmos foram propostos recentemente para a solução deste problema e permitir que um SMA, de maneira distribuída, faça a alocação de tarefas de forma a maximizar o uso dos recursos dos agentes. Neste artigo serão analisados os algoritmos DSA [Zhang and Wittenburg 2002], LA-DCOP [Scerri et al. 2005] e Swarm-GAP [Ferreira Jr. et al. 2007].

Neste artigo, um cenário de balanceamento de carga em Redes de Sensores Sem Fio é modelado como um E-GAP. Os algoritmos mencionados no parágrafo anterior para a solução do E-GAP são executadas em um ambiente de simulação de SMA para que se possa comparar o desempenho de cada um deles.

Os resultados obtidos nos experimentos realizados mostram que os algoritmos de SMA, por dependerem de uma grande quantidade de comunicação, demandaram significativamente mais bateria que a não utilização de nenhuma técnica de balanceamento de carga. No entanto, o LA-DCOP e o Swarm-GAP conseguiram, mesmo consumindo cerca de 10 vezes mais energia, sensorear o mesmo número de eventos. O DSA também consumiu cerca de 10 vezes mais energia mas não foi capaz de se igualar ao número de eventos sensoreados.

Este artigo está organizado da seguinte forma: na Seção 2 são discutidos os trabalhos relacionados na área de RSSF, abordando a forma como a área tem tratado o pro-

blema de balanceamento de carga através da alocação de tarefas; na Seção 3 são apresentadas a definição formal do E-GAP e uma breve descrição de como funcionam os algoritmos DSA, LA-DCOP e Swarm-GAP; a Seção 4 apresenta o cenário de RSSF onde os algoritmos foram experimentados; na Seção 5 são discutidos os resultados obtidos nos experimentos; e, finalmente, na Seção 6 são apresentados as conclusões e direções futuras deste trabalho.

2. Trabalhos Relacionados

O mapeamento entre tarefas de um serviço em RSSF e os nodos da rede que vão realizar este serviço pode ser feito de forma estática [Shivle et al. 2006, Braun et al. 2002, Martinovic et al. 2003], em tempo de projeto da rede, ou de forma dinâmica [Pathak and Prasanna 2010, Zeng et al. 2008], em tempo de execução. Este mapeamento define quais dos nodos vão lidar com as tarefas que compõe o serviço percebido pela rede.

Técnicas de mapeamento estáticas usualmente enfrentam o problema de serem incapazes de lidar com ambientes fortemente dinâmicos em que RSSF podem ser aplicadas. Considerando-se redes heterogêneas como no caso dos trabalhos citados a pouco, este mapeamento também precisa lidar com as diferentes capacidades dos nodos em executar diferentes tarefas.

Além disso, a decisão sobre o mapeamento que será estabelecido pode ser tomada de forma centralizada [AbdelSalam and Olariu 2011], por algum nodo específico que se ponha na posição de coordenador [Caliskanelli et al. 2013], ou por todos os nodos, de maneira distribuída [Uney and Cetin 2007].

As diversas técnicas dinâmicas para o balanceamento de carga enfrentam vários problemas. Um dos mais relevantes destes é a grande necessidade de comunicação entre os nodos para que haja a coordenação sobre o a mapeamento que será realizado. Sabe-se que a comunicação consome significativamente a energia de um nodo sensor.

Os algoritmos de alocação de tarefas em SMA mais atuais nunca foram experimentados para tratar esta questão específica de RSSF. Este trabalho pretende comparar o desempenho dos algoritmos mencionados neste artigo entre si para, em trabalhos futuros, comparar os resultados obtidos com o estado-da-arte na área de RSSF discutidas nesta seção.

3. Fundamentação Teórica

3.1. Alocação de Tarefas

O *Generalized Assignment Problem* (GAP) [Martello and Toth 1990] é um problema geral que trata da alocação de tarefas a agentes, respeitando sua capacidade (recursos que estes têm disponíveis), buscando maximizar uma recompensa total associada às competências destes agentes para tal alocação.

Define-se o GAP como segue. Seja $\mathcal{J}$ o conjunto de tarefas e $\mathcal{I}$ o conjunto de agentes. Cada agente $i \in \mathcal{I}$ tem capacidade r_i associada a uma quantidade limitada de recursos. Um único tipo de recurso é considerado. Quando uma tarefa $j \in \mathcal{J}$ é alocada por um agente i, esta consome u_{ij} unidades do recurso deste agente. Cada agente tem uma competência para alocar cada tarefa j dada por k_{ij} ($0 \leq k_{ij} \leq 1$).

A matriz de alocação A, onde a_{ij} é o valor da i-ésima linha e a j-ésima coluna, é dada pela equação 1.

$$a_{ij} = \begin{cases} 1 & se\ a\ tarefa\ j\ está\ alocada\ pelo\ agente\ i \\ 0 & caso\ contrário \end{cases} \tag{1}$$

O objetivo do GAP é encontrar a matriz A que maximiza a recompensa do sistema dada pela equação 2, respeitando as limitações de recursos e a limitação de ter apenas um agente alocado a cada tarefa.

$$A = argmax_{A'} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} k_{ij} \times a'_{ij} \tag{2}$$

tal que

$$\forall i \in \mathcal{I},\ \sum_{j \in \mathcal{J}} u_{ij} \times a_{ij} \leq r_i \tag{3}$$

e

$$\forall j \in \mathcal{J},\ \sum_{i \in \mathcal{I}} a_{ij} \leq 1 \tag{4}$$

Uma extensão para o GAP, para lidar com problemas dinâmicos e com a interrelação entre tarefas, foi proposta e denominada *Extended* GAP (E-GAP) [Scerri et al. 2005]. O E-GAP estende o GAP de duas maneiras diferentes, as quais são:

Interrelacionamentos entre tarefas. As tarefas no E-GAP podem ser interrelacionadas por restrições do tipo AND. Todas as tarefas interrelacionadas por esta restrição devem ser alocadas simultaneamente para serem consideradas no cálculo da recompensa. Esta é uma extensão importante porque o GAP, por definição, restringe a alocação de cada tarefa a um agente apenas. Quando uma tarefa requer mais recursos que um único agente pode oferecer, esta tarefa deve ser decomposta em subtarefas interrelacionadas pela restrição AND. Define-se $\bowtie = \{\alpha_1, ..., \alpha_k\}$, onde $\alpha_k = \{j_{k_1}, ..., j_{k_q}\}$ denota o k-ésimo conjunto de tarefas interrelacionadas pela restrição AND. Assim, a recompensa local w_{ij} para a alocação da tarefa j ao agente i é dada pela equação 5.

$$w_{ij} = \begin{cases} k_{ij} \times a_{ij} & if\ \forall \alpha_k \in \bowtie,\ j\ \notin \alpha_k \\ k_{ij} \times a_{ij} & if\ \exists \alpha_k \in \bowtie\ with\ j\ \in \alpha_k \wedge \\ & \forall j_{k_q} \in \alpha_k,\ a_{x j_{k_q}} \neq 0\ with\ x \in \mathcal{I} \\ 0 & caso\ contrário \end{cases} \tag{5}$$

Recompensa determinada dinamicamente. A recompensa total W é calculada no E-GAP como o somatório das recompensas parciais w_{ij}^t obtidas em cada intervalo de tempo t. Nesse caso, o que está sendo considerado é a recompensa resultante de uma seqüência de alocações, diferentemente do GAP que considera uma única alocação. Com isso, todos os conjuntos e parâmetros da equação 2 são indexadas pelo tempo. Além disso, um custo v_j pode ser usado para punir os agentes

quando a tarefa j não é alocada no tempo t. O objetivo do E-GAP é maximizar a recompensa total W dada pela equação 6.

$$W = \sum_{t} \sum_{i^t \in \mathcal{I}^t} \sum_{j^t \in \mathcal{J}^t} w_{ij}^t \times a_{ij}^t - \sum_{t} \sum_{j^t \in \mathcal{J}^t} (1 - \sum_{i^t \in \mathcal{I}^t} a_{ij}^t) \times v_j^t \tag{6}$$

tal que

$$\forall t \forall i^t \in \mathcal{I}^t, \sum_{j^t \in \mathcal{J}^t} u_{ij}^t \times a_{ij}^t \leq r_i^t \tag{7}$$

e

$$\forall t \forall j^t \in \mathcal{J}^t, \sum_{i^t \in \mathcal{I}^t} a_{ij}^t \leq 1 \tag{8}$$

Os agentes devem determinar quais tarefas alocar de maneira que suas competências associadas a estas tarefas sejam as maiores possíveis. A quantidade de tarefas que os agentes podem alocar simultaneamente é restrita pela quantidade de recursos que estes têm disponível em relação a quantidade demandada pela tarefas a serem alocadas. A duração d_j destas tarefas é irrelevante para a tomada de decisão pois os agentes precisam definir quais alocar a cada unidade de tempo.

As competências associadas aos agentes e as tarefas por eles alocadas determinam a recompensa do sistema como um todo. As recompensas parciais são obtidas em cada unidade de tempo e a total é o somatório das recompensas parciais durante um intervalo de tempo.

3.2. Algoritmos para Alocação de Tarefas

Os autores que definiram o E-GAP propuseram um algoritmo, denominado LA-DCOP (*Low-communication Aproximation* DCOP) [Scerri et al. 2005], para tratar este problema. O LA-DCOP utiliza um protocolo baseado em *tokens* para aumentar seu desempenho quanto à comunicação. Neste algoritmo os agentes podem perceber tarefas no ambiente ou receber *tokens* de outros agentes.

Cada um dos *tokens* recebidos também contém tarefas, uma cada um. Os agentes decidem ou não alocar uma tarefa baseados em um limiar global, tentando maximizar o uso de seus recursos. Depois de decidir quais tarefas alocar, os agentes enviam *tokens* com tarefas não alocadas para outros agentes.

O limiar mencionado representa a capacidade dos agentes para alocar cada tarefa e pode ser fixo ou dinamicamente calculado. *Tokens* adicionais, chamados "*tokens* potenciais", são utilizados pelos agentes para estabelecer compromissos com relação à alocação de algumas tarefas para lidar com a interrelação de alocação simultânea (AND) que podem existir entre elas.

Os autores do LA-DCOP argumentam que o seu algoritmo é melhor que abordagens anteriormente propostas, comparando-o diretamente com o DSA [Zhang and Wittenburg 2002].

O DSA utiliza uma estratégia baseada em *hill-climbing* para permitir que os agentes, com base nas informações de outros agentes próximos, aloquem as tarefas de modo a maximizar a solução do problema.

Mais recentemente um novo algoritmo, denominado Swarm-GAP [Ferreira Jr. et al. 2007] foi proposto para lidar com o E-GAP. Este algoritmo foi concebido inspirado na forma como as colônias de insetos sociais (*swarms*) dividem o trabalho [Bonabeau et al. 1999].

O modelo de divisão de trabalho dos insetos sociais é baseado em um modelo que usa o estímulo produzido pelas tarefas e um limiar interno específico de cada indivíduo associado a cada tarefa para determinar a tendência de cada indivíduo de realizar cada tarefa [Theraulaz et al. 1998]. Os agentes utilizando o Swarm-GAP decidem quais tarefas realizar baseado neste mecanismo. O Swarm-GAP também utiliza um coeficiente para modificar a tendência do agente em realizar uma tarefa de acordo com a alocação de tarefas interrelacionadas (AND).

Segundo seus autores, o Swarm-GAP tem vantagens em relação à complexidade de tempo se comparado com o LA-DCOP pois seu processo de tomada de decisão é mais eficiente computacionalemente. Por outro lado, os resultados apresentados [Boffo et al. 2007, Ferreira Jr. et al. 2008, Ferreira et al. 2010] mostram algumas vantagens no emprego do LA-DCOP quanto a maximização da solução obtida para o E-GAP em alguns cenários.

4. Descrição do Cenário

O problema em RSSF que este artigo trata, como comentado na Seção 1, diz respeito ao balanceamento de carga visando aumentar a vida útil da rede, a qual é alimentada por baterias, sem a perda de eficiência quanto ao sensoriamento de eventos. Com isso, o cenário abaixo foi elaborado de maneira que fosse possível realizar experimentos com este problema.

A RSSF é composta de um conjunto N de nodos distribuídos aleatoriamente em um espaço 2D toroidal. Cada nodo tem um raio de sensoriamento R igual. Os nodos prestam serviços à uma aplicação que basicamente são a coleta de informações deste espaço, a partir da percepção de um evento, e a realização de alguns cálculos com essa informação.

Os serviços prestados pelos nodos são divididos em tarefas (organização hierárquica), as quais podem ser de sensoriamento ou processamento, definidas de forma abstrata e consumindo a mesma quantidade de energia. Três diferentes serviços são prestados por esta rede, S_a, S_b e S_c.

Para a realização do serviço S_a, o nodo precisa realizar as tarefas a_1 e a_2. A tarefa a_1 é dividida em subtarefas $a_{1.1}$ e $a_{1.2}$ que precisam ser realizadas primeiro que sua supertarefa para habilitar a realização desta. A tarefa a_2 é independente. Da mesma forma, o serviço S_b é composto pelas tarefas b_1, b_2 e b_3, sem subtarefas e o serviço S_c é composto pelas tarefas c_1, c_2 e c_3, tendo a tarefa c_1 subdividida em $c_{1.1}$, $c_{1.2}$, $c_{1.3}$ e $c_{1.4}$. Um nodo pode realizar todo o serviço (todas as subtarefas) ou delegar a realização de todas ou algumas subtarefas para outros nodos.

Um nodo que perceba um evento, o qual demanda a realização de um serviço, é responsável por sua realização. Os eventos que demandam a realização dos serviços ocorrem aleatoriamente com uma probabilidade igual $P(S)$. Um nodo percebe apenas um evento por vez e um evento pode ser percebido por vários nodos.

Os nodos da rede são numerados e interligados em uma rede de comunicação *ad hoc* por proximidade física. Um nodo se liga com todos os nodos que estiverem a um raio D de distância. Apenas o nodo 0 (id = 0) se comunica com a aplicação, ou seja, os nodos precisam enviar uma mensagem para o nodo mais perto de si e que esteja mais perto do nodo 0, e assim por diante. O nodo zero não está conectado a baterias e não reage a eventos de sensoriamento. Além disso, este nodo fica no centro do espaço a ser sensoreado. Essa topologia de comunicação é fixa e definida na inicialização do sistema. Todos os nodos conhecem esta topologia e sabem para quem enviar uma mensagem para que ela chegue na aplicação.

Cada nodo tem uma quantidade de bateria inicial de I unidades. A realização de cada tarefa j consome uma quantidade de energia Q_j. A comunicação da informação referente a uma tarefa j entre os nodos consome uma quantidade de energia $C_j \leq Q_j$. Os serviços devem ser prestados pelos nodos da rede de forma a minimizar o consumo de energia do sistema com um todo e, com isso, aumentar a vida útil do sistema.

Este cenário de RSSF pode ser traduzido facilmente em um E-GAP. As tarefas que compõe os serviços podem ser vistas como as tarefas no E-GAP. Os nodos, que precisam decidir a alocação destas tarefas, são equivalentes aos agentes no E-GAP. Como os nodos são limitados apenas pela sua bateria para realizar as tarefas, pode-se considerar que não existe limitação de recursos para os agentes. Todas as tarefas que compõe um serviço são interrelacionadas (AND) no E-GAP e só são consideradas para o cálculo da recompensa se todas forem alocadas em cada instante.

O objetivo do balanceamento de carga em RSSF é garantir que todos os nodos atuem de forma equânime, ou seja, manter cada nodo o maior tempo possível com a mesma quantidade de bateria dos demais. Com isso, a competência do agente para alocar a tarefa pode ser dado diretamente pela quantidade de bateria do nodo em cada instante. A maximização da recompensa no E-GAP implicará na alocação das tarefas para os agentes com maior quantidade de bateria em cada instante.

5. Experimentos e Resultados

O cenário apresentado na Seção 4 foi implementado na ferramenta NetLogo e foi baseado no cenário utilizado em um recente trabalho relacionado na área de RSSF [Caliskanelli et al. 2013]. Foram feitas algumas adaptações com relação ao cenário referenciado uma vez que no artigo os autores utilizam um simulador de RSSF, que trata diversos detalhes da rede. Os valores iniciais de bateria e seu consumo a cada iteração, por exemplo, foram escolhidos para manter a proporção destes em relação ao consumo para a realização de tarefas que se pode encontrar no referido trabalho. Em contrapartida, o cenário implementado aqui é bem mais complexo no que tange a topologia da rede.

Foram realizadas 30 execuções com cada cada algoritmo. Cada execução termina quando nenhum nodo consegue mais se comunicar com a aplicação hipotética, ou seja, não possui um caminho de comunicação com o nodo especial que serve de *gateway* da rede toda.

Foram considerados como métrica de qualidade de cada algoritmo a quantidade total de eventos sensoriados e a média de bateria dos nodos da rede em cada execução. Foram computadas a média e o desvio padrão das 30 execuções destas duas métricas.

Testes t de student, com confiança de 0.95, foram usados para comparar o desempenho dos algoritmos segundo tais médias.

As simulações foram realizadas com 30 nodos distribuídos aleatoriamente pelo espaço 2d toroidal padrão do NetLogo. Os eventos são objetos que se movem aleatoriamente pelo cenário e são percebidos pelos nodos quando se encontram a uma distância 10 do nodo. Cada evento demanda um e somente um dos serviços descritos na Seção 4 durante toda a simulação e a distribuição inicial de serviços é aleatória.

Os nodos que ficaram distantes menos de 10 unidades, quando estes são distribuídos no espaço, são conectados por um canal de comunicação. Um nodo especial, no centro do espaço e com bateria infinita, é utilizado como *gateway* da rede para a comunicação com uma aplicação hipotética. A comunicação dos nodos com a aplicação se da utilizando *gateways* determinados de forma *ad-hoc*, com cada nodo utilizando como *gateway* o nodo mais próximo do nodo especial (*gateway* da rede com a aplicação) segundo a distância euclidiana. Todos os nodos se comunicam diretamente com seus vizinhos.

Cada nodo inicialmente possui 54000000 unidades de bateria, consumindo 300 por passo de simulação independentemente do que estiverem fazendo. Quando um nodo processa uma tarefa ele consome 6 unidades de bateria e quando este envia uma mensagem são consumidas 2 unidades de bateria.

O simulador busca evitar que mais de um nodo sensoreie um evento ao mesmo tempo. Isso é feito fazendo com que cada nodo verifique se algum de seus vizinhos percebeu o evento antes de decidir por sensoreá-lo. O custo desta comunicação não foi considerado.

Além do DSA, LA-DCOP e Swarm-GAP, nos resultados discutidos a seguir, é considerada uma série de execuções sem que nenhuma técnica de balanceamento de carga seja utilizada. Essa abordagem foi denominada "SemBal" nas tabelas que seguem.

Dada uma limitação da implementação atual dos algoritmos no NetLogo, os eventos foram tratados de forma atômica. Todas as tarefas de um evento são tratadas pelo mesmo agente. O balanceamento então é feito através da alocação das tarefas que compõe o serviço entre os nodos vizinhos ao nodo que decidiu sensorear determinado evento que demanda tal serviço.

Várias simulações com os diferentes parâmetros do DSA foram realizadas para determinar empiricamente a melhor combinação destes para o problema em questão. Com isso, o limiar de probabilidade p para aceitação de um novo Δ foi fixado em $p = 0.2$ e foram adotadas 5 rodadas para a parada do algoritmo. A utilidade U foi computada como sendo $U = e^{(\frac{(Q_i)}{I} * 10}$, onde Q_i é a quantidade atual de bateria do nodo e I a quantidade inicial de bateria dos nodos.

O LA-DCOP e o Swarm-GAP também foram configurados empiricamente. Em ambos os casos, o limiar interno do agente i para a realização de uma determinada tarefa j é calculado como sendo $\theta_{ij} = 1 - \frac{\sum^{j \in S} Q_j}{Q_i}$, onde Q_j é a quantidade de bateria demandada pelas tarefas que compõe o serviço completo e Q_i é o total de bateria do nodo. No LA-DCOP o limiar foi fixado em 0.6 e no Swarm-GAP o estímulo S_j é dado pela quantidade de bateria que as tarefas consomem Q_j.

Como mostrado na Tabela 1, a não utilização de técnicas de balanceamento de carga necessariamente implica em uma quantidade média maior de bateria disponível todo o tempo de cada execução. A média da quantidade de bateria disponível utilizando os algoritmos de SMA foi cerca de 10 vezes menor em média nos experimentos realizados. Esse comportamento era esperado uma vez que a alocação de tarefas em SMA necessariamente é realizada através de muitas trocas de mensagens.

Tabela 1. Média de bateria dos nodos da rede utilizando os algoritmos experimentados.

	SemBal	DSA	LA-DCOP	Swarm-GAP
Média	783057351,5	65971973,5	73640156,7	77173282,6
Desvio Padrão	23248782,6	173921138,3	233104604,9	323160163,4

No entanto, mesmo com um consumo de bateria bem maior, os algoritmos LA-DCOP e Swarm-GAP foram capazes de sensorear a mesma quantidade de eventos que a não utilização de técnicas de balanceamento, como mostra a Tabela 2. Apenas o DSA que apresentou diferença significativa em relação aos eventos sensoreados, sendo menos eficiente cerca de 6% menos eficiente nos experimentos realizados.

Tabela 2. Número total de eventos sensoriados utilizando os algoritmos experimentados.

	SemBal	DSA	LA-DCOP	Swarm-GAP
Média	801757,6	754783,1	779909,6	789694,1
Desvio Padrão	29884,9	70130,4	99945,9	33409,60381

Levando em conta os resultados obtidos, é promissora a aplicação de algoritmos de alocação de tarefas em SMA para o balanceamento de carga em RSSF. Contudo, o consumo de bateria para a comunicação precisa ser bem menor se comparado ao consumo de bateria pelo processamento das tarefas para que o emprego de tais algoritmos tenha um efeito positivo real em RSSF. Nos experimentos realizados o consumo de bateria para a comunicação é apenas $\frac{1}{3}$ do consumo de bateria do processamento das tarefas.

6. Conclusões

Este artigo apresentou um estudo sobre a aplicação de algoritmos para a alocação de tarefas em SMA no problema de balanceamento de carga em RSSF. O objetivo foi de verificar se tais algoritmos podem contribuir para a construção de RSSF com maior vida útil, sem que sua eficiência quanto ao sensoriamento seja prejudicada.

Para tanto, um cenário de RSSF foi especificado e implementado em um simulador de SMA. Os algoritmos DSA, LA-DCOP e Swarm-GAP foram implementados no simulador para lidar com tal cenário. As métricas adotadas foram o total de eventos sensoreados pela rede e a média de bateria presente na rede toda durante cada execução.

Os resultados mostraram que, apesar de consumir uma quantidade de bateria significativamente maior dada a necessidade de estabelecer muita comunicação, os algoritmos LA-DCOP e Swarm-GAP foram capazes de sensorear a mesma quantidade de eventos que a rede sem o uso de balanceamento de carga. Isso mostra que estes algoritmos podem

ser úteis se o consumo de bateria para a comunicação for significativamente mais baixo que o consumo de bateria do processamento dos serviços.

Como trabalhos futuros pretende-se realizar experimentos com um custo de comunicação menor que o utilizado neste artigo, bem como implementar e experimentar o estado-da-arte na área de balanceamento de carga em RSSF para comparar com o desempenho dos algoritmos utilizados neste artigo.

Referências

AbdelSalam, H. and Olariu, S. (2011). Toward efficient task management in wireless sensor networks. *Computers, IEEE Transactions on*, 60(11):1638–1651.

Boffo, F., Jr., P. R. F., and Bazzan, A. L. C. (2007). A comparison of algorithms for task allocation in robocup rescue. In *Proceedings of the 5th European Workshop on Multiagent Systems*. To appear.

Bonabeau, E., Theraulaz, G., and Dorigo, M. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, USA.

Braun, T., Siegel, H., and Maciejewski, A. (2002). Static mapping heuristics for tasks with dependencies, priorities, deadlines, and multiple versions in heterogeneous environments. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 8 pp–.

Caliskanelli, I., Harbin, J., Indrusiak, L. S., Mitchell, P., Polack, F., and Chesmore, D. (2013). Bioinspired load balancing in large-scale wsns using pheromone signalling. *International Journal of Distributed Sensor Networks*, 2013.

Ferreira, Jr., P. R., dos Santos, F., Bazzan, A. L. C., Epstein, D., and Waskow, S. J. (2010). Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. 20(3):421–443.

Ferreira Jr., P. R., Boffo, F., and Bazzan, A. L. C. (2007). A swarm based approximated algorithm to the extended generalized assignment problem (E-GAP). In *Proceedings of the 6th International Joint Conference on Autonomous Agents And Multiagent Systems (AAMAS)*, pages 1231–1233.

Ferreira Jr., P. R., Boffo, F., and Bazzan, A. L. C. (2008). Using Swarm-GAP for distributed task allocation in complex scenarios. In Jamali, N., Scerri, P., and Sugawara, T., editors, *Massively Multiagent Systems*, Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin. To appear.

Lesser, V., Ortiz, C. L., and Tambe, M. (2003). *Distributed sensor networks: A multiagent perspective*, volume 9. Springer.

Martello, S. and Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc.

Martinovic, G., Budin, L., and Hocenski, Z. (2003). Static-dynamic mapping in heterogeneous computing environment. In *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS '03. 2003 IEEE International Symposium on*, pages 32–37.

Nwana, H. S., Lee, L. C., and Jennings, N. R. (1996). Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4):79–88.

Pathak, A. and Prasanna, V. (2010). Energy-efficient task mapping for data-driven sensor network macroprogramming. *Computers, IEEE Transactions on*, 59(7):955–968.

Scerri, P., Farinelli, A., Okamoto, S., and Tambe, M. (2005). Allocating tasks in extreme teams. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 727–734, New York. ACM Press.

Shivle, S., Siegel, H. J., Maciejewski, A. A., Sugavanam, P., Banka, T., Castain, R., Chindam, K., Dussinger, S., Pichumani, P., Satyasekaran, P., Saylor, W., Sendek, D., Sousa, J., Sridharan, J., and Velazco, J. (2006). Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment. *J. Parallel Distrib. Comput.*, 66(4):600–611.

Theraulaz, G., Bonabeau, E., and Deneubourg, J. (1998). Response threshold reinforcement and division of labour in insect societies. *Royal Society of London Series B - Biological Sciences*, 265:327–332.

Uney, M. and Cetin, M. (2007). Graphical model-based approaches to target tracking in sensor networks: An overview of some recent work and challenges. In *Image and Signal Processing and Analysis, 2007. ISPA 2007. 5th International Symposium on*, pages 492–497.

Zeng, Z., Liu, A., Li, D., and Long, J. (2008). A highly efficient dag task scheduling algorithm for wireless sensor networks. In *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, pages 570–575.

Zhang, W. and Wittenburg, L. (2002). Distributed breakout revisited. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 352–357. Menlo Park, CA: American Association for Artificial Intelligence.

Monitoramento de SMA Normativos – Uma Implementação para Agentes Reativos Simples em JAMDER 2.0

Francisco I. S. Cruz , Emmanuel S. S. Freire, Mariela I. Cortés e Nécio L. Veras

Grupo de Engenharia e Sistemas Inteligentes (GESSI)
Departamento de Computação – Universidade Estadual do Ceará (UECE)
Avenida Paranjana, 1700 – 60740-000 – Fortaleza – CE – Brazil

{israel.santos.cr, savio.essf, necioveras}@gmail.com,
mariela@larces.uece.br

Abstract. *Norms in multi-agent systems are used in order to regulate the behavior of agents to archive the system global goal. For this, the existence of a mechanism that allows to determine the performance of the agents in relation of the fulfillment of the system norms is essential. The present paper presents the implementation, in JAMDER 2.0, of an abstract architecture that allows the monitoring of the agent's behavior in relation to the fulfillment or the violation of norms. A case study considering the simple reactive agents in the vacuum cleaner world is presented to illustrate the approach of enforcement was implemented in this paper.*

Resumo. *Normas em sistemas multi-agente são utilizadas para regular o comportamento dos agentes de forma a alcançar os objetivos globais do sistema. Para isso, é crucial a existência de um mecanismo que permita determinar o desempenho dos agentes em relação ao cumprimento das normas especificadas no sistema. O presente artigo apresenta a implementação, em JAMDER 2.0, de uma arquitetura abstrata que possibilita o monitoramento do comportamento dos agentes em relação ao cumprimento ou violação de normas. Um estudo de caso considerando agentes reativos simples para o mundo do aspirador de pó é apresentado para ilustrar a abordagem de coesão (enforcement) implementada neste artigo.*

1. Introdução

Um sistema multi-agente (SMA) pode ser interpretado como uma sociedade onde entidades autônomas e heterogêneas podem cooperar ou competir para alcançar seus objetivos. A fim de lidar com a heterogeneidade nestes sistemas, mecanismos de governança são definidos por meio de um conjunto de normas que guiam o comportamento das entidades do sistema. Quando normas são inseridas para restringir as ações dos agentes por meio dos conceitos deônticos de obrigação, proibição e permissão [Figueiredo; Silva 2010], caracteriza-se um sistema multi-agente normativo (SMAN).

As normas são utilizadas para regular e coordenar o comportamento dos agentes [Modgil et al. 2009] e influenciam nos processos de tomada de decisão, no comportamento racional e, consequentemente, no desempenho do agente. As normas podem ser interpretadas como padrões de comportamento [Dybalova et al. 2013].

Portanto, é necessário observar o comportamento dos agentes diante do cumprimento e/ou violação das normas que regem o ambiente que tais agentes estão inseridos. Neste contexto, existem abordagens para implementar normas em SMAN. As principais são: regimentação (*regimentation*) [Jones; Sergot 1993] e coesão (*enforcement*) [Dastani et al. 2008].

Na regimentação, o comportamento do agente é restringido pelas normas, diminuindo a sua autonomia, pois o agente só pode executar as atividades permitidas pelas normas, eliminando consequentemente estados ou comportamentos proibidos pelas normas. Em contrapartida, as normas definidas na coesão podem influenciar o comportamento do agente informando as ações que podem, devem ou não podem ser executadas. Com isso, a última abordagem pode ser considerada mais flexível uma vez que o agente não é impedido de violar as normas, no entanto requer de um mecanismo normativo adicional para o monitoramento do comportamento dos agentes em relação ao cumprimento ou não das normas.

Em Modgil et al. (2009), é apresentada uma arquitetura genérica para a observação do comportamento dos agentes em sistemas multi-agente normativos, de forma a determinar se estão cumprindo ou violando as normas estabelecidas. Esta arquitetura prevê monitores que recebem informações de observadores, e processam estas informações através de redes de transição de normas individuais, de forma a determinar o cumprimento ou a violação de tais normas.

Em contrapartida, o *framework* JAMDER 2.0 [Cruz et al. 2014] permite a implementação de todos os elementos que compõem os sistemas multi-agente normativos, porém não contém a abordagem de coesão. O presente artigo implementa a abordagem de coesão utilizando o *framework* JAMDER 2.0 para o caso de agentes reativos simples e avalia o comportamento desses agentes em um sistema a partir de uma simulação para o mundo do aspirador de pó. Este trabalho está organizado da seguinte maneira: a Seção 2 apresenta os conceitos relacionados com o monitoramento em sistema multi-agente normativos. Em seguida, na Seção 3, é descrita a implementação da abordagem coesão utilizando o *framework* JAMDER 2.0. Um estudo de caso utilizando agente reativos simples no mundo do aspirador de pó é apresentado na Seção 4 e, finalmente, as conclusões e trabalhos futuros são descritos na Seção 5.

2. Monitoramento de Sistemas Multi-agente Normativos

A abordagem de coesão ou *enforcement* pode ser efetivamente utilizada para regular o comportamento dos agentes em um sistema multi-agente normativo desde que seja fornecido um mecanismo de monitoramento que possibilite o reconhecimento do cumprimento ou não das normas no ambiente [Modgil et al. 2009], de forma a viabilizar a aplicação de sanções de punição ou recompensa.

A abordagem para monitoramento utilizada no *framework* de Modgil et al. (2009) é baseada na técnica de *overhearing* [Kaminka; Pynadah; Tambe 2012], que consiste em observar a troca de mensagens entre os agentes com o objetivo de inferir o comportamento dos mesmos, em contrapartida a abordagens intrusivas [Jennings 1995] [Mazouzi; Seghrouchni; Haddad 2002] [Tambe 1997] as quais pressupõem que os estados mentais dos agentes são acessíveis à inspeção.

Em resumo, a técnica contempla a existência dos seguintes aspectos em relação ao monitoramento de comportamentos de agentes governados por normas:

- Modelo observador das mensagens dos agentes e demais estados de interesse de forma a prover subsídios para a aplicação adequada dos mecanismos de coesão.
- Normas são individualmente representadas por ATNs (*Augmented Transition Network,* um grafo que representa os três estados que uma norma pode assumir) independentes utilizadas para o monitoramento de comportamentos complexos de agentes em conjunto.
- Monitoramento de normas em sistemas multi-agente dinâmicos e abertos.

De acordo com a abordagem apresentada em Modgil et al. (2009), os agentes são tratados como caixas pretas e as transições de estados internos são invisíveis ao monitor. Um *mapper* mapeia as normas para representações ATN, as quais irão alimentar o processamento do monitor. Em tempo de execução, o monitor reporta ao mecanismo de observação sobre os estados ou ações de interesse identificados a partir dos componentes das normas.

Ainda em tempo de execução, o mecanismo de observação notifica o monitor sempre que for requerida a aplicação de uma sanção. Esse mecanismo considera a observação do comportamento dos agentes em relação aos estados e ações sendo regulamentados pelas normas segundo estabelecido anteriormente.

O monitor processa esta informação levando em conta a representação da ATN da norma para determinar se a norma se encontra ativa, foi atendida ou violada, ou expirou. Finalmente, o monitor sinaliza sobre a aplicação de uma sanção a um determinado agente, se for apropriado.

3. Implementação do *Framework* de Monitoramento em JAMDER 2.0

O *framework* JAMDER 2.0 contempla todos os elementos típicos que compõem um sistema multi-agente normativo. Entretanto, esse *framework* não possui um mecanismo para verificar se os agentes cumpriram ou não as normas definidas no contexto de uma organização, suborganização ou ambiente.

O conceito de normas utilizado em JAMDER 2.0 se refere ao monitoramento das ações dos agentes de forma a guiar a execução de ações. Com isso, as normas definidas em JAMDER 2.0 não garantem que as ações proibidas não sejam executadas nem que as ações obrigatórias sejam executadas. Portanto, a abordagem de coesão é mais adequada ao *framework*.

A solução implementada em JAMDER 2.0 contempla um agente de monitoramento, definido a partir da classe abstrata *jamder.monitoring.Monitor* que herda de jamder.agents.*GenericAgents* e que possui o método *percept(Object, Object)*. Esse método é chamado pelo agente vinculado ao monitor (instância de *jamder.monitoring.Monitor*) sempre que ele executa uma ação. Dessa forma, torna-se possível o monitoramento dos agentes considerando as normas do ambiente.

A classe *jamder.monitoring.Monitor* possui como atributos um conjunto de agentes (monitorados) e as ATNs que são geradas por meio das normas que restringem os agentes que são adicionados, ou quando uma nova norma passa a restringir um agente

que está sendo monitorado. A definição do comportamento do agente de monitoramento fica a cargo do usuário. Isso é importante, pois a literatura costuma divergir sobre como e quando as sanções devem ser aplicadas ao agente monitorado [Piunti et al., 2010]. A estrutura de monitoramento no *framework* JAMDER 2.0 é ilustrada na Figura 1.

Adicionalmente, o monitoramento em um sistema normativo requer uma abstração para representar os estados possíveis de uma norma. Para isso, utilizam-se *Augmented Transition Networks* (ATNs) [MODGIL et al. 2009]. Uma ATN é um grafo que representa os três estados que uma norma pode assumir. São eles: *INACTIVE*, *ACTIVE* e *COMPLIANCE-OR-INFRINGEMENT*. Para incorporar esse conceito em JAMDER 2.0, foram criados a classe *jamder.monitoring.ATN* e o *enumeration jamder.monitoring.ATNState* que representam as ATNs e seus estados, respectivamente. O *enumeration* é do tipo *String* e contém os três estados de uma ATN.

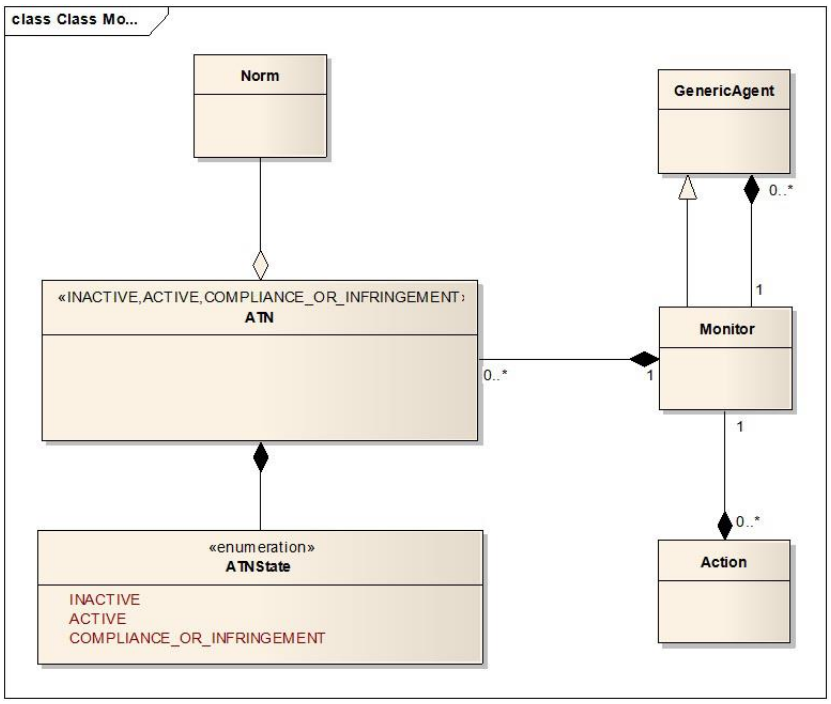


Figura 1. Estrutura de monitoramento no JAMDER 2.0

3.1 Agente monitor

As principais funções do agente de monitoramento são: (i) avaliar se um agente cumpriu ou violou uma norma e (ii) aplicar as sanções relacionadas às normas violadas ou cumpridas, quando apropriado.

A classe *jamder.monitoring.Monitor* (Figura 1) possui os seguintes métodos:

- **getAgents()** - que retorna os agentes que estão sendo monitorados pelo monitor;
- **addAgent(String, GenericAgent)** - que é utilizado para adicionar um agente que será monitorado;
- **addATN(Norm)** - adiciona um ATN por meio de uma norma. Este método é usado no método apply() da classe jamder.norms.Norm. Quando uma norma passa a

restringir um agente monitorado, uma nova ATN é criada para monitorar esse agente;

- **removeATN(String)** - remove uma ATN por meio de sua chave correspondente. Este método é usado no método disapply() da classe jamder.norms.Norm. Quando uma norma deixa de restringir um agente monitorado, a ATN correspondente é removida;
- **getAllAtns()** - que retorna a Hashtable que contém as ATNs vinculadas às normas que restringem os agentes que estão sendo monitorados;
- **punish(Norm)** - que aplica a punição (se ela existir) vinculada à norma que recebe como parâmetro;
- **reward(Norm)** - que aplica a recompensa (se ela existir) vinculada à norma que recebe como parâmetro.
- **percept(Object, Object)** - um método abstrato que é uma implementação do padrão GoF *Observer* [Vlissides et al. 1995]. Quando uma ação é executada, o agente envia um sinal para seu monitor através deste método.

A Figura 2 apresenta a classe abstrata do agente de monitoramento.

4. Estudo de caso

Nesta seção, é apresentado um estudo de caso baseado no Mundo do Aspirador de Pó Normativo. Tanto os agentes aspirador de pó como o ambiente foram implementados utilizando JAMDER 2.0, contemplando o agente monitor descrito na Seção 3. Este estudo de caso foi utilizado por Campos, Freire e Cortés (2012) para ilustrar sua abordagem.

Considerando um Mundo do Aspirador de Pó Normativo com somente duas salas, onde cada sala pode estar limpa ou suja, em nossos experimentos, a informação percebida do ambiente foi representada por estados do ambiente. Adicionalmente, o mundo possui normas que restringem o comportamento dos agentes para executar suas atividades. O Mundo do Aspirador de Pó Normativo é composto por:

- ***Place*** - um objeto que é um grafo de dois vértices. Estes vértices denotam a sala: *roomA* e *RoomB*;
- ***CleanserOrg*** - uma organização que possui os seguintes papéis de agente: *management e cleaner. O* primeiro é associado ao *ManagerAgent* (agente de monitoramento) e o outro é associado ao *VacuumCleaner* (agente aspirador);
- ***ManagerAgent*** **-** um agente de monitoramento que utiliza o papel de management na organização *CleanserOrg*. Sua função é monitorar o agente *VacuumCleaner* e modificar seu comportamento de acordo com as normas do ambiente. Suas ações são: *ActionMonitorCyclic* e *ActionMonitorReflex*. A implementação do agente é apresentada na Figura 3;
- ***NormativeVacuumCleaner*** **-** um agente reativo simples normativo (Campos, Freire and Cortés 2012) que utiliza o papel *cleaner* na organização

CleanserOrg. Suas ações são: *Right*, *Left*, *Suck* e sua percepção é *NextAction*. A implementação do agente é apresentada na Figura 4;

- ***VacuumCleaner*** - um agente reativo simples que utiliza o papel *cleaner* na organização *CleanserOrg*. Suas ações são: *Right*, *Left*, *Suck* e sua percepção é *NextAction*. A implementação do agente é apresentada na Figura 4.

```
public abstract class Monitor extends GenericAgent {
    private static final long serialVersionUID = 1L;
    private Hashtable<String, ATN> atns=new Hashtable<String, ATN>();
    private Hashtable<String, GenericAgent> agents=new Hashtable<String, GenericAgent>();


    public Monitor(String name, Environment environment, Organization owner) {
        super(name, environment, null);
        if (owner!=null && owner.getName()!=null)
            addOrganization(owner.getName(), owner);
    }

    public Hashtable<String, GenericAgent> getAgents() {
        return agents;
    }

    public void addAgent(String key, GenericAgent agent) {
        this.agents.put(key, agent);
        for (Norm nor: agent.getAllRestrictNorms().values()){
            ATN atn=new ATN(nor, nor.getName());
            atns.put(atn.getName(), atn);
        }
    }

    public void addATN(Norm nor){
        if (!atns.containsKey(nor.getName())){
            ATN atn=new ATN(nor, nor.getName());
            atns.put(atn.getName(), atn);
        }
    }
    public void removeATN(String key){
        atns.remove(key);
    }

    public Hashtable<String, ATN> getAllAtns() {
        return atns;
    }

    public void punish(Norm norm){
        Hashtable<String, Norm> punishments=norm.getSactionPunishment();
        for (Norm punishment: punishments.values()){
            if (punishment.isApply())
                punishment.disapply();
            Object context= norm.getContext();
            Object restrict=norm.getRestrict();
            punishment.setContext(context);
            punishment.setRestrict(restrict);
            punishment.apply();
        }
    }

    public void reward(Norm norm){
        Hashtable<String, Norm> rewards=norm.getSactionReward();
        for (Norm reward: rewards.values()){
            if (reward.isApply())
                reward.disapply();
            Object context= norm.getContext();
            Object restrict=norm.getRestrict();
            reward.setContext(context);
            reward.setRestrict(restrict);
            reward.apply();
        }
    }


    public abstract void percept(Object perception1, Object perception2);


}
```

Figura 2. Classe abstrata do agente de monitoramento.

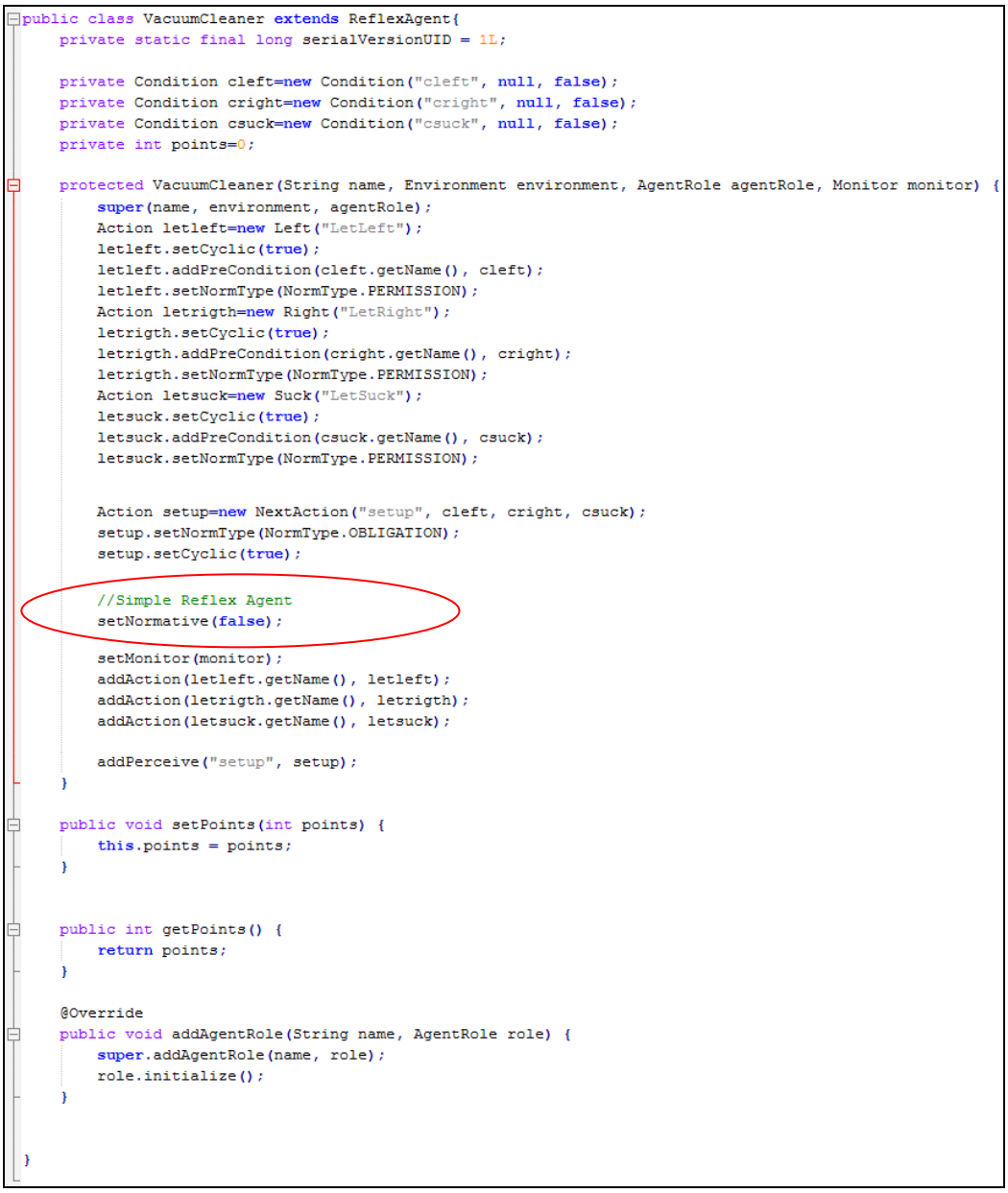
Adicionalmente, o mundo possui as seguintes normas para restringir o comportamento dos agentes para executar suas atividades:

- **N1** – O agente é obrigado a limpar a *roomB*;
- **N2** – O agente é obrigado a limpar a *roomA*. Sanção de recompensa da norma N1. Quando a norma N1 é cumprida, N2 é aplicada ao agente que nesse momento ganha 3 pontos por cumprir a norma N1.

No início de cada experimento, tanto o ***NormativeVacuumCleaner*** como o ***VacuumCleaner*** não conhecem a configuração do mundo em termos de salas sujas. Nós consideramos que quando o mundo está sem a presença de normas, a medida de desempenho oferece uma recompensa de um ponto para cada sala limpa (+1) e penaliza com a perda de um ponto para cada movimento (-1). No caso da presença de normas no mundo, a medida de desempenho precisa ser adaptada para considerar as recompensas (+pontos) e as penalidades (-pontos), que são consequências da escolha do agente em seguir ou rejeitar alguma norma.

```
public class ManagerAgent extends Monitor {


    private static final long serialVersionUID = 5670776386732896828L;

    public ManagerAgent(String name, Environment environment, Organization owner) {
        super(name, environment, owner);
        Action ac=new ActionMonitorCyclic("ActionMonitorCyclic");
        addAction("ActionMonitorCyclic",ac);

        //------Definição de papel do agente de monitoramento-----
        AgentRole ar=new AgentRole("management", owner, this);
        ar.addAction("ActionMonitorCyclic",ac);

        Calendar date1a=new GregorianCalendar(2014, GregorianCalendar.JANUARY, 29, 15, 8, 20);
        Calendar date2b=new GregorianCalendar(2014, GregorianCalendar.MARCH, 29, 15, 9, 40);
        NormConstraint cond=new Between(date1a, date2b);
        Hashtable<String, NormConstraint> constraints=new Hashtable<String, NormConstraint>();
        constraints.put("cond", cond);

        NormResource nre= new NormResource(ac);
        NormAction noa= new AtomicAction(AtomicActionType.AtomicExecute, nre);

        Norm no=new Norm("N1", NormType.OBLIGATION, ar, owner, noa, constraints);
        ar.addRestrictNorm(no.getName(), no);

        //Inicialização
        addAgentRole("management", ar);
        ar.initialize();
    }

    @Override
    public void percept(Object perception1, Object perception2) {
        if (perception1 instanceof ReflexAgent && perception2 instanceof Action){
            addBehaviour(new ActionMonitorReflex("ActionMonitor", (ReflexAgent)perception1, (Action)perception2));
        }
    }
}
```

Figura 3. ManagerAgent.



```
public class VacuumCleaner extends ReflexAgent{
    private static final long serialVersionUID = 1L;

    private Condition cleft=new Condition("cleft", null, false);
    private Condition cright=new Condition("cright", null, false);
    private Condition csuck=new Condition("csuck", null, false);
    private int points=0;

    protected VacuumCleaner(String name, Environment environment, AgentRole agentRole, Monitor monitor) {
        super(name, environment, agentRole);
        Action letleft=new Left("LetLeft");
        letleft.setCyclic(true);
        letleft.addPreCondition(cleft.getName(), cleft);
        letleft.setNormType(NormType.PERMISSION);
        Action letrigth=new Right("LetRight");
        letrigth.setCyclic(true);
        letrigth.addPreCondition(cright.getName(), cright);
        letrigth.setNormType(NormType.PERMISSION);
        Action letsuck=new Suck("LetSuck");
        letsuck.setCyclic(true);
        letsuck.addPreCondition(csuck.getName(), csuck);
        letsuck.setNormType(NormType.PERMISSION);


        Action setup=new NextAction("setup", cleft, cright, csuck);
        setup.setNormType(NormType.OBLIGATION);
        setup.setCyclic(true);

        //Simple Reflex Agent
        setNormative(false);

        setMonitor(monitor);
        addAction(letleft.getName(), letleft);
        addAction(letrigth.getName(), letrigth);
        addAction(letsuck.getName(), letsuck);

        addPerceive("setup", setup);
    }

    public void setPoints(int points) {
        this.points = points;
    }


    public int getPoints() {
        return points;
    }

    @Override
    public void addAgentRole(String name, AgentRole role) {
        super.addAgentRole(name, role);
        role.initialize();
    }


}
```

Figura 4. Quando o atributo *setNormative* é *true*, tem-se o NormativeVacuumCleaner. Caso contrário, tem-se o *VacuumCleaner*.

5. Experimentos

Uma vez definidas as entidades do estudo de caso seguiram-se os experimentos nos quais, incluímos os agentes *VacuumCleaner* e *NormativeVacuumCleaner* em mundos específicos com a presença ou não de normas.

5.1 *VacuumCleaner* em ambiente com normas inativas

Primeiramente, o agente *VacuumCleaner* foi inserido em um Mundo Aspirador de Pó sem normas. A Tabela 1 apresenta os resultados deste cenário. Observa-se que, como

não há normas, a pontuação só decresce quando o agente de desloca de uma sala a outra e cresce quando ele limpa uma sala suja.

Tabela 1 - Execução do agente *VacuumCleaner* em um ambiente com normas inativas.

	Estado			Ação	Pontos
	Onde está?	Estado roomA	Estado roomB		
1	*roomA*	Sujo	Sujo	*suck*	1
2	*roomA*	Limpo	Sujo	*right*	0
3	*roomB*	Limpo	Limpo	*suck*	1
4	*roomA*	Limpo	Limpo	*left*	0
5	*roomA*	Limpo	Limpo	*right*	-1
6	*roomB*	Limpo	Limpo	*left*	-2
7	*roomA*	Limpo	Limpo	*right*	-3

5.2 VacuumCleaner em ambiente com normas ativas

Em seguida, o agente *VacuumCleaner* foi inserido em um Mundo do Aspirador de Pó Normativo com a norma N1 ativa. A Tabela 2 apresenta os resultados desse cenário.

Tabela 2 - Execução do agente *VacuumCleaner* em um ambiente com normas ativas.

	Estado			Ação	Pontos
	Onde está?	Estado roomA	Estado roomB		
1	*roomA*	Sujo	Sujo	*suck*	1
2	*roomA*	Limpo	Sujo	*right*	0
3	*roomB*	Limpo	Limpo	*suck*	4
4	*roomA*	Limpo	Limpo	*left*	3
5	*roomA*	Limpo	Limpo	*right*	2
6	*roomB*	Limpo	Limpo	*left*	1
7	*roomA*	Limpo	Limpo	*right*	0

As linhas 1 e 2 descrevem o comportamento do agente no período que a norma está inativa. A linha 3 da tabela (destacada) ilustra o comportamento do agente quando a norma N1 está ativa, informando que o agente foi recompensado com três pontos por ações executadas durante esse período juntamente com um ponto por ter limpado a sala. Nas linhas 4 a 7, as normas foram desativadas.

5.3. NormativeVacuumCleaner em ambiente com normas inativas

Primeiramente, o agente *NormativeVacuumCleaner* foi inserido em um Mundo Aspirador de Pó sem normas. A Tabela 3 apresenta os resultados deste cenário. Observa-se que, como não há normas, a pontuação só decresce quando o agente de desloca de uma sala a outra e cresce quando ele limpa uma sala suja. O mesmo resultado é apresentado pelo VacuumCleaner quando inserido no mesmo ambiente com normas inativas.

Tabela 3 - Execução do agente *NormativeVacuumCleaner* em um ambiente com normas inativas.

	Estado			Ação	Pontos
	Onde está?	Estado roomA	Estado roomB		
1	*roomA*	Sujo	Sujo	*suck*	1
2	*roomA*	Limpo	Sujo	*right*	0
3	*roomB*	Limpo	Limpo	*suck*	1
4	*roomA*	Limpo	Limpo	*left*	0
5	*roomA*	Limpo	Limpo	*right*	-1
6	*roomB*	Limpo	Limpo	*left*	-2
7	*roomA*	Limpo	Limpo	*right*	-3

5.4. NormativeVacuumCleaner em ambiente com normas ativas

Em seguida, o agente *NormativeVacuumCleaner* foi inserido em um Mundo do Aspirador de Pó Normativo com a norma N1 ativa. A Tabela 4 apresenta os resultados desse cenário.

Tabela 4 - Execução do agente *NormativeVacuumCleaner* em um ambiente com normas ativas.

	Estado			Ação	Pontos
	Onde está?	Estado roomA	Estado roomB		
1	*roomA*	Sujo	Sujo	*suck*	1
2	*roomA*	Limpo	Sujo	*right*	0
3	*roomB*	Limpo	Limpo	*suck*	4
4	*roomA*	Limpo	Limpo	*suck*	3
5	*roomA*	Limpo	Limpo	*suck*	4
6	*roomA*	Limpo	Limpo	*suck*	5
7	*roomA*	Limpo	Limpo	*suck*	6

A norma N1 está ativa entre as linhas 1 e 3. Na linha 3, o agente monitor aplica a sanção de recompensa vinculada à norma N1 que é a norma N2. A partir da linha 4, a única norma ativa é N2. Através da aplicação de uma sanção, ou seja, a partir momento que a norma N2 foi aplicada ao agente (N2 passou a restringi-lo), o gente alterou seu comportamento enquanto a norma N2 esteve ativa. Uma modificação no ambiente por parte do monitor atuou redirecionando esse agente.

6. Conclusão e Trabalhos Futuros

A influência das normas na tomada de decisão dos agentes é essencial para regular e monitorar o comportamento de tais agentes quando inseridos em ambientes normativos. Neste contexto, esse artigo apresenta uma adaptação do *framework* JAMDER 2.0 para possibilitar a implementação de um agente de monitoramento seguindo os conceitos relacionados na abordagem de coesão ou *enforcement*. Adicionalmente, um exemplo baseado no Mundo do Aspirador de Pó Normativo foi utilizado para ilustrar a utilização da adaptação do JAMDER 2.0, demonstrando sua aplicabilidade para o desenvolvimento de sistemas multi-agente normativos.

Por meio do estudo de caso, pode-se perceber que os agentes reativos normativos quando inseridos em ambientes que não possuem normas ativas, se comportam como agentes reativos normais. Entretanto, quando inseridos em ambientes com normas ativas, a coesão (*enforcement*) garante que os agentes terão seu comportamento monitorado por alguma entidade do sistema. Com isso, a adaptação de JAMDER 2.0 permite a comparação do desempenho dos agentes quando inseridos ou não em ambientes normativos. Adicionalmente, o agente de monitoramento pode ser estendido para ser utilizado em novos cenários incluindo outras arquiteturas de agentes, como por exemplo, a arquitetura do agente baseado em modelos normativo [Campos et al. 2013] [Freire et al. 2013].

Como trabalhos futuros, pode-se relacionar: (i) a utilização do *framework* juntamente com o agente monitor em cenários que permitam a inclusão de normas de forma dinâmica, e (ii) a geração automática de código para os agentes reativos simples normativos e de monitoramento, baseada na adaptação de JAMDER 2.0 proposta nesse trabalho.

References

Campos, G. A.; Freire, E. S. S. and Cortés, M. I. (2012) Norm-based behavior modification in reflex agents. In: 14th International Conference on Artificial Intelligence (ICAI), 2012, Las Vegas, Nevada, USA, Proceedings of the 14th International Conference on Artificial Intelligence.

Campos, G. A. L. ; Freire, E. S. S. ; Cortés, M. I. ; Vasconcelos, W. W. (2013) An Approach for Norm-Based Behavior Modification in Model-Based Reflex Agents. In: 15th International Conference on Artificial Intelligence (ICAI), 2013, Las Vegas, Nevada, USA, Proceedings of the 15th International Conference on Artificial Intelligence. Cruz, F. I. S; Rocha Jr., R. M.; Freire, E. S. S. and Cortés, M. I. (2014) Norm-Based Behavior Modification in Reflex Agents - An Implementation in JAMDER 2.0. In: 16th International Conference on Enterprise Information Systems

(ICEIS), 2014, Lisboa, Portugal, Proceedings of the 16th International Conference on Enterprise Information Systems.

Dastani, M. D.; Grossi, J.-J. C.; Meyer, and Tinnemeier, N. (2018) Normative multi-agent programs and their logics. In Proc. Workshop on Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS'08), pages 236–243.

Dybalova, D.; Testerink, B.; Dastani, M.; Logan, B.; Dignum, F. and Chopra, A. (2013) A Framework for Programming Norm-Aware Multi-Agent Systems In: 15th International Workshop on Coordination, Organisations, Institutions and Norms (COIN 2013).

Figueiredo, K. and Silva, V. T. (2010) NormML: A Modeling Language to Model Norms. In: 1st Workshop on Autonomous Software Systems, 2010, Salvador, Brazil.

Freire, E. S. S.; Campos, G. A. L.; Cortes, M. I.; Vasconcelos, W. W. (2013) Norm-Based Behavior Modification in Model-Based Reflex Agents. In: 2013 Brazilian Conference on Intelligent Systems (BRACIS), 2013, Fortaleza, Proceedings of the 2013 Brazilian Conference on Intelligent Systems. p. 38.

Jennings, N. R.. (1995) Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artificial Intelligence, 75(2):195–240.

Jones, A. J. I. and Sergot, M. (1993) On the characterisation of law and computer systems: The normative systems perspective. In Deontic Logic in Computer Science: Normative System Specification, pages 275–307. John Wiley and Sons.

Kaminka, G., Pynadah, D. and Tambe, M. (2002) Monitoring teams by overhearing: A multi-agent plan-recognition approach. Journal of Artificial Intelligence Research, 17: p.83–135.

Mazouzi, H., Seghrouchni, A. E. F., and Haddad, S.. (2002) Open protocol design for complex interactions in multi-agent systems. In 1st Int. Joint Conference on Autonomous Agents and Multiagent Systems, pages 517–526.

Modgil, S.; Faci, N.; Meneguzzi, F.; Oren, N.; Miles, S. and Luck, M. (2009) A framework for monitoring agent-based normative systems. In: 8th International Conference on Autonomous Agents and Multiagent Systems, 2009, Budapeste, Hungria, Proceedings of the 8th International Foundation for Autonomous Agents and Multiagent Systems. Volume 1p. 153–160.

Piunti, M.; Ricci, A.; Boissier, O. and Hübner, J. F. (2010) Programming open systems with agents, environments and organizations. In: 11th Workshop nazionale 'Dagli Oggetti agli Agenti', 2010, Rimini, Italy, Proceedings of the WOA 2010 11th Workshop nazionale 'Dagli Oggetti agli Agenti'.

Tambe, M. (1997) Towards flexible teamwork. Journal of Artificial Intelligence Research, 7:83–124.

Vlissides, J.; Helm, R.; Johnson, R. and Gamma, E. (1995) Design patterns: Elements of reusable object-oriented software. Massachusetts: Addison-Wesley. p120.

Autorregulação de Processos de Trocas Sociais em SMA: um modelo de sociedade de agentes BDI evolucionários e culturais no contexto do JaCaMo

Andressa von Laer[1], Graçaliz P. Dimuro[1], Diana Adamatti[1], Marilton S. Aguiar[2]

[1] Programa de Pós-Graduação em Computação, Centro de Ciências Computacionais
Universidade Federal do Rio Grande (FURG)
Campus Carreiros: Av. Itália km 8 - Bairro Carreiros – Rio Grande – RS – Brasil

[2] Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas (UFPel)
Rua Gomes Carneiro, 1 - Centro – Pelotas – RS – Brasil

{andressavonlaer,gracaliz,dianaada}@gmail.com,marilton@inf.ufpel.edu.br

Resumo. *Na literatura foi proposto um modelo híbrido evolucionário de autorregulação de processos de trocas sociais entre agentes em um sistema multiagente baseado em Teoria dos Jogos e Algoritmos Genéticos, procurando tornar os agentes independentes e reguladores dos processos de trocas com seus parceiros. Este modelo foi implementado no Netlogo. Entretanto observa-se que certas características envolvidas em trocas sociais são mais adequadamente tratadas com agentes cognitivos, como Agentes BDI (Belief, Desire, Intention). Este artigo introduz um modelo de sociedade de agentes BDI evolucionários e culturais para autorregulação de processos de trocas sociais na plataforma JaCaMo com base no Jogo de Autorregulação de Trocas Sociais, com a adição do conceito de reputação como uma crença de grupo, analisando a evolução das estratégias de troca dos agentes, o aumento de interações bem sucedidas, e a melhora dos resultados obtidos nas interações.*

1. Introdução

O comportamento coletivo dos agentes é utilizado para resolver problemas em sistemas multiagentes [Woolridge 2001]. Para tornar isto possível é necessário que haja *interação* entre estes agentes e por consequência a *qualidade* destas interações é decisiva para o bom funcionamento do sistema, já que, por exemplo, uma falha de comunicação, falta de confiança, atitudes egoístas, ou uma ação desleal, podem afastar o sistema de uma solução.

Com base na *Teoria das Trocas Sociais* de Piaget [Piaget 1995], interações em uma organização social podem ser entendidas como *trocas de serviços* realizadas entre os agentes que compõem esta organização, e a respectiva *avaliação destes serviços* por parte dos agentes envolvidos [Rodrigues and Luck 2009, Grimaldo et al. 2007]. Os agentes podem ter comportamentos de trocas distintas, configurando, por exemplo, atitudes egoístas (busca de vantagens), altruístas, tolerantes, entre outras, as quais podem determinar diferentes estratégias de troca de agentes [Pereira et al. 2008a, Farias et al. 2013, Dimuro et al. 2005]. De acordo com os comportamentos dos agentes em suas interações, pode-se pensar em qualificar a sua *reputação* [Castelfranchi et al. 2000,

Castelfranchi and Falcone 1998]. Reputação é um dos conceitos oriundos das ciências sociais que pode auxiliar os agentes na escolha de parceiros em uma sociedade onde há outros agentes que podem agir de forma a prejudicar o equilíbrio da sociedade. Por exemplo, em [Schmitz 2011], foi proposta uma arquitetura de implementação para o ForTrust, um modelo teórico para o conceito de reputação entre agentes. Outros autores que tratam do conceito de reputação em sistemas multiagentes são, por exemplo, [Sabater and Sierra 2002, Huynh et al. 2006, Serrano et al. 2012, Yu and Singh 2002, Zacharia 2000]. Por outro lado, existem vários trabalhos na literatura que se preocupam com o problema da *regulação* das interações sociais em sistemas multiagentes. Por exemplo, em alguns trabalhos [Dimuro et al. 2005, Dimuro et al. 2011] o equilíbrio dessas interações é determinado de acordo com o *balanço de valores* que os agentes trocam enquanto interagem. Em geral, mecanismos para controle social podem operar de duas formas. Por um lado, regras sociais podem ser impostas por autoridades que têm a capacidade de forçar os agentes a seguir tais regras; por outro lado, regras sociais podem ser *interiorizadas* pelos agentes, de tal forma que eles seguem as regras porque estas são incorporadas pelos comportamentos dos agentes.

Em [Macedo 2013] foi proposto um modelo híbrido evolucionário de *autorregulação* de processos de trocas sociais entre agentes em um sistema multiagente baseado em Teoria dos Jogos [Fiani 2006] e Algoritmos Genéticos [Goldberg 1989], procurando tornar os agentes independentes e reguladores dos processos de trocas com seus parceiros. O modelo proposto em [Macedo 2013] foi implementado no Netlogo[1]. Entretanto observa-se que certas características envolvidas em trocas sociais são mais adequadamente tratadas com agentes cognitivos, como *Agentes BDI* (*Belief, Desire, Intention*). A teoria BDI foi desenvolvida pelo filósofo Michael Bratman [Wooldridge 2000, Bratman 1999], sobre o raciocínio prático, que consiste em ponderar considerações conflitantes a favor e contra alternativas competitivas. As considerações relevantes são determinadas pelos desejos e crenças do agente.

Os *Algoritmos Genéticos* (AGs) e *Culturais* (ACs) [Reynolds 1994] situam-se dentro de um paradigma na Inteligência Artificial (IA) que acredita na possibilidade de reproduzir características humanas em uma máquina para que esta possa resolver problemas. Os AGs foram introduzidos nos anos 60 por John Holland e posteriormente desenvolvidos por alunos da Universidade de Michigan. Constituem de uma técnica de busca e otimização inspirada na seleção natural e reprodução genética (Darwinismo) [Goldberg 1989], com o propósito de estudar os fenômenos da evolução. Os AGs são a base dos Algoritmos Culturais (ACs), porém estes dispõem de um componente chamado Espaço de Crenças. Os ACs baseiam-se na ideia de que a cultura também evolui, e sua evolução é mais rápida que a genética, possibilitando uma melhor adaptação do agente ao ambiente [Reynolds and Zanoni 1992].

A questão de pesquisa que surgiu foi *a possibilidade de estudar o comportamento de um modelo BDI híbrido evolucionário de autorregulação de processos de trocas sociais em sistemas multiagentes, com base na Teoria dos Jogos e em Algoritmos Genéticos e Culturais, e que outros conceitos relacionados à consideração da cultura na sociedade de agentes poderiam ser incorporados neste modelo*. Este artigo introduz a versão inicial de um modelo de tal sociedade de agentes no contexto do JaCaMo, onde a cultura de grupo,

[1]Disponível em: `http://ccl.northwestern.edu/netlogo`

baseada-se no conceito de reputação, é modelada usando artefatos do CArtAgO.

Este artigo está organizado da seguinte maneira: a Seção 2 descreve o modelo proposto neste artigo e apresenta sua implementação; na Seção 3 são apresentados e analisados os resultados obtidos; por fim na Seção 4 são apresentadas as considerações finais deste trabalho e trabalhos futuros.

2. Modelo

O modelo de jogo proposto neste trabalho toma como base o jogo apresentado em [Macedo 2013], um jogo de informação incompleta de autorregulação de processos de trocas sociais onde os agentes evoluem suas estratégias com o objetivo de maximizar o seu *fitness* através de um algoritmo evolutivo. O fitness é uma função que avalia o resultado material líquido das trocas com todos os agentes de seu entorno, recebendo influência de fatores que caracterizam as estratégias e atitudes de troca dos agentes, perante os resultados dos outros agentes. Neste trabalho, Algoritmos Genético e Cultural são usados, respectivamente, como base no processo de aprendizado do agente e em um espaço de crença comum a todos os agentes que influencia na tomada de decisões do jogo. A adição de um espaço de crença (ou uma cultura) comum a todos os agentes envolvidos no sistema trabalha como um Ponto Focal [Fiani 2006], servindo de referência para os agentes, já que os agentes não conhecem uns aos outros e portanto não sabem contra quais estratégias estão jogando. Em Teoria dos Jogos, um ponto focal é um elemento que se destaca entre outros e serve de referência aos jogadores para eles coordenarem suas decisões entre vários equilíbrios de Nash possíveis [Fiani 2006]. Este elemento deve ser de conhecimento comum a todos, ou seja, deve haver o compartilhamento de experiências entre os agentes. O espaço de crenças usado neste trabalho é baseado no trabalho desenvolvido em [Schmitz 2011].

O modelo foi implementado em Jason usando o conceito de Agentes e Artefatos [Ricci et al. 2008] para implementar as crenças de grupo. Foi utilizada a plataforma CArtAgO, que dá suporte a este conceito, para a implementação dos artefatos e do espaço de crença. A concepção deste modelo foi organizada em duas partes: a primeira é a base do jogo, onde ocorrem as trocas e as avaliações das mesmas, estas são tratadas como objetivos dos agentes envolvidos no sistema, enquanto os parâmetros que definem suas estratégias são tratados como crenças; a segunda é a criação das crenças de grupo (CGs) com artefatos. Por restrição de espaço o modelo BDI não é detalhado neste artigo (veja em [von Laer 2014].

A Figura 1 mostra a sequência básica e simplificada da primeira parte, onde ocorrem as trocas entre dois agentes. Na primeira etapa do jogo o agente a realiza uma oferta com algum valor de investimento (R) para o agente b. Este gera um valor de satisfação (S) e um valor de reconhecimento (T), ou débito, ambos referentes à oferta de a. No final desta etapa o agente a acaba com um valor virtual (V), ou seja, crédito devido à ação que realizou para o agente b. A segunda etapa do jogo é semelhante a primeira porém referente a uma possível cobrança de dívida do agente a ao agente b, onde o agente a cobra do agente b um serviço relativo ao seu valor virtual (V) de crédito adquirido na primeira etapa. O agente b por sua vez possui em sua consciência um valor de débito (T) e então realiza uma oferta com valor de investimento (R) ao agente a, que por sua vez também gerará um valor de satisfação (S) referente a oferta de b. Os agentes avaliam os

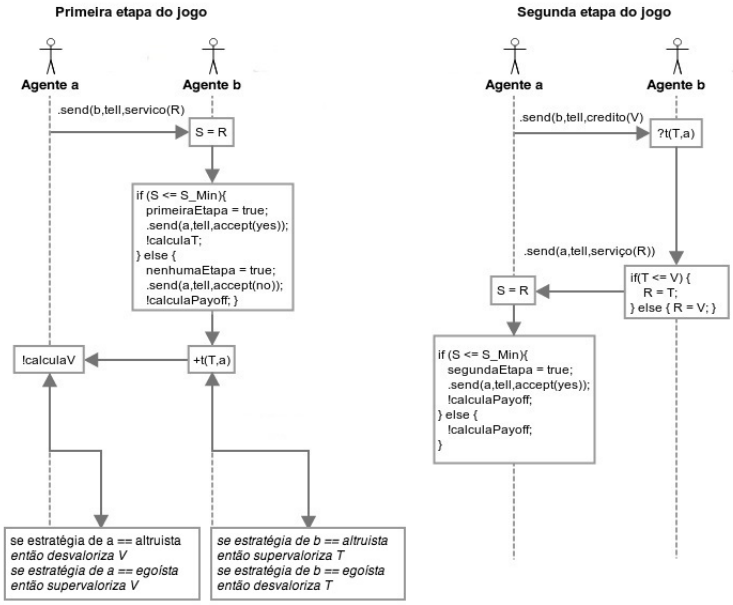


Figura 1. Sequência das trocas

serviços/ações de acordo com suas estratégias, por exemplo, um agente com estratégia de troca egoísta terá maior probabilidade de desvalorizar um serviço recebido e supervalorizar um serviço oferecido. Estes cálculos são feitos através das seguintes equações [Macedo 2013]:

Depreciação:

$t_{ij} = (1 - k_{ij}^{dt})s_{ij}$ e $v_{ij} = (1 - k_{ij}^{dv})r_{ij}$

Valorização:

$t_{ij} = s_{ij} + (1 - s_{ij})k_{ij}^{ot}$ e $v_{ij} = r_{ij} + (1 - r_{ij})k_{ij}^{ov}$

onde: $r, s \in [0, 1]$ são, respectivamente, o valor atual de investimento feito por um agente e o valor atual de satisfação pelo serviço recebido; $k^{\rho t}$, $k^{\rho v} \in [0, 1]$ são, respectivamente, os valores de depreciação ($\rho = d$) e valorização ($\rho = o$) do serviço recebido, usados para o cálculo de débito e crédito (t_{ij} e v_{ij}).

Na instância do jogo que é considerada para implementar/avaliar o modelo, existem cinco agentes que realizam as trocas, cada um com uma estratégia de troca diferente, são elas: altruísta, altruísta fraco, egoísta, egoísta fraco e racional. A diferença entre o altruísta e o altruísta fraco, o egoísta e egoísta fraco, está nos parâmetros menos extremos que definem sua estratégia de troca, ou seja, os valores dos mesmos são mais equilibrados em relação aos valores do agente racional, por exemplo.

A segunda parte do modelo consiste nos artefatos de crença de grupo e de reputação, que constituem a cultura da sociedade de agentes. Estes artefatos são criados no início do jogo por um agente **mediador**, que também é responsável por dar início às trocas enviando uma mensagem para todos os agentes começarem a sequência de trocas, apresentada na Figura 1. Os artefatos criados, respectivamente, armazenam as crenças enviadas pelos agentes após obter experiências nas trocas e criam a reputação dos agentes através de um método de síntese. As crenças que compõem os artefatos são propriedades observáveis, e os anúncios são tratados como uma operação de interface onde são

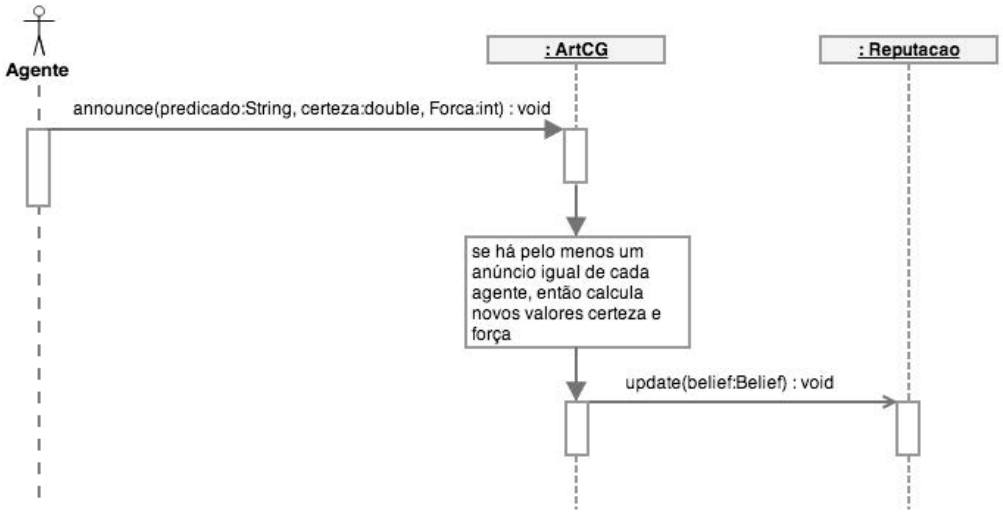


Figura 2. Sequência do método announce

passados como parâmetros: o predicado anunciado, o grau de certeza de uma crença e a força desta certeza. A composição de uma crença de grupo (CG) funciona da seguinte maneira: as regras de formação das crenças individuais se encontram dentro da mente dos agentes, as regras que formam as crenças de grupo (regras de síntese) se encontram em uma entidade externa aos agentes e a comunicação para a formação da CG é feita através de anúncios, enviados a um componente que as agrega, formando uma CG. Um anúncio é composto por um predicado, um grau de certeza e uma força, por exemplo, no anúncio `personalidade ("egoísta", bob)` com um grau de certeza 0.8 e força 6, o anunciante tem bastante certeza que o agente *bob* possui uma estratégia de troca egoísta, baseado em 6 experiências em trocas que teve com o agente bob.

Ao receber um anúncio o artefato adiciona-o à lista de anúncios, se há pelo menos um anúncio igual a este proveniente de cada agente presente no sistema, este anúncio se torna uma reputação. Este processo pode ser observado na Figura 2.

Para criar uma reputação, os valores de certeza e força são calculados através do processo de síntese e então o artefato `Reputação` é notificado sobre a nova crença de grupo através do método `update` (Figura 2). Caso já exista crença de grupo no artefato `Reputação` com o mesmo predicado, ela é substituída atualizando-se assim os seus valores de certeza e força, caso contrário é adicionada como nova crença de grupo.

Para implementar o modelo desenvolvido neste trabalho, considerou-se uma sociedade hetereogênea (composta por agentes de cinco diferentes estratégias de troca), e devido a este fato, a métrica escolhida para este trabalho é a de **síntese ponderada** [Schmitz 2011], onde os anúncios são sintetizados de maneira a buscar um termo intermediário entre eles, não beneficiando assim uma sociedade somente otimista ou pessimista. A função de síntese ponderada é apresentada a seguir, onde $sinpon_p$ é a função, c é o grau de certeza, s é a força calculada e $|C_p|$ é o subconjunto contendo todos os anúncios de um predicado p.

$$sinpon_p =< p, c, s >$$

$$c = \frac{\sum_{a \in C_p} c_a s_a}{\sum_{a \in C_p} s_a} \qquad s = \frac{\sum_{a \in C_p} s_a}{|C_p|}$$

2.1. Implementação

Cada agente possui em sua base de crenças uma crença "cromossomo", por exemplo, do agente altruísta:

```
chromosome([r(0),s(0),rmax(0.8),smin(0.2),a(0.1),b(0.9),kt(0.2),kv(0.2)]).
```

composto por parâmetros usados nas trocas que representam características dos mesmos, e entre eles há cinco parâmetros que determinam sua estratégia

$$r, r^{max}, s^{min}, k^{\rho t} \text{ e } k^{\rho v}$$

onde, r é o valor de investimento do agente, r^{max} é o valor máximo que o agente pode investir, s^{min} é o valor mínimo de satisfação que o agente aceita, $k^{\rho t}$ é usado no cálculo de quanto o agente depreciará o serviço recebido, e $k^{\rho v}$ é usado no cálculo de quanto o agente valorizará o serviço oferecido por outro agente.

Após cada troca entre agente i e j, estes calculam através de uma função *payoff* a recompensa recebida em cada troca. Cada troca consiste em duas etapas, conforme a Figura 1. Se as duas etapas forem bem sucedidas, então a recompensa dos agentes é maior. Se nenhuma etapa ocorrer, ou seja, o agente j recusou o serviço na primeira etapa, a recompensa é 0 (zero). A equação que define estes valores é a seguinte [Macedo 2013, Bo 2010, Macedo et al. 2012]:

$$p_{ij} = \begin{cases} \frac{1-r_{\mathrm{I}_{ij}}+s_{\mathrm{II}_{ij}}}{2} & \text{se} \quad (r_{\mathrm{I}_{ij}} \leq r_i^{\max} \wedge s_{\mathrm{I}_{ji}} \geq s_j^{\min}) \wedge (r_{\mathrm{II}_{ji}} \leq r_j^{\max} \wedge s_{\mathrm{II}_{ij}} \geq s_i^{\min}) \\ \frac{1-r_{\mathrm{I}_{ij}}}{2} & \text{se} \quad (r_{\mathrm{I}_{ij}} \leq r_i^{\max} \wedge s_{\mathrm{I}_{ji}} \geq s_j^{\min}) \wedge (r_{\mathrm{II}_{ji}} > r_j^{\max} \vee s_{\mathrm{II}_{ij}} < s_i^{\min}) \\ 0 & \text{se} \quad (r_{\mathrm{I}_{ij}} > r_i^{\max} \vee s_{\mathrm{I}_{ji}} < s_j^{\min}) \wedge (r_{\mathrm{II}_{ji}} > r_j^{\max} \vee s_{\mathrm{II}_{ij}} < s_i^{\min}) \end{cases} \tag{1}$$

Após um agente i jogar com todos os outros agentes presentes no sistema, ele calcula o seu grau de adaptação através da função de *fitness*, definida pela seguinte equação [Macedo 2013, Bo 2010, Macedo et al. 2012]:

$$\mathrm{F}_i(\mathrm{X}) = x_i - \frac{a_i}{(m-1)} \sum_{j \neq i} \max(x_j - x_i, 0) - \frac{b_i}{(m-1)} \sum_{j \neq i} \max(x_i - x_j, 0),$$

onde x é a recompensa material de um agente, m é o número total de agentes, X é o vetor das recompensas adquiridas por cada um dos m agentes após jogar com seus vizinhos, a_i é o valor que representa o grau de tolerância do agente i ao quando o retorno dele é menor do que o de seus vizinhos (podemos chamar de inveja), e b_i é o valor que representa o grau

Tabela 1. Vetor de Ajuste

	r_i	r_i^{max}	s_i^{min}		r_i	r_i^{max}	s_i^{min}		r_i	r_i^{max}	s_i^{min}
p_i^0	$\uparrow$	$\uparrow$	$\uparrow$	p_i^9	$=$	$\uparrow$	$\uparrow$	p_i^{18}	$\downarrow$	$\uparrow$	$\uparrow$
p_i^1	$\uparrow$	$\uparrow$	$=$	p_i^{10}	$=$	$\uparrow$	$=$	p_i^{19}	$\downarrow$	$\uparrow$	$=$
p_i^2	$\uparrow$	$\uparrow$	$\downarrow$	p_i^{11}	$=$	$\uparrow$	$\downarrow$	p_i^{20}	$\downarrow$	$\uparrow$	$\downarrow$
p_i^3	$\uparrow$	$=$	$\uparrow$	p_i^{12}	$=$	$=$	$\uparrow$	p_i^{21}	$\downarrow$	$=$	$\uparrow$
p_i^4	$\uparrow$	$=$	$=$	p_i^{13}	$=$	$=$	$=$	p_i^{22}	$\downarrow$	$=$	$=$
p_i^5	$\uparrow$	$=$	$\downarrow$	p_i^{14}	$=$	$=$	$\downarrow$	p_i^{23}	$\downarrow$	$=$	$\downarrow$
p_i^6	$\uparrow$	$\downarrow$	$\uparrow$	p_i^{15}	$=$	$\downarrow$	$\uparrow$	p_i^{24}	$\downarrow$	$\downarrow$	$\uparrow$
p_i^7	$\uparrow$	$\downarrow$	$=$	p_i^{16}	$=$	$\downarrow$	$=$	p_i^{25}	$\downarrow$	$\downarrow$	$=$
p_i^8	$\uparrow$	$\downarrow$	$\downarrow$	p_i^{17}	$=$	$\downarrow$	$\downarrow$	p_i^{26}	$\downarrow$	$\downarrow$	$\downarrow$

de tolerância de i quando este recebe uma recompensa maior do que a de seus vizinhos (pode-se chamar de culpa).

Para avaliar o *fitness*, o agente compara o atual resultado com o anterior, se o *fitness* atual supera o valor do *fitness* anterior significa que a atual estratégia é melhor que a anterior, então o agente faz um ajuste no vetor de estratégias aumentando a probabilidade da atual estratégia ser escolhida novamente e aumentando ou diminuindo os parâmetros (conforme a estratégia) do cromossomo que definem sua estratégia de troca. O vetor de ajustes das estratégias é representado pela Tabela 1. Nele há 27 estratégias de ajuste, por exemplo, p_i^0 representa a probabilidade de aumentar os valores de r_i, r_i^{max} e s_i^{min}, p_i^5 representa a probabilidade de aumentar o valor de r_i, manter o valor de r_i^{max} e diminuir o valor de s_i^{min}.

Para dar início à segunda etapa na troca entre dois agentes i e j, o agente i "cobra" o j pelo serviço prestado na primeira etapa, e para isto ele envia a j o valor de crédito que se acha merecedor. Através de uma comparação entre o valor de crédito e o valor r que i investiu na primeira etapa, j é capaz de tirar uma conclusão sobre a estratégia de troca de i:

- Se $Ri > Vi$: se o valor investido por i na primeira etapa é maior que o valor de crédito que ele atribuiu a si mesmo, j conclui que i é altruísta;
- Se $Rj < Vj$: se o valor investido por i na primeira etapa é menor que o valor de crédito que ele atribuiu a si mesmo, j conclui que i é egoísta, pois supervalorizou o seu serviço oferecido;
- Se o valor investido é igual ao valor de crédito, o agente j conclui que i é um agente racional.

Feita esta análise, o agente j envia sua conclusão sobre a estratégia do agente i para o artefato de crença de grupo `ArtCG` através do método `announce`, conforme mostrado na Figura 2, para formar uma reputação sobre o agente i. Se no artefato `ArtCG` já existe pelo menos um anúncio de cada agente do sistema, com o mesmo predicado, então uma nova reputação é criada. Formada a reputação no artefato de `Reputação`, ela é adicionada às crenças do agentes, tornando-se assim uma crença de grupo comum a todos os participantes do jogo. Quando existe uma reputação que diz que um agente i é egoísta, os agentes enviam uma mensagem informando o agente **mediador**, e este envia uma mensagem ao agente egoísta informando-lhe que este não poderá participar da próxima jogada. Assim o agente egoísta não consegue evoluir seu valor de *fitness*, o que o obriga a modificar sua estratégia para poder entrar no jogo novamente, aumentando o seu

valor de investimento r e o valor máximo de investimento r_{max}, e diminuindo seu valor de satisfação mínimo s_{min}.

3. Análise das simulações

A estratégia social dos agentes é determinada através da forma como o agente se comporta perante às trocas propostas por outros agentes, pela forma como este agente determina o valor do investimento que pretende realizar, e também pelo seu grau de culpa/inveja quando compara resultados com os dos outros agentes. Conforme os resultados globais emergem na evolução no tempo, os agentes se tornam reguladores dos processos de troca.

As características avaliadas que definem cada estratégia e são determinantes na evolução são: o valor máximo que o agente pretende investir, o valor de satisfação mínima aceito quando um agente recebe um serviço/proposta e o valor de investimento que pretende realizar.

Foram definidos dois cenários diferentes, um sem as crenças de grupo, e outro com as crenças de grupo como uma "cultura" comum a todos os agentes. Em cada cenário há cinco agentes, cada um com uma estratégia diferente, e cada simulação foi realizada com 300 ciclos, em um total de 20 simulações por cenário. Conforme as duas etapas mostradas na Figura 1, dado n agentes, cada um joga com $n-1$ agentes podendo haver zero, uma, ou duas etapas bem sucedidas em cada troca, então um ciclo de simulação é composto de,

$$n(n-1) + w_1 + w_2 + \cdots + w_n$$

de jogadas de etapas do tipo I e/ou II (bem sucedidas ou não), onde w_1 é o número de agentes com que o agente 1 tem crédito após ter realizado a primeira etapa de troca com todos os outros agentes (isto é, o número de trocas bem sucedidas para o agente 1), e analogamente definem-se $w_2, \ldots, w_n$. Portanto, em um ciclo, o número de trocas de tipo I (bem sucedidas ou não) é $n(n-1)$, e o número de trocas do tipo II (bem sucedidas ou não) é $w_1 + w_2 + \cdots + w_n$. Observe que, se todas a trocas do tipo I foram bem sucedidas para todos os agentes, então um ciclo de simulação vai ter $2n(n-1)$ trocas do tipo I ou II (bem sucedidas ou não).

Em ambos os cenários foi constatado que o sistema se estabiliza antes dos 300 ciclos. A Figura 3 apresenta a contagem do número de uma, duas, e nenhuma troca no estágio inicial (primeiro ciclo) e final das trocas em um intervalo de 10 ciclos. A evolução do número de duas trocas bem sucedidas é representada pela linha rosa, uma troca representada pela linha azul e nenhuma troca pela linha vermelha.

Analisando o comportamento das curvas, observa-se que a evolução nas estratégias dos agentes proporcionou o aumento no número de trocas bem sucedidas, que começa em 8 e termina em 20, e a queda na quantidade de interações sem sucesso em um curto período de tempo. A média e o desvio padrão do número de trocas é representada na Tabela 2. Na Figura 4 é apresentada a evolução do valor de fitness dos agentes no sistema em um período de 300 ciclos. O agente altruísta é representado pela linha da cor rosa, o altruísta fraco pela linha laranja, o agente racional pela linha azul, o egoísta pela linha vermelha e o egoísta fraco pela linha verde. Observa-se a evolução do valor de fitness por parte de todos os agentes presentes no sistema, todos acabam convergindo para o valor máximo de *fitness*: 1. Na Tabela 3 observa-se a média do valor de *fitness* no

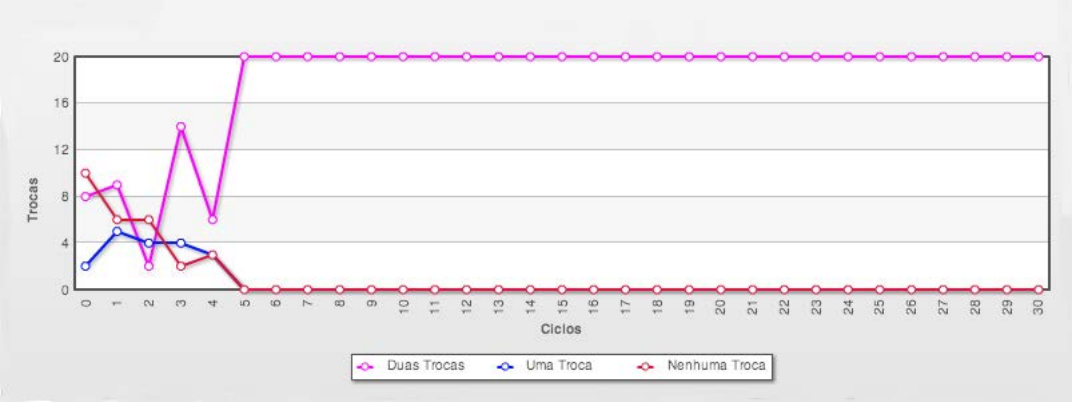


Figura 3. Evolução do número de trocas com cultura

Tabela 2. Média e desvio padrão do número de trocas com cultura

Média		
	Início	Fim
Uma troca	3,65	0,1
Duas trocas	2,8	13,6
Nenhuma troca	8,65	0,1

Desvio padrão		
	Início	Fim
Uma troca	2,3680	0,4472
Duas trocas	3,1722	4,9672
Nenhuma troca	3,3289	0,4472

ciclo inicial e final das simulações. Os valores do desvio padrão da evolução do *fitness* são apresentados na Tabela 4. Por fim, na Tabela 5 observa-se os valores de desvio padrão e média globais. Observa-se através da Tabela 2 que o número de duas trocas aumentou em 385,71% com cultura. Em relação as estratégias, através da Tabela 3 observa-se que o aumento do *fitness* do agente altruísta se deu em 252,49%, enquanto do agente altruísta fraco se deu em 258,20%, do agente racional em 188,94%, do agente egoísta em 385,77% e, por fim, do agente egoísta fraco em 258,58%.A estratégia que mostrou menor evolução foi a estratégia racional, enquanto a estratégia egoísta apresentou uma evolução maior.

Observou-se que no caso com cultura, o aumento do número de duas trocas foi maior (385,71%) em relação ao cenário sem a cultura (171,73%). Em relação ao *fitness*, no cenário com cultura apenas a estratégia egoísta fraco não apresentou maior aumento na média de *fitness* (343,95% sem cultura e 258,58% com cultura), as outras quatro estratégias mostraram um maior aumento em seus valores de *fitness*, como mostrado na Tabela 6. Também observa-se que em ambos os cenários, a estratégia racional foi a que mostrou menor evolução em relação as outras, enquanto as estratégias egoístas mostraram maior evolução.

4. Considerações Finais

Sistemas multiagentes [Wooldridge 2000] é uma sub-área da Inteligência Artificial focada na resolução de problemas complexos e no estudo de agentes autônomos em um universo multiagente. Vários trabalhos foram desenvolvidos [Pereira et al. 2008a, Dimuro et al. 2005] para tratar a questão da autorregulação dos agentes, ou seja, na descentralização do mecanismo de regulação do sistema. O desenvolvimento de aplicações baseadas na abordagem orientada a agentes possibilita um maior nível de abstração, uma vez que o mundo real está repleto de agentes, e tal nível de abstração

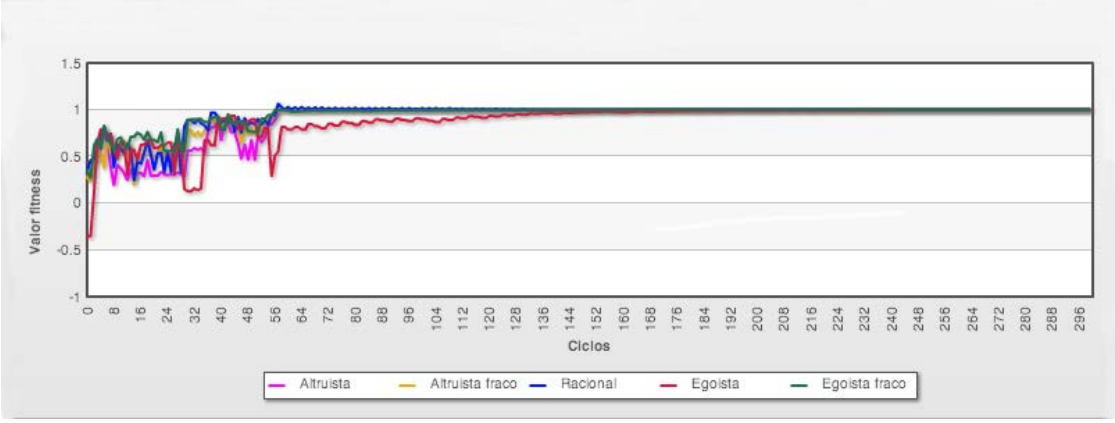


Figura 4. Evolução do valor fitness com cultura

Tabela 3. Média de fitness com cultura

	Fitness inicial	Fitness final
Altruísta	0,2784	0,9817
Altruísta fraco	0,2741	0,9818
Racional	0,3374	0,975
Egoísta	-0,3317	0,9479
Egoísta fraco	0,2767	0,9923

não é usual nas metodologias tradicionais de modelagem e implementação. Em certos sistemas multiagentes (por exemplo, utilizados em simulação social e de valores, tais como: confiança, reputação, e outras características mais subjetivas), existe a necessidade de que os agentes possuam a capacidade de raciocínio, e a Arquitetura BDI viabiliza isto na implementação de agentes conforme o modelo BDI.

Neste trabalho, a aplicação de autorregulação de trocas sociais proposta em [Macedo 2013] foi reestruturada para a arquitetura BDI utilizando a linguagem Jason e com a adição de crenças de grupo, ou seja, uma "cultura" comum a todos os agentes envolvidos no sistema tratada como um artefato do CArtAgO. Segundo a Teoria de Piaget o equilíbrio é alcançado quando ocorre reciprocidade nas trocas durante as interações, e através da evolução das estratégias os agentes maximizam seus valores de adaptação tornando-se autorreguladores dos processos de trocas e contribuindo assim para o crescimento do número de interações bem sucedidas. Foi observado nos dois cenários que todos os agentes evoluíram e contribuíram para a evolução da sociedade, e também que quando os serviços oferecidos são mais justos (ou equilibrados), maior é a quantidade de interações bem sucedidas. O número de trocas bem sucedidas aumentou relativamente em ambos os casos, resultado da autorregulação dos processos de trocas. Comparando os dois cenários foi observado que a adição da reputação como ponto focal nas trocas sociais teve a influência esperada na evolução das estratégias dos agentes e dos processos de troca. Notou-se também que o valor médio de *fitness* final foi maior para todos os agentes quando há o uso da reputação como cultura. Quando considera-se a cultura, a auto-regulação das trocas se dá em torno de 5 ciclos, enquanto que sem cultura é em 10 ciclos.

Como trabalhos futuros é possível realizar simulações com outras composições

Tabela 4. Desvio padrão do fitness com cultura

	Fitness inicial	Fitness final
Altruísta	0,0253	0,0810
Altruísta fraco	0,0112	0,0781
Racional	0,0074	0,1118
Egoísta	0,0293	0,2005
Egoísta fraco	0,1284	0,0343

Tabela 5. Desvio padrão e média globais do fitness com cultura

	Inicial	Final
Desvio padrão global	0,2800	0,0167
Média global	0,1670	0,9757

Tabela 6. Aumento do fitness por cenário

Estratégia	Sem cultura	Com cultura
Altruísta	164,32%	252,49%
Altruísta fraco	144,89%	258,20%
Racional	71,79%	188,94%
Egoísta	297,05%	385,77%
Egoísta fraco	343,95%	258,58%

da sociedade utilizando artefatos de crença em diferentes escopos além da reputação, ou criando modos diferentes dos agentes racionalizarem as crenças de grupo, explorando a capacidade do modelo. Existem outras maneiras de utilizar os artefatos de crença de grupo para implementar o modelo de reputação, por exemplo, pode haver um artefato único que recebe todos os anúncios dos membros, um artefato por crença, distribuindo a carga dos anúncios entre vários artefatos, ou até um artefato por tipo de crença.

Referências

Bo, X. (2010). Social preference, incomplete information, and the evolution of ultimatum game in the small world networks: An agent-based approach. *J. Artificial Societies and Social Simulation*, 13(2).

Bratman, M. E. (1999). *Intention, plans, and practical reason*. Cambridge U. Press.

Castelfranchi, C. and Falcone, R. (1998). Principles of trust for MAS: Cognitive anatomy, social importance and quantification. In *Intl. Conf. of Multi-agent Systems (ICMAS)*, pages 72–79.

Castelfranchi, C., Falcone, R., Firozabadi, B., and Tan, Y. (2000). Special issue on trust, deception and fraud in agent societies. *Applied Artificial Intelligence Journal*, 1:763–768.

Dimuro, G. P., Costa, A. C. R., Gonçalves, L. V., and Pereira, D. R. (2011a). Recognizing and learning models of social exchange strategies for the regulation of social interactions in open agent societies. *Journal of the Brazilian Computer Society*, 17(3):143–161.

Dimuro, G. P., Costa, A. C. R., and Palazzo, L. (2005a). Systems of exchange values as tools for multi-agent organizations. *Journal of the Brazilian Computer Society*, 11:27–40.

Farias, G. P., Dimuro, G., Dimuro, G., and Jerez, E. D. M. (2013). Exchanges of services based on Piaget's theory of social exchanges using a BDI-fuzzy agent model. In *Proc. BRICS Countries Congress (BRICS-CCI) and 11th Braz. Cong. (CBIC) on Comp. Intelligence*, Los Alamitos. IEEE.

Fiani, R. (2006). *Teoria Dos Jogos*. CAMPUS.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Artificial Intelligence. Addison-Wesley.

Grimaldo, F., Lozano, M., and Barber, F. (2007). Coordination and sociability for intelligent virtual agents. In Sichman, J., Noriega, P., Padget, J., and Ossowski, S., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *LNAI*, pages 58–70. Springer, Berlin.

Huynh, T. D., Jennings, N. R., and Shadbolt, N. R. (2006). An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154.

Macedo, L., Dimuro, G., Aguiar, M., Costa, A., Mattos, V., and Coelho, H. (2012). Analyzing the evolution of social exchange strategies in social preference-based mas through an evolutionary spatial approach of the ultimatum game. In *Social Simulation (BWSS), 2012 Third Brazilian Workshop on*, pages 83–90. IEEE, Los Alamitos.

Macedo, L. F. K. (2013). Uma abordagem evolucionária e espacial para o jogo da autorregulação de processos de trocas sociais em sistemas multiagentes. Dissertação de mestrado, FURG.

Pereira, D., Gonçalves, L., Dimuro, G. P., and Costa, A. R. C. (2008a). Towards the self-regulation of personality-based social exchange processes in multiagent systems. In Zaverucha, G. and Costa, A., editors, *Advances in Artificial Intelligence - SBIA 2008*, volume 5249 of *Lecture Notes in Computer Science*, pages 113–123. Springer, Berlin.

Piaget, J. (1995). *Sociological Studies*. Routlege, London.

Reynolds, R. (1994). An introduction to cultural algorithm. In *Proc. 3rd Annual Conf. on Evolutionary Programming*, pages 131–139.

Reynolds, R. and Zanoni, E. (1992). Why cultural evolution can proceed faster than biological evolution. In *Proc. Intl. Symp. on Simulating Societies*, pages 81–93.

Ricci, A., Viroli, M., and Omicini, A. (2008). The a&a programming model and technology for developing agent environments in mas. In *Proc. 5th Intl. Conf. Programming Multi-agent Systems*, number 18 in ProMAS'07, pages 89–106, Berlin, Heidelberg. Springer-Verlag.

Rodrigues, M. R. and Luck, M. (2009). Effective multiagent interactions for open cooperative systems rich in services. In Sierra, C., Castelfranchi, C., Decker, K. S., and Sichman, J. S., editors, *Proc. 8th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, Budapest*, pages 1273–1274, Richland. IFAAMAS.

Sabater, J. and Sierra, C. (2002). Reputation and social network analysis in multi-agent systems. In *Proc. of AAMAS 2002*, pages 475–482. ACM.

Schmitz, T. L. (2011). Crenças de grupo como instrumento de formação da reputação: uma arquitetura baseada em agentes e artefatos. Dissertação de mestrado, UFSC.

Serrano, E., Rovatsos, M., and Botía, J. A. (2012). A qualitative reputation system for multiagent systems with protocol-based communication. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *Intl. Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2012*, pages 307–314. IFAAMAS.

Wooldridge, M. (2000). *Reasoning about rational agents*. MIT press.

Woolridge, M. (2001). *Introduction to Multiagent Systems*. Wiley, NY.

Yu, B. and Singh, M. P. (2002). An evidential model of distributed reputation management. In *In Proc. 1st Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 294–301. ACM Press.

Zacharia, G. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14:881–907.

von Laer, A. G. (2014). Autorregulação de processos de trocas sociais em SMA: um modelo de sociedade de agentes BDI evolucionários e culturais no contexto do JaCaMo. Dissertação de mestrado, Universidade Federal do Rio Grande.

Using Agents & Artifacts to Access the Semantic Web A Recommender System Case Study

Jéssica Pauli de C. Bonson[1], **Elder Rizzon Santos**[1]

[1]Laboratório IATE – Universidade Federal de Santa Catarina(UFSC)
Florianópolis – SC – Brazil

{jpbonson, elder}@inf.ufsc.br

Abstract. *In this paper we integrate the access of agents to the Semantic Web by means of the Agents & Artifacts model, using the Cartago framework. Our case study is a recommender system to define metadata of a Learning Object according to a metadata standard. The agents are able to provide recommendations by querying the DBPedia through artifacts. The contribution of our work is to develop a prototype integrating MAS to the Semantic Web using artifacts.*

1. INTRODUCTION

In this paper we research the use of the Agents & Artifacts (A&A) [Omicini et al. 2008a] model to ease the access of agents in a Multi-Agent Systems (MAS) to the functionalities of the Semantic Web. To test our approach we adopted the case study of a recommender system for the educational area. The A&A model emerged from the need of modelling the environment of a Multi-Agent Systems (MAS) as a first-class entity [Ricci et al. 2007, Omicini et al. 2008b, Ricci et al. 2008, Weyns et al. 2007]. Artifacts can model tools and resources to help the agents execute their tasks at runtime. In this work the artifacts model the semantic searches to access the Web of Data [Berners-Lee et al. 2001], more specifically, the DBPedia [Bizer et al. 2009]. Our agents access this repository to get recommendations for the metadata of a Learning Object (LO) through inferences on the partial knowledge provided by the user. Following the knowledge representation available on the Semantic Web, our artifacts aid the agents on the recommendations by querying linked data sources considering context-specific categories, classes or individuals.

2. The Recommender Model

In this paper we developed a recommender system for the metadata fields of a LO, based on an application profile of the metadata standard OBAA. To generate the recommendations our system uses a MAS composed of agents and artifacts specialized in accessing the DBPedia entities through SPARQL queries. The system execution starts with the user providing the partial metadata and requesting the system to provide recommendations for other metadata fields. The GUI gathers the partial knowledge and sends it to the MAS, the agents then use the artifacts to query the DBPedia to find recommendations based on the partial knowledge provided. Finally, the recommendations are sent back to the GUI that shows them to the user.

We focused the metadata Title, Description and Keywords of the OBAA profile, that can easily be mapped to some of DBPedia properties, such as dbpedia-owl:abstract, rdfs:comment, db-prop:title, db-prop:name, foaf:name and rdfs:label. To generate the recommendations, the partial data is transformed into keywords through Natural Language

Processing (NLP), using the Apache Lucene. The keywords are ranked based on their frequency, and the most frequent ones are provided to the agents at the artifacts. Each keyword is obtained by three agents, each one specialized in a type of semantic search. The agents process the keywords in parallel through semantic artifacts that query the DBPedia using SPARQL, returning ontology individuals that are similar to the keyword. The properties of the most returned individuals are used as a basis for the recommendations.

Figure 1 describes the recommender model in more details. The cloud represents the MAS, that contains the agents and artifacts. The InputArtifact and the OutputArtifact work as an integration between the MAS and the GUI, and are responsible for the agents coordination, where the agents get the source data to process and put the resultant outputs, respectively. The main part of the system is composed by nine agents and artifacts specialized in three types of semantic searches at DBPedia: individuals, classes and categories, where category is an informational structure derived from Wikipedia. Each of these artifacts are used by only one agent, we did it so the system can process the queries in parallel, because the A&A model doesn't allow to more than an agent using an artifact at the exact same time. The semantic artifacts provide two ways of performing a semantic search, where the difference is a trade-off between quality and quantity. The artifacts return the results as the individuals that were more similar to the keyword.

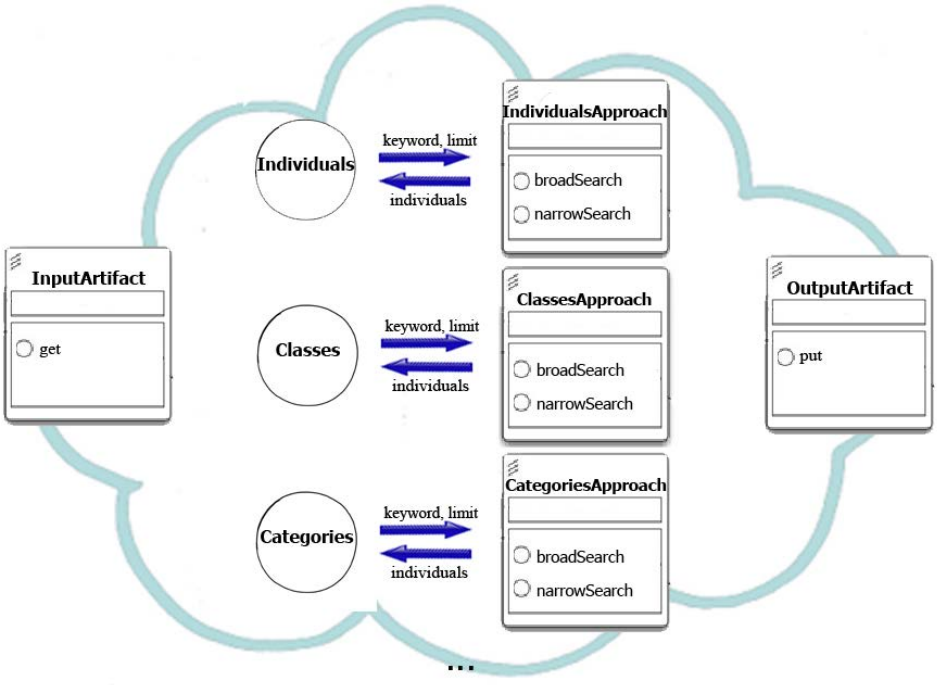


Figure 1. Overview of the recommender model.

As an example, the individual approach process the keywords in order to obtain individuals that contain the keyword in the properties db-prop:title or db-prop:name. A shorter version of the SPARQL query for the narrow search of this approach is shown below. The words preceded by the symbol @ are parameters that will be informed by

the agents when calling the artifact's operation. The query states the prefixes, defines the result being returned at SELECT and the search criteria at WHERE. The filters ban the results that belong to categories which results are poor related with an educational context. The parameters that the agent informs in this query are the keyword being searched, the depth of the search, and the limit of results.

```
PREFIX db-prop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl:
   <http://dbpedia.org/ontology/>
PREFIX rdf:
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:
   <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia_cat:
   <http://dbpedia.org/resource/Category:>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct ?object
WHERE {
{?object db-prop:title ?title . ?title
   <bif:contains> "+@keyword+" . }
UNION
{?object  db-prop:name ?name. ?name
   <bif:contains> "+@keyword+" . }
filter (NOT EXISTS
   {?object dcterms:subject ?Category .
   ?Category skos:broader dbpedia_cat:Sports
   option(TRANSITIVE, T_DISTINCT,
   t_max(@t_max)) }) .
filter (NOT EXISTS
   {?object dcterms:subject ?Category .
   ?Category skos:broader dbpedia_cat:Companies
   option(TRANSITIVE, T_DISTINCT,
   t_max(@t_max)) }) .
}
LIMIT @limit
```

Below we show a shorted version of the code for the classes_approach agent. It starts by discovering the artifacts that it will use through the goal myTools. Those are the artifacts for input, output and the one specialized on the classes approach. Then the goal consumeItems checks if new keywords are available in the InputArtifact at 0.5 seconds intervals. When a keyword is obtained it is used in the processItem goal, which executes the semantic search. First, the agent calls the artifact's operation execute_broad. the the agent uses the results from the broad search to execute a narrow search. If after executing the narrow search there are no results, the agent will send to the OutputArtifact the results from the broad search, otherwise the results from the narrow search will be sent. This decision is made by the goal decideOutcome. Before sending the results to the OutputArtifact through the put operation, the agent calls the operation execute_getIndividuals to obtain the individuals from each class as the final result from the semantic search.

```
!observe.

+!observe : true
  <- .wait(200);
    ?myTools(A1, A2, Id);
    !consumeItems(Id).

+!consumeItems(Id): true
<- .wait(500);
get_for_ClassesApproach(Item);
!processItem(Item, Id);
!!consumeItems(Id).

(...)

+!processItem(Keyword, Id) : true
<-  execute_broad(Keyword, 30, R1)
   [artifact_id(Id)];
execute_narrow(R1, R2)[artifact_id(Id)];
isEmpty(R2, Test)[artifact_id("input")];
!decideOutcome(R1, R2, Test, Id).

+!decideOutcome(R1, R2, Test, Id) : Test
<- execute_getIndividuals(R1, 20, R3)
   [artifact_id(Id)];
put(R3).

(...)

+?myTools(A1, A2, A3): true
  <- lookupArtifact("input",A1);
    lookupArtifact("output",A2);
 .my_name(N);
 lookupArtifact(N,A3).

(...)
```

3. Conclusion and Future Work

In this paper we developed a recommender system where agents access the Semantic Web by means of artifacts that model the environment. The system is able to obtain recommendations for LO metadata using the partial knowledge provided by the user to process semantic queries on DBPedia through SPARQL. The contributions and results of this work can be observed in two points of view:

a) Metadata recommenders systems: To the best of our knowledge, in the context of recommenders specific for educational LO metadata there are no works available to compare our results with. The system developed in this paper is an initial prototype for this context. We noticed that our systems' recommendations were useful to obtain more information about a concept, but most of the time they were unrelated with the educa-

tional context. We believe that it happened because DBPedia isn't a semantic repository specific for educational purposes, then lacking the pedagogical information necessary for the recommendations.

b) Architectures that enable agents to access the Semantic Web: The main objective of our paper was to developed a model of system where agents can have access to the Semantic Web by means of artifacts, so this is context where we are able to make most of our comparisons. The main difference between our work and the related work ([Klapiscak and Bordini 2009, da Silva and Vieira 2007, Kagal et al. 2003, Zou et al. 2003, Chen et al. 2004, Katasonov and Terziyan 2008]) regarding the access to the Semantic Web by agents is that in this work the integration is accomplished by means of artifacts.

Due to the adoption of artifacts, our model provides reusability: By using artifacts agents are able to access the Semantic Web without the need to implement code or an architecture specific for them. Also, conceptually any BDI agent is able to use artifacts are developed. Other relevant points are that artifacts decreases the computational burden on the agent side, since the agents just activate the desired operation at the artifact, and they can perform other tasks while the artifact process the operation.

The semantic artifacts use SPARQL queries with a predefined general structure. In the current version the customizable parts by the agents are the keywords being searched, the prefixes, the filters and some parameters of the search. In a future version we could use the functionality of internal properties of the artifacts, which can be modified at runtime by agents to change the services being offered by the artifact. So the agents would be able to customize the queries to DBPedia at runtime. Other possible future works include: real time recommendations; utilize the contents of the LO or the user's personal data as context; and an empirical evaluation of the system.

References

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165.

Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., and Chakraborty, D. (2004). Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6).

da Silva, D. M. and Vieira, R. (2007). Argonaut: Integrating jason and jena for context aware computing based on owl ontologies. *Agent, Web Services, and Ontologies Integrated Methodologies*, page 19.

Kagal, L., Perich, F., Chen, H., Tolia, S., Zou, Y., Finin, T., Joshi, A., Peng, Y., Cost, R. S., and Nicholas, C. (2003). Agents making sense of the semantic web. In *Innovative Concepts for Agent-Based Systems*, pages 417–433. Springer.

Katasonov, A. and Terziyan, V. (2008). Semantic agent programming language (s-apl): A middleware platform for the semantic web. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing*, ICSC '08, pages 504–511. IEEE Computer Society.

Klapiscak, T. and Bordini, R. H. (2009). Jasdl: A practical programming approach combining agent and semantic web technologies. In *Declarative Agent Languages and Technologies VI*, pages 91–110. Springer.

Omicini, A., Ricci, A., and Viroli, M. (2008a). Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.

Omicini, A., Ricci, A., and Viroli, M. (2008b). Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.

Ricci, A., Viroli, M., and Omicini, A. (2007). Give agents their artifacts: the a&a approach for engineering working environments in mas. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 150. ACM.

Ricci, A., Viroli, M., and Omicini, A. (2008). The a&a programming model and technology for developing agent environments in mas. In *Programming multi-agent systems*, pages 89–106. Springer.

Weyns, D., Omicini, A., and Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, 14(1):5–30.

Zou, Y., Finin, T., Ding, L., Chen, H., and Pan, R. (2003). Using semantic web technology in multi-agent systems: a case study in the taga trading agent environment. In *Proceedings of the 5th international conference on Electronic commerce*, pages 95–101. ACM.

Trocas Sociais em Sistemas Multiagentes: Transferência de Confiança com Base na Reputação e na Relação de Dependência

Yunevda E. León Rojas[1], Graçaliz Pereira Dimuro[1], Diana F. Adamatti[1]

[1]Mestrado em Engenharia de Computação

Centro de Ciências Computacionais
Universidade Federal do Rio Grande (FURG) – RS – Brazil
Av. Itália Km 8, Campus Carreiros – 96.201-900 – Rio Grande – RS – Brazil
yuniekita@hotmail.com,{gracaliz, dianaada}@gmail.com

Abstract. *In this article, agents interactions will be analyzed in a share environment by other agents, social exchanges using as a case of study "Horta San Jeronimo" that is a self-sustained urban Horta, localized at Sevilla, Spain. The studies will be supported by the confidence transfer based in the dependence reputation.*

Resumo. *Nesse artigo serão analisadas as interações dos agentes em um ambiente compartilhado por outros agentes, produto das trocas sociais, utilizando como estudo de caso da "A Horta San Jerónimo", que é uma horta urbana de auto-consumo, localizada na cidade de Sevilla, Espanha. Os estudos serão sustentados na transferência de confiança com base na reputação e na relação de dependência.*

1. Introdução

As pesquisas realizadas por [FARIAS, 2012], [SANTOS, 2014] e [SCHMITZ, 2011] tratam isoladamente de sistemas multiagentes, trocas sociais e confiança e reputação. Todos estes trabalhos foram desenvolvidos sobre o *framework* JaCaMo e são focados na avaliação dos valores de trocas sociais no recebimento de serviços, na integração de artefatos organizacionais e nas crenças do grupo e dos agentes. Dessa forma, gera-se a possibilidade de ampliar o conhecimento sobre a interação entre os agentes quando se une os conhecimentos desenvolvidos por cada um deles.

Observa-se que os sistemas multiagentes, utilizados geralmente para modelar e simular sistemas complexos, estão formados por agentes autônomos que, por um lado têm a capacidade de interagir entre si e com o ambiente onde se encontram, e por outro, tem a capacidade de observar e comunicar-se atuando de modo cooperativo a fim de atingir metas ou objetivos. Assim, cada agente tem existência própria independente dos demais agentes.

Segundo [RUSSEL e NORVIG, 1995], os sistemas multiagentes podem ser reativos ou cognitivos. Os sistemas reativos são formados por um grande número de agentes, os quais são caracterizados por não possuírem conhecimento do ambiente e não utilizarem raciocínio lógico, vivendo no "aqui e agora". Os sistemas multiagentes

cognitivos tipicamente possuem poucos agentes, os quais têm uma representação explícita de seu ambiente e dos outros agentes da sociedade. Nesse modelo, cada agente possui memórias do passado e comunicação de modo direto com os demais, sendo estes racionais, com alguma representação de seu conhecimento e objetivo [OLIVEIRA, 1996], [FARIAS, 2012]. Na literatura, mostra-se que certas características dos agentes envolvidos em trocas sociais são mais adequadamente tratadas com agentes cognitivos.

Dentro dos sistemas multiagentes, está a capacidade dos agentes de agir socialmente, ou seja, interagir. Estas interações, chamadas de trocas, podem ser definidas, com base na teoria de Piaget. Segundo [PIAGET, 1995], troca social é qualquer sequência de ações entre dois sujeitos, tal que um dos sujeitos, pela realização de suas ações, preste um serviço para o outro. Como se pode verificar, cada interação entre agentes é representada como uma troca chamada troca de serviço, a qual posteriormente poderá ser avaliada pelos agentes que interagem, gerando assim o conceito de valor de trocas sociais. Os valores de trocas envolvidos são: investimento, satisfação, débito e crédito e podem ser de natureza qualitativa, subjetiva, imprecisa, ambígua ou incompleta. Além disso, podem ser influenciados por diferentes aspectos relacionados à natureza interna dos agentes e a efeitos externos do ambiente [FARIAS, 2012]. Tais características de valoração das trocas podem ser abordadas por conceitos da lógica *fuzzy*. Esta trabalha com informações imprecisas, com a finalidade de tomar decisões e se sustenta na teoria dos conjuntos *fuzzy*, nas funções de pertinência e nas suas álgebras em geral. A palavra "*fuzzy*", de origem inglesa, significa incerto, vago, impreciso, subjetivo, difuso, etc. A teoria dos Conjuntos *Fuzzy* foi introduzida em 1965, pelo matemático Lotfi Asker Zadeh, com o fim de dar tratamento matemático a certos termos, como, "aproximadamente", "em torno de", dentre outros [BARROS e BASSANEZI, 2006].

Das interações mencionadas no parágrafo anterior, tem-se o seguinte exemplo de trocas entre dois agentes α e β, conforme a teoria de Piaget. Foi usada uma representação por dois estágios de trocas. No primeiro estágio Iαβ o agente α realiza um serviço para o agente β e os valores de trocas envolvidos neste estagio são os seguintes: rIαβ, o valor do investimento realizado por α para a realização de um serviço para β, que sempre tem um valor negativo; sIβα, o valor da satisfação de β com o serviço realizado por α; tIβα, o valor do débito de β para com α por sua satisfação com o serviços realizados por α; e vIαβ, o valor do crédito que α adquire de β por ter realizado o serviço. No segundo estágio, IIαβ, o agente α solicita a β a realização de serviço em pagamento pelo serviço realizado anteriormente (no caso em que α tem crédito), e os valores relacionados com este estágio de troca são análogos aos dos estágios de tipo Iαβ.

rIαβ, sIβα, rIIβα e sIIαβ são denominados valores materiais (associado imediatamente a uma troca realizada). tIβα, vIαβ, tIIβα, vIIαβ são conhecidos como valores virtuais (valores associados a trocas postergadas, que podem ser negociados em futuro próximo).

Durante o processo de interações entre os agentes existe a necessidade de reconhecer o parceiro de interação. Desse modo, é preciso saber como realizar uma avaliação das informações, baseada determinadas regras, que irão influenciar no comportamento dos agentes e permitir a eles assumir diferentes estratégias com relação às trocas que podem optar por efetuar. Essas regras estarão apoiadas na transferência da

confiança com base nos conceitos da reputação e da relação de dependência no contexto das trocas sociais em sistemas multiagentes.

Conforme [CASTELFRANCHI e FALCONE, 2001], o termo confiança é definido como um estado mental que apresenta ingredientes mentais, tais como as crenças e os objetivos. Também se pode dizer que é uma atitude e uma relação social entre os agentes envolvidos. Dentro do processo da confiança, os agentes têm que ser cognitivos, dotados de objetivos e crenças específicos. Como a confiança é um processo dinâmico que influencia as crenças de cada um dos agentes envolvidos, esse processo gera um sistema de delegações entre eles, permitindo que alguns agentes alcancem seus objetivos por meio de atitudes e habilidades dos outros agentes, em um Sistema Multiagentes.

De acordo com [SCHMITZ, 2011], a confiança ocorre por uma relação entre agentes. Logo, as crenças adquiridas destas interações são apenas de conhecimento dos agentes envolvidos. É baseado nessas crenças que o agente decide renunciar ou não um objetivo específico. Em [CASTELFRANCHI e FALCONE, 2001], foram estudados diferentes tipos de crenças. As consideradas como básicas da confiança são: a crença na "competência", em que um agente acredita que o outro agente tem todas as capacidades, habilidades e funções, ou seja, a capacidade necessária para obter o resultado esperado, alcançar o objetivo; a crença na "disposição", em que um agente acredita que o outro fornecerá o que ele precisa.

Por outro lado, a reputação trata da relação de um grupo para com um agente. Sendo assim, as crenças deixam de ser definidas como do agente e passam a ser definidas como crenças do grupo. As reputações são definidas como um coletivo de crenças e opiniões que influenciam as ações dos indivíduos em relação aos seus pares. A reputação é definida como o núcleo das impressões compartilhadas por uma rede social. A reputação pode ainda ser vista como uma ferramenta social com o objetivo de reduzir a incerteza de se interagir com indivíduos de atributos desconhecidos [BROMLEY, 1993].

Portanto, observa-se que a abordagem dos conceitos relacionados à transferência da confiança e da reputação no processo das trocas sociais entre agentes possui características diferentes e, de certa forma, complementares, com diferentes enfoques na avaliação de trocas de serviços entre os agentes. A lógica *fuzzy* se mostra bastante eficiente para efetuar análise dos resultados do processo dessas trocas de serviços, facilitando o estudo desses dados, que geralmente são incertos e subjetivos, por dependerem da confiança e da reputação existente entre os agentes ou grupos de agentes envolvidos. Consequentemente, a confiança encontra-se na crença individual e a reputação esta na crença de grupo [HERZIG, LORINI, *et al.*, 2008].

2. Cenário

O cenário para o desenvolvimento dos sistemas multiagentes vai ser o JaCoMo, que é um *framework* para programação Multiagentes, que integra três tecnologias diferentes: (a) *Jason*, que permite a programação de agentes autônomos, é um interpretador para uma versão estendida do *AgentSpeak*, o qual é uma linguagem de programação orientada a agentes que baseia-se em eventos e ações, fornecendo uma plataforma para o desenvolvimento de sistemas multiagentes com diversas características; (b) *Cartago*,

que permite representar artefatos do ambiente de programação, tornando possível programar e executar ambientes virtuais; e (c) *Moise+*, que será usado para programar organizações multiagentes, baseadas em noções como papéis, grupos e missões, permitindo que sistemas multiagentes tenham especificações explícitas de sua organização [JaCaMo Project, 2014].

Este estudo será contextualizada no ambiente "A Horta San Jerónimo", que compreende uma horta urbana de auto-consumo mantida por seus próprios usuários, localizada na cidade de Sevilla, Espanha, onde os processos de trocas de serviços entre os participantes não envolvem valores monetários, se caracterizando por permitir e promover uma série de interações e trocas sociais entre os participantes, baseadas em normas dadas pela assembleia da comunidade [SANTOS, 2014].

3. Proposta de Trabalho

O presente trabalho apresenta uma proposta de estudo que estende as pesquisas feitas por [FARIAS 2012], [SANTOS 2014] e [SCHMITZ 2011], interligando os temas desenvolvidos pelos mesmos: trocas sociais, sistemas multiagentes e reputação, respectivamente. Como foi mencionado anteriormente, o processo das trocas sociais gera valores de trocas, os quais tem dependência recíproca e, além disso, tem influência na realização das trocas futuras, assim, a transferência de valores entre os agentes influenciarão na confiança e na reputação, e vice-versa. Isto ocorre porque a reputação é um fator envolvido na construção da confiança e a confiança apresenta-se como uma relação entre agentes que tem como parte de seu estado mental crenças e objetivos. Pela mesma razão, o conhecimento do grau da confiança e reputação envolvido no processo das trocas, vai prover relações mais seguras entre os agentes e credibilidade de seus pares. Consequentemente, o presente trabalho procurará abstrair o conceito das trocas sociais dentro dos sistemas multiagentes, tentado saber quanto e como é a influência da confiança e da reputação na transferência de valores na realização dos processos das trocas sociais em sistemas multiagentes.

4. Conclusão

Este artigo discute os conceitos de trocas sociais, confiança e reputação dentro de sistemas multiagentes, analisando as relações de dependência entre agentes e levando à possibilidade de analisar trocas sociais influenciadas pela reputação e pela confiança. Essa análise é feita por meio de uma formalização teórica, com definição do conceito de regras para a realização de trocas de serviço, com o objetivo de prever o comportamento futuro dessas interações e seus agentes.

Referências

BARROS, L. C.; BASSANEZI, R. C. **Tópicos de Lógica Fuzzy e Biomatemática**. Campinas: IMECC, 2006.

BROMLEY, D. **Reputation, Image and Impression Management**. [S.l.]: John Wiley & Sosns Ltd, 1993.

CASTELFRANCHI, C.; FALCONE, R. Social Trust: A Cognitive Approach. Roma: National Research Council - Institute of Psychology, 2001. p. 55-90.

FARIAS, G. **Um Modelo de Agentes BDI- Fuzzy para Trocas de Serviços Não - Econômicos com Base na Teoria das Trocas Sociais**. UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG. Rio Grande, p. 119. 2012.

HERZIG, A. et al. Prolegomena for a logic of trust and reputation. In 4th For-Trust Meeting, 2008.

JACAMO Project. **Multi-Agent Programming Framework**, 2014. Disponivel em: <http://jacamo.sourceforge.net/>. Acesso em: 26 março 2014.

OLIVEIRA, F. **Inteligência Artificial Distribuída**. Canoas,RS: IV ESCOLA REGIONAL DE INFORMÁTICA, 1996.

PIAGET, J. **Sociological Studies**. 1a. ed.. ed. London: Routlege, 1995.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence:** A Modern Approach. 1a edition. ed. [S.l.]: Prentice-Hall, 1995.

SANTOS, F. **Um Sistema Multiagente Multidimensional para Simulação de Processos de Produção e Gestão Social de um Ecossistema Urbano: uma abordagem baseada na Integração de Artefatos Organizacionais, Normativos, Físicos e de Comunicação no JaCaMo**. Universidade Federal de Rio Grande. Rio Grande, p. 116. 2014.

SCHMITZ, T. L. **Crenças de Grupo Como Instrumento de Formação da Reputação: Uma Arquitetura Baseada em Agentes e Artefatos**. Universidade Federal de Rio Grande. Rio Grande, p. 92. 2011.

Fragmenting Prometheus: new choices for developing multiagent systems

Daniela S. Yassuda, Sara J. Casare, Anarosa A. F. Brandão

Laboratório de Técnicas Inteligentes
Escola Politécnica – Universidade de São Paulo (USP).

{daniela.yassuda, anarosa.brandao}@usp.br, sjcasare@uol.com.br

Abstract. *The Medee Method Framework supports the development of multiagent systems based on a Situational Method Engineering approach. It involves a repository that stores method fragments which can be used to compose specific methods to each particular project situation. This work consists in fragmenting the agent-oriented method Prometheus following the Medee approach. In this paper, we present a brief explanation of the repository structure and the step-by-step process of fragmenting methods using Medee. Then, we illustrate how this process was applied to define Prometheus fragments: first, by describing how the method was modeled in SPEM elements; second, by discussing the decisions made in order to outline standardized activity, phase and process fragments.*

1. Introduction

The availability of tools, frameworks and methods to assist the development process is one of the factors that impact the adoption of multiagent systems by the software industry. While the multiagent paradigm is a fitting solution for various problems, the supporting assets for the design and implementation of said projects are still evolving.

The Medee Method Framework [2] aims to provide this kind of support for developers who work with multiagent systems. It allows them to build customized methods on demand according to their project characteristics. This is based on Situational Method Engineering concepts, which is a section of Method Engineering [1] that focuses on these tailored solutions to each given situation. They are called situational methods. In order to achieve that, Medee stores pieces of agent-oriented methods, such as Tropos [4] and Gaia [9], as well as organizational models, like MOISE+ [6]. Each one of these pieces is called a fragment. Fragments are standardized and represent a coherent part of a method that can be used and re-used to solve entire problems or parts of them.

The scope of this paper is the fragmentation of Prometheus [8] following the Medee Delivery Process, which is part of the Medee Method Framework. It consists of an initial step towards the use of Prometheus in situational approaches for developing multiagent systems. The next section will further detail the Medee Repository and explain the Delivery Process. Then, the fragmentation itself will be described by indicating which tools and resources were used, how Prometheus tasks and work products were depicted and discussing the aspects taken in consideration while outlining the fragments.

2. Medee Method Framework

The Medee Method Framework is composed of a method repository, a composition model and a delivery process. Here, we briefly explain the Medee Repository and the Medee Delivery Process, which is the process to guide populating the repository.

2.1 Medee Method Repository

The Medee Method Repository is constituted by three layers: Medee Elements Pillar, which stores method elements and methods "as is"; Medee Fragments Pillar, storing the fragments themselves; and Medee Methods Pillar, that keeps Medee AOSE Methods and Medee Situational Methods.

The first layer is the Elements Pillar, the foundation of the repository. It keeps method elements, which are SPEM elements – tasks, roles, work products, guidance and categories - that synthetize the content of each method or organizational model. Using said elements, it is possible to describe the entire methods with SPEM notation: these are called Methods As Is, also stored in this pillar.

The second layer stores the method fragments. The elements of the first pillar go through a standardization process, then they are assembled to build fragments. In Medee, fragments can cover five different disciplines - Requirements, Analysis, Design, Implementation and Test – and they have four different granularities – Activity, Phase, Iteration and Process.

The Medee Methods Pillar stores two kinds of methods: AOSE Methods, which are complete agent-oriented methods, and Situational Methods, which are custom-made according to project factors specified by the user. All these methods are built using Medee Fragments from the second pillar.

2.2 Medee Delivery Process

The Medee Delivery Process is the step-by-step specification on how to obtain method elements, method fragments and situational methods by populating all three pillars of the repository. In order to fragment Prometheus, we followed the first two phases of the Medee Delivery Process: Medee Method Element Capture (MMEC) Phase and Medee Method Fragment Elaboration (MMFE) Phase.

During the first phase, which aims to capture method elements to fill the first pillar of the repository, the agent-oriented method is studied, analyzed and modeled in terms of SPEM elements. All content is outlined and then detailed in form of tasks, work products, roles, guidance and categories. These method elements are used to build the activities and phases that constitute the Methods As Is, which are full methods represented in SPEM elements.

The Medee Method Fragment Elaboration Phase involves the fragmentation itself. It is necessary to standardize the method elements from the first pillar before using them to build fragments. The concern regarding making the elements and fragments uniform is due to the reusability aspect of the framework. It is expected that the fragments can be reused and recombined with fragments sourced from other methods. After that, the fragmentation starts from the smaller granularity to the bigger one. First we obtain Activity fragments, then Phase and Iteration, and, lastly, a Process fragment.

The Delivery Process has a third phase, the Medee Situational Method Composition Phase, that discusses how to build situational methods with Medee using project factors and either a top-down or a bottom-up fragment composition approach, but it is out of the scope of the paper. An interested reader can find more information in [2].

3. Fragmenting the Prometheus method

The fragmentation process is described through the two phases of the Medee Delivery Process that were performed during this work: the capture of Prometheus method content in terms of SPEM elements and the elaboration of the fragments themselves. We present a brief introduction to Prometheus, the tools used during the process and considerations about the decisions made to get the fragments. Due to space limitations, we will describe only one fragment of each discipline or component.

3.1 First layer – Capturing Method Elements

The Medee Delivery Process starts by understanding the agent-oriented method that will be fragmented. Prometheus is a method that focuses on supporting the development of goal and plan-oriented agents, though it is not limited to this approach. It is intended to be a practical method, providing detailed steps and guidance to help perform each task. Prometheus has three phases: the System Specification Phase, the Architectural Design Phase and the Detailed Design Phase.

After studying the method, the next step is to outline the method content using SPEM elements, with the tool support of the Eclipse Process Framework (EPF) Composer [4], which allows the authoring of methods with various development cycles and applications. We start the MMEC Phase capturing the method elements by modeling Prometheus tasks, roles, work products and guidance with EPF Composer, first by just sketching them out and later by detailing their descriptions and setting up all the links between them. With these method elements, it was possible to assemble the entire method together with its phases, activities and tasks, therefore mounting the Prometheus As Is method. Figure 1 shows the thirteen tasks and twenty-one work products captured.

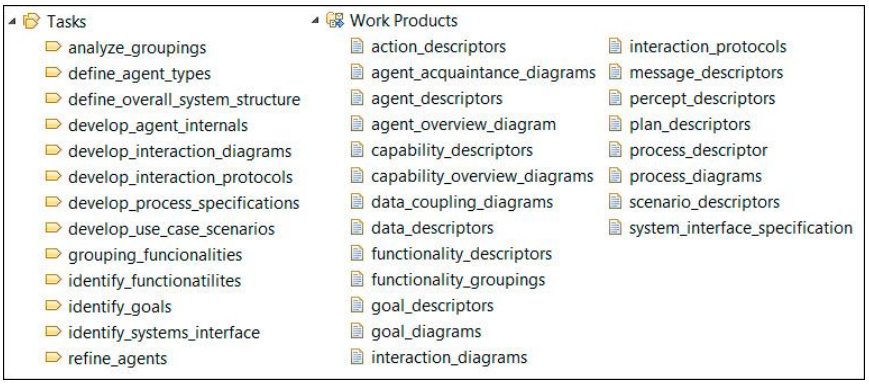


Figure 1. Prometheus tasks and work products captured during MMEC Phase.

The System Specification Phase in Prometheus As Is has four activities: *Identify Goals*, *Identify Functionalities*, *Identify System's Interface* and *Develop Use Case Scenarios*. The Architectural Design Phase contains four activities as well. There are three activities related to the definition of agent types that will exist within the system:

Grouping Functionalities, *Analyze Groupings* and *Define Agent Types*. During this phase, there is also the *Define Overall System Structure* activity and the *Develop Interaction Specification* activity. The last phase, Detailed Design, contains the following activities: *Develop Agent Internals*, *Develop Process Specifications* and *Refine Agents*. The activities involving agents include the design of its internals by using capabilities and by developing plans.

Most of the resulting work products from each task are either descriptors or diagrams. The guidance includes concepts, descriptors templates, diagram examples and a tool mentor for the Prometheus Design Tool (PDT). As for the roles, we chose to already use the Medee Roles, since they fit the specifications of Prometheus.

3.2 Second layer – Elaborating Method Fragments

The next phase of the Delivery Process, the MMFE Phase, concerns the elaboration of fragments. We use the method elements captured previously as a foundation. While crafting fragments, it is possible to extend and revise each element, as well as creating new elements if the fragmentation calls for so. The Prometheus As Is method can be used as a reference to elaborate fragments and check for consistencies.

The tasks and work products have to go through a standardization process, so all the content stored in Medee can be uniform and follow the same guidelines. In Medee Framework, all task names start with MTV (Medee Task Variability) and work product names start with MPV (Medee Product Variability). Fragment names, as seen further in this sub-section, start with MMF (Medee Method Fragment) and explicitly state its source method name. Since we already used the Medee Role Set during the first phase, there is no need to change them.

We start with the activity fragments. Then, we use these fragments to build phase fragments and finish by building a process fragment. It is necessary to have in mind that Medee Method Fragments have particular classifications that aim to ease the situational method composition process. They can cover one of the five process disciplines: Requirements, Analysis, Design, Implementation and Test. Regarding the MAS component they focus on, Medee has four categories, which are Agent, Interaction, Environment and Organization. Prometheus covers the Requirements, Analysis and Design disciplines, as well as Agent, Interaction and Environment components.

From the System Specification Phase of Prometheus, fragments of Requirements and Analysis disciplines are obtained. An example is MMF Identify Requirements with Prometheus. The activity consists of one task, MTV Identify Goals, whose aim is to describe the system's goals and sub-goals using descriptors and diagrams. These are encapsulated in one work product, MPV Goal Specification. Since descriptors and diagrams of the same subject complete each other, they were grouped together to facilitate the understanding of the task.

An important characteristic of Prometheus is that the developer is encouraged not to perform tasks sequentially. Instead, the method proposes to keep refining work products in an iterative manner, always using the knowledge and suggestions obtained while performing other tasks. This aspect is noticeable in this fragment: the task has multiple optional inputs, to prompt the developer to look for cues in other work products (Figure 2).

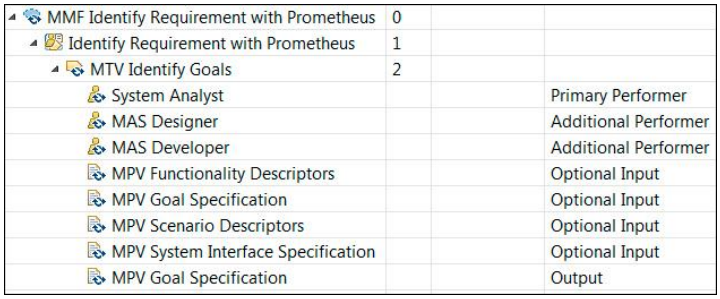


Figure 2. Consolidated view of MMF Identify Requirements with Prometheus.

The MMF Analyze Agent with Prometheus includes the System Specification task MTV Identify Functionalities, as well as two tasks sourced from the original Architectural Design Phase, MTV Group Functionalities and MTV Analyze Groupings. Since the identification of system's functionalities involves the analysis of the goal specification and also takes in consideration information from the use case scenario, this task was placed in the Analysis discipline instead of the Requirements one. The functionalities are also used to define the agent types that will be developed. This is done during the MTV Group Functionalities task, whose outcomes are diagrams that depict various possible designs. Said designs are evaluated during MTV Analyze Groupings, resulting in a list of agent types, each one encapsulating a set of functionalities.

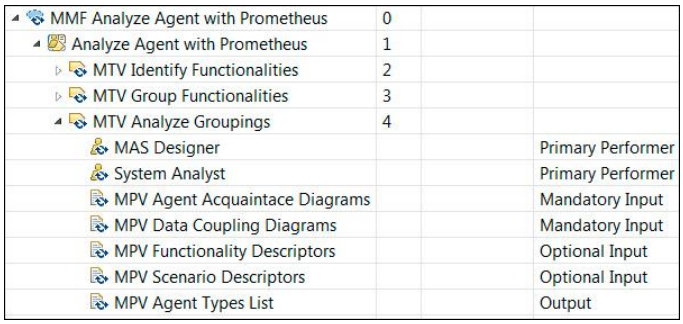


Figure 3. Consolidated view of MMF Analyze Agent with Prometheus.

During the MMF Design Interaction with Prometheus, the MAS Designer takes the use case scenarios from the Requirements phase to scheme the interaction between agents in form of MPV Interaction Protocols, MPV Interaction Diagrams and MPV Message Descriptors. These three work products contribute to the same specification, so they are grouped together in the MPV Interaction Design artifact.

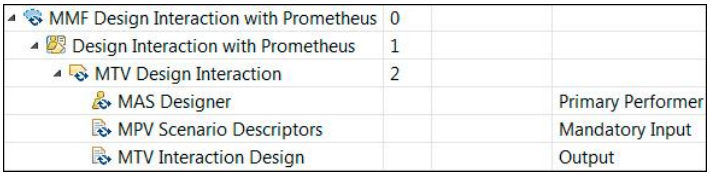


Figure 4. Work breakdown of Design activity fragments.

4. Conclusion

The described process resulted in twelve tasks, thirty-one work products, twenty-five guidance, seven activity fragments, three phase fragments and one process fragment

sourced from Prometheus. For now on, these fragments can be used to compose situational methods for multiagent systems. The fragments related to the Design discipline and Agent component are the most detailed of them, which is coherent with the characteristics of Prometheus. Due to limited space, it was not possible to display in detail all fragments with their respective tasks, inputs, outputs, assigned roles and guidance. However, they will be available in the Medee website[1]. The next step is to evaluate the fragments using the Medee Improvement Cycle [3] which allows to continuously improving fragments by submitting them to experts' assessment. In this case, we can involve Prometheus experts, MAS developers and Method engineers. Besides that, the practical use of the fragments in project situations will tell how they can be improved.

Acknowledgements

Daniela Yassuda is partially supported by grant #013/14642-4, São Paulo Research Foundation (FAPESP). Anarosa A. F. Brandão is partially supported by grant #010/2640-5, São Paulo Research Foundation (FAPESP).

References

1. Brinkkemper, S. 1996. Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology, vol. 38 (4), 275-280
2. Casare, S. J. ; Brandão, Anarosa A.F.; Guessoum, Z. ; Sichman, J.S. 2013. Medee Method Framework: a Situational Approach for Organization-Centered MAS. Autonomous Agents and Multi-Agent Systems (Dordrecht. Online), v. tbd, p. 1-44, 2013. (http://dx.doi.org/10.1007/s10458-013-9228-y).
3. Casare, S.J. 2012. *Medee: A Method Framework for Multiagent Systems*, PhD Thesis, Universidade de São Paulo, Brazil.
4. Giorgini, P. et al. 2004. The Tropos Methodology. In: Bergenti, V; Gleizes, M. P.; Zambonelli, F.(Ed.), *Methodologies and software engineering for agent systems*, Kluwer Academic Publishers, pp. 89-106.
5. Haumer, P. 2007. Eclipse Process Framework Composer – Part 1 – Key Concepts. Available at: <http://www.eclipse.org/epf>.
6. Hubner J., Sichman J., Boissier O. 2002. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Bittencourt, G. & Ramalho, G. L. (Eds.), *16th Brazilian Symposium on AI, SBIA'02*, LNAI 2507, Berlin: Springer, p. 118-128
7. OMG. (2008). Object Management Group. Software & Systems Process Engineering Meta-Model Specification, version 2.0. OMG document number: formal/2008-04-01. http://www.omg.org/spec/SPEM/2.0/PDF.
8. Winikoff, M. & Padgham, L. (2004) The Prometheus Methodology, In Bergenti,F. Gleizes, M-P, Zambonelli, F. (Eds) Methodologies and Software Engineering for Agent Systems - The Agent-Oriented Software Engineering Handbook, Kluwer Acad. Publ.
9. Zambonelli F., Jennings N. R., Wooldridge M. 2003. Developing multiagent systems: The Gaia methodology. *ACM Transaction on Software Engineering and Methodology*, vol 12(3), 417-470

[1] http://medee.poli.usp.br/

Extending JaCaMo for organizational interoperability

Tomas M. Vitorello, Fabio T. Muramatsu, Anarosa A. F. Brandão

Computer Engineering and Digital Systems Department
Escola Politécnica – Universidade de São Paulo (USP)
CEP – 05508-900 – São Paulo – SP – Brazil

{tomas.vitorello, fabio.muramatsu, anarosa.brandao}@usp.br

Abstract. *Traditionally organization-centered multiagent systems (OC-MAS) are strongly dependent on their own organizational infrastructure. This dependence is one of the main concerns when trying to achieve interoperability among them. In this paper, we investigate the possibility of addressing the issue by interpreting the organization as a set of norms imposed upon the agents. This work is based on the JaCaMo infrastructure, which already implements the Moise+ organization model through this approach. Our aim is to extend this platform to support other organizational models, effectively allowing agents to interoperate among organizations described according to different organization models.*

1. Introduction

The development of open Multiagent Systems (MAS) faces a natural difficulty when the aim is to achieve a collective goal through the interaction of self-interested, autonomous agents [Wooldridge 2009]. In this context, the adoption of an Agent-Centered MAS (AC-MAS) revealed to be problematic, leading to the proposal of exploiting organizational aspects of such systems to promote coordination and cooperation towards the system's goals [Ferber et al 2004][Hübner et al 2002]. Following this trend, many organizational models were proposed to develop the so-called Organizational-Centered MAS (OC-MAS) [Dignum 2004][Ferber et al 2004][Hübner et al 2002], which were implemented based on a particular Organization Infrastructure (OI). However, this approach turned out to be restrictive, meaning that agents got heavily dependent on a particular OI. This raises the problem of agent interoperability among different organizations, which can be defined as the inability of agents designed to run in a particular organization to perform tasks in a different one. In this paper, we investigate the possibility of addressing this issue by extending the JaCaMo infrastructure [Boissier et al 2013], which already implements the Moise+ organizational model with artifacts, following the ORA4MAS approach [Hübner et al 2010] as an alternative to the OI, in order to incorporate other organizational models such as AGR [Ferber et al 2004] and OperA [Dignum 2004].

This article is organized as follows. Section 2 briefly explains the models and implementations upon which this paper is based. Section 3 presents a simplified model of Moise+ implementation using artifacts, and its proposed extension to include support to AGR and OperA, as a sequence of our previous work in [Muramatsu et al 2014]. Finally, in Section 4 we discuss some preliminary results about the extension.

2. Background: JaCaMo, ORA4MAS and NPL

2.1. JaCaMo

JaCaMo is a platform for programming MAS that combines agent, organization and environment programming into a single framework. This approach is possible due to integration of several technologies: (i) Jason [Bordini et al 2007], a platform for programming MAS using an agent-oriented approach; (ii) CArtAgO [Ricci et al 2007] for an environment-oriented approach; and (iii) Moise+ [Hübner et al 2002], for an organizational-oriented approach. In addition, ORA4MAS provides the means to integrate Moise+ organizational model through organizational artifacts provided by CArtAgO framework.

During runtime, agents in JaCaMo have direct access to the environment, by means of manipulating artifacts in the same way as described in CArtAgO. Concurrently, their actions are monitored by an organization, modeled in accordance to Moise+ and implemented in the environment as proposed by ORA4MAS. An important feature in JaCaMo, which will be explored in this work, is the presence of a Normative Programming Language (NPL) interpreter within the ORA4MAS organizational artifacts to run the organization. This implies that the organization description must be translated into NPL before it is loaded by the organizational artifacts.

2.2. ORA4MAS

Traditionally, the implementation of OI is based on an architecture composed of services and special agents, located in a layer inaccessible to ordinary agents and dependent on its underlying organizational model [Coutinho et al 2007][Hübner et al 2010]. Although this approach successfully achieves its goals of running an OI, it has the disadvantage to be too strict and inflexible, as agents are required to know how the OI in which they are running is structured.

ORA4MAS (Organizational Artifacts for Multiagent Systems) was proposed as a solution to the limits imposed by the previously mentioned approach. Its goal is to make organizations more flexible by implementing it in a layer accessible to agents, exploiting artifacts to deal with aspects of the organizational model. It is important to stress that ORA4MAS solution is to move the required knowledge of the OI from the agents to the organizational artifacts.

The role of artifacts in this context is to provide operations and information regarding the organization to any agent that participates in it. For instance, if an agent wants to adopt a role, it must trigger the correspondent operation on the artifact. Moreover, it can easily get information about the organization state (for example, the available roles) by inspecting the artifact's observable properties. Organizational agents are proposed in ORA4MAS to deal with aspects that require reasoning. For instance, whenever an agent triggers a forbidden operation, one of two actions can be taken: regimentation or sanction. Regimentation immediately blocks the operation and recovers the state of the system. Sanction involves notifying this organizational agent about the operation, who in turn decides on what action to take.

2.3. Normative Programing Language (NPL)

ORA4MAS organizational artifacts have embedded within them an NPL engine in order to execute its functions. This language is, as the name implies, a programming language based on norms. In general, norms are composed of a statement describing an expected pattern of behavior and the consequences of disregarding it. This way, each norm can be considered an obligation to all agents it is imposed upon. In addition, the language utilizes facts, which are statements of information, and inference rules in order to track the system state. Therefore, NPL programs are composed of facts and inference rules plus a set of norms.

Utilizing the OM translations into NPL from our previous work, it should be possible to have JaCaMo working with organizational models other than Moise+. The next step is to conceptualize the structure of the organizational artifacts needed to run these normative programs, to reflect the features of each organizational model. Section 3 describe the process of modifying JaCaMo framework.

3. Extending JaCaMo

In this section, we will explore how it is possible to create a set of organizational artifacts to support AGR and OperA, based on the existent implementation of Moise+ artifacts. The idea is to check the possibility to extend JaCaMo with these artifacts, making it capable to run multiple organizations and have agents to interoperate among them. Our analysis of the Moise+ artifacts resulted in the simplified class diagram shown in Figure 1.

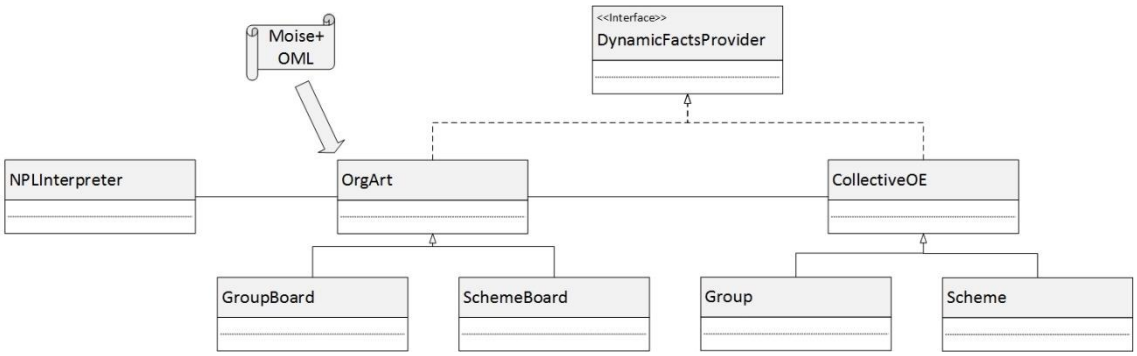


Figure 1: Class diagram showing a simplified structure of Moise+ artifacts

We notice from the class diagram that the organizational artifacts for Moise+ consist of two artifacts, one for each organizational dimension, modelled by the `GroupBoard` and `SchemeBoard` classes. Both artifacts inherit from the `OrgArt` class, which defines the general structure of an organizational artifact. As mentioned before, a normative engine is embedded in these artifacts to effectively run the organization in terms of norms. That's the role of the `NPLInterpreter` class, which compounds the structure of `OrgArt` and is triggered by any action to verify compliance with norms. This is the reason why there is no artifact dedicated to the Moise+'s normative dimension. Furthermore, the `CollectiveOE` class is also an important component of `OrgArt`. At runtime, several actions take place in an organization, leading to the generation of many dynamic facts. This class is responsible to define and store these dynamic facts, and answer to any queries regarding them. The `Group` and `Scheme` classes are specializations of this class, defining constructs specific to the `GroupBoard` and

SchemeBoard artifacts, respectively. Finally, the DymanicFactsProvider interface provides a model for the OrgArt and CollectiveOE classes to respond to the aforementioned queries related to dynamic facts.

In the current implementation, the OrgArt artifact receives a Moise+ description in its Organizational Modeling Language (OML). However, as we want organizational artifacts to deal with models other than Moise+, we have modified OrgArt to accept a description in NPL. Moise+ will remain using the original OrgArt, while other models will use this new version.

3.1. AGR

AGR is a simple organizational model that focuses on the concepts of Agent, Group and Role, as its acronym suggests. It allows the designer to define some constraints upon the structure of the organization (in terms of groups and roles, similarly to Moise+) and upon the communication of agents (in terms of interactions). A detailed description of this model can be found in [Ferber et al 2004].

A natural approach to convert the structure shown in Figure 1 to AGR might consist in adapting the GroupBoard, due to its similarity to the structural dimension of AGR, removing the SchemeBoard, since there is no such notion in this model, and creating a CommunicationBoard to deal with the communication constraints.

However, since this model restricts interactions to be placed only between agents enacting roles in the same group, it seems more convenient to implement this portion of the organization also in the GroupBoard. Therefore, we believe that a possible structure of AGR would consist only in a GroupBoard artifact, with the correspondent Group class with all the dynamic facts specified. Of course, proper norms dealing with AGR constructs should be loaded in this artifact's normative engine for it to work as predicts its model. Figure 2 illustrates the proposed changes for AGR.

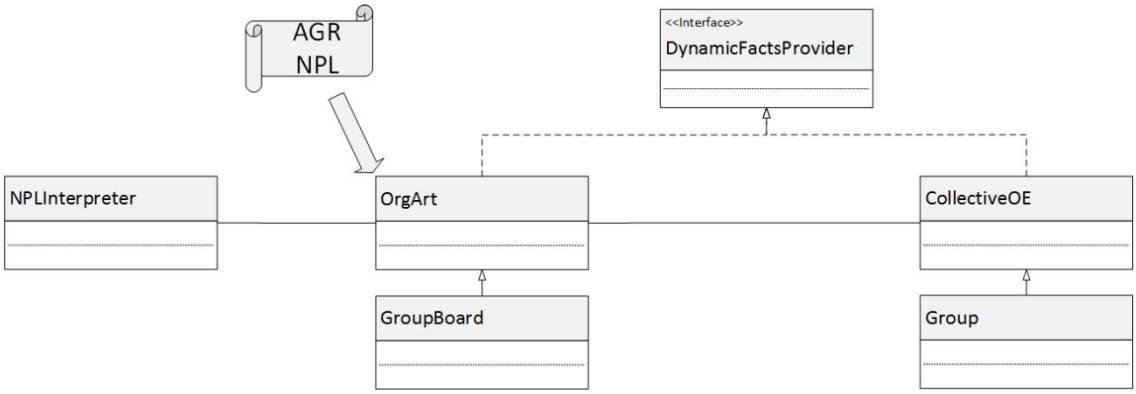


Figure 2: Class diagram showing a simplified structure of AGR artifacts

3.2. OperA

The OperA organizational model was designed to provide interaction and collaboration among agents immersed in the system without compromising autonomy between society design and agent design.

Due to the differences between Moise+ and OperA organizational model, several modifications have to be made in order to take the existing Moise+ artifacts' structure and transform it into one suitable for the OperA model.

One of the biggest changes is to the `GroupBoard` artifact as OperA's structural specification is not group centered as Moise+ or AGR are. However, most of the current functionality of the artifact should be preserved as there are several other structural characteristics shared between the models. Another major change is the replacement of the `SchemeBoard` by `SceneBoard` in order to implement OperA's functional structure based on scenes rather than schemes as Moise+ is.

These modifications to `GroupBoard` and `SchemeBoard` also imply that the `Group` and `Scheme` classes are not relevant to the model and new classes have to be created to provide the necessary constructs for the new artifacts. Finally, a `CommunicationBoard` class has to be implemented to deal with the communication aspects. Figure 3 illustrates the proposed changes for OperA.

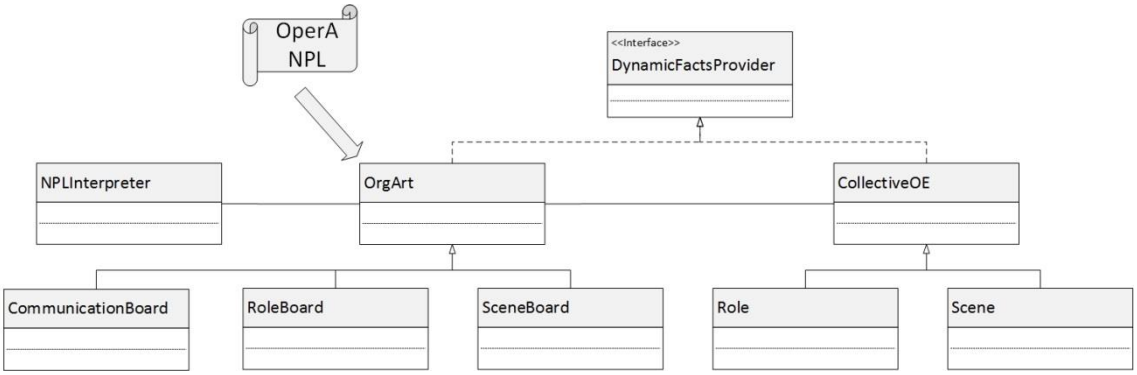


Figure 3: Class diagram showing a simplified structure of OperA artifacts

4. Discussion

Currently, our work is focused in analyzing at what extent we must change the artifacts' code in order to make it accept AGR or OperA. Fortunately, it is getting clear that thanks to the normative engine, organizational artifacts are not deeply dependent on Moise+ constructs, since they work upon the primitives of norms, facts and rules that are common to any normative program describing an organization model.

In the case of AGR, some changes in the `GroupBoard` have already been made to remove unused concepts (related schemes and sub-groups, for instance) and to add new ones, such as the notion of group structures and the dependence and correspondence rules between roles. As for OperA, the `GroupBoard` class was removed and some of its functions have been moved to another artifact (`RoleBoard`) that does not include the group concept. However, our work is still ongoing in these implementations.

Once we have completely developed these artifacts, we believe it should be possible to build MAS in which a single agent can interact with multiple organizations. This is because agents are no longer required to comprehend the organization structure, thus making their design independent of organizational models. The only knowledge agents are required to have in order to interact with any organization is how to handle CArtAgO artifacts and how to deal with obligations.

5. Conclusion

In this paper we presented an approach to provide organization interoperability by extending JaCaMo with the inclusion of other organizational artifacts. It is still ongoing work, since coding is at its very beginning. Nevertheless, since it is strongly based on the idea already implemented within JaCaMo for the Moise+ organizational model, we believe that we will succeed.

6. Acknowledgements

Tomas M. Vitorello is partially supported by grant 013/17973-1, São Paulo Research Foundation (FAPESP). Fabio T. Muramatsu is partially supported by grant #013/17948-7, São Paulo Research Foundation (FAPESP). Anarosa A. F. Brandão is partially supported by grant #010/2640-5, São Paulo Research Foundation (FAPESP).

7. References

[Boissier et al 2013] Boissier, O.; Bordini, R.; Hübner, J.; Ricci, A.; Santi, A. Multi-agent oriented programming with JaCaMo, Science of Computer Programming 78 (2013) 747–761.

[Bordini et al 2007] Bordini, R.; Hübner, J.; Wooldridge, M.; Programming Multi-Agent Systems in AgentSpeak using Jason, Wiley, 2007.

[Coutinho et al 2007] Coutinho, L.R.; Brandão, Anarosa A.F. ; Sichman, J.S. ; Boissier, O. . Organizational Interop-erability in Open Multiagents Systems - An Approach based on Metamodels and Ontologies. In: Proc.of the 2nd Workshop on Ontologies and Metamodels in Software Engineering WOMSDE 2007, 2007.

[Dignum 2004] Dignum. V. A Model for Organizational Interaction: based on Agents, founded in Logic. SIKS Dissertation Series 2004-1. SIKS, 2004. PhD Thesis.

[Ferber et al 2004] Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. In: AOSE'03. Volume 2935 of LNCS., Springer (2004) 214–230.

[Hübner et al 2002] Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: SBIA'02: 16th Brazilian Symposium on Artificial Intelligence. Volume 2507 of LNAI., Springer (2002) 118– 128.

[Hübner et al 2010] Hübner, J., Boissier, O., Kitio, R., Ricci, A., Instrumenting multi-agent organisations with or-ganisational artifacts and agents. In Auton Agent Multi-Agent Syst (2010) 20:369–400.

[Muramatsu et al 2014] Muramatsu, F.T, Vitorello, T.M and Brandão, A.A.F. Towards organizational interoperability through artifacts. In Proceedings of Environments for Multiagent Systems 10 years later - E4MAS 2014 (to appear)

[Ricci et al 2007] Ricci, A., Viroli, M., Omicini, A., CArtAgO: A Framework for Prototyping Artifact-Based En-vironments in MAS. In D. Weyns, H.V.D. Parunak, and F. Michel (Eds.): E4MAS 2006, LNAI 4389, pp. 67–86, 2007. Springer-Verlag Berlin Heidelberg 2007.

[Wooldridge 2009] Wooldridge, M. An Introduction to MultiAgent Systems, John Wiley & Sons, 2009.

Uma Ontologia para a Representação de Sistemas Multiagentes Culturais

Vinícius de Figueiredo Marques[1], Karen da Silva Figueredo[2]

Instituto de Computação – Universidade Federal do Mato Grosso (UFMT)
Cuiabá, Mato Grosso, Brasil

vini.type@gmail.com[1], karen@ic.com.br[2]

Abstract. *The inclusion of cultural elements in Multi-Agent Systems enables the sharing of behavior patterns by the agents in open systems, it promotes as well the coordination and cooperation between them. In order to support the modeling of these systems, this paper proposes an ontology that allows the system's designer to represent the multi-agent system and run logical assertions about it to verify the system's consistency, and therefore, to identify some conflicts at design time.*

Resumo. *A inclusão de elementos culturais nos Sistemas Multi-Agentes possibilita o compartilhamento de padrões de comportamento entre os individuos de sistemas abertos, promovendo a coordenação e cooperação. Para auxiliar na tarefa de modelagem de tais sistemas, este artigo propõe a utilização de uma ontologia que permita ao analista, além de representar o sistema, fazer asserções lógicas – verificando assim, a consistência do sistema e, por conseguinte, identificar possíveis conflitos em tempo de design.*

1. Introdução

Culturas podem ser entendidas como coleções de padrões de comportamento que são aprendidos, transmitidos e compartilhados com o objetivo de adaptar os indivíduos de um grupo [Keesing 1974]. Em Sistemas Multiagentes (SMA), sugere-se incluir os conceitos de cultura e seus elementos, tais como normas e valores, para promover a coordenação, cooperação e autonomia entre os indivíduos do sistema [Antunes 1997]. Por sua vez, a inclusão destes elementos aumenta a complexidade destes sistemas, tornando mais difíceis as atividades de *design*.

Uma ontologia é um artefato projetado para um permitir a modelagem do conhecimento acerca de algum domínio, real ou imaginado [Gruber 1993]. Através das ontologias é possível representar conceitos, propriedades e relações, bem como suas restrições lógicas.

Desta forma, este artigo propõe uma ontologia para a representação de SMA culturais que permite, além da modelagem do sistema com os elementos culturais, a realização de asserções lógicas para verificar a consistência do sistema e de consultas em geral a fim de analisar o sistema.

2. Ontologia para SMA Culturais

Conforme descrito anteriormente, uma ontologia define um modelo de representação de

um determinado domínio de conhecimento, incluindo seus conceitos, propriedades e relações. Nesta seção, apresentamos a ontologia proposta para descrever os conceitos de SMA contemplando os elementos culturais do sistema, as propriedades inerentes a estes conceitos e seus relacionamentos.

No escopo deste artigo, SMA Cultural é definido como um sistema que explicita a maneira que os agentes devem agir num sentindo mais amplo (preservando a sua autonomia), a fim de estabelecer a ordem social entre os agentes autônomos e fazer o sistema tender a uma solução desejada, mas sem predefinir especificamente o que os agentes farão. Um sistema normativo, ou seja, cultural, se esforça em estabelecer convenções para seus membros. Isto é vantajoso, pois aprendizagem social permite os indivíduos a imitar e aprender com outros, poupando-os do processo de teste e erro [Boyd 1985].

As normas num sistema normativo informam as atitudes que se espera que os agentes cumpram, além de prover incentivos à conformidade [McBreen 2011], por meio de sanções.

Para a criação da ontologia proposta, foi utilizada a ferramenta Protégé[1], que possibilita a definição dos conceitos da ontologia através de Classes que possuem propriedades que estabelecem relações e restrições. Através da ontologia proposta, o analista do SMA pode gerar instâncias da ontologia que representam o seu sistema, verificar a consistência do mesmo e realizar consultas lógicas (*queries*) sobre estas instâncias. A Seção 2.1 apresenta os conceitos da ontologia para SMA culturais e a Seção 2.2 explica como são feitas a verificação e análise dos modelos de instâncias criados a partir da ontologia proposta.

2.1. Conceitos da Ontologia

Os conceitos da ontologia proposta são baseados nos elementos componentes de SMA já consolidados e em elementos culturais já encontrados em pesquisas de SMA como cultura [Heinrich et al. 2011, Dechesne et al. 2013], normas [McBreen et al. 2011, Dechesne et al. 2013] e valores [Dechesne et al. 2013, Figueiredo e da Silva 2013].

Os conceitos da ontologia podem ser vistos na Figura 1 e são representados através das seguintes classes:

***Environment*:** ambiente em que as organizações e agentes do sistema habitam. Os ambientes podem possuir normas para restringir o comportamento dos agentes do ambiente.

***Organization*:** organização ou grupo de agentes. Uma organização pode ser composta de suborganizações, neste caso as suborganizações desempenham um papel na superorganização. As organizações possuem uma cultura e podem descrever normas para serem seguidas pelos agentes que a habitam.

***Agent*:** agentes do sistema. Os agentes podem habitar ambientes e organizações, onde por sua vez são membros de sua cultura e devem desempenhar papéis. Agentes

[1] http://protege.stanford.edu/

possuem objetivos, planos, crenças, valores e tem seu comportamento regulado por normas.

***Role*:** papéis que agentes e (sub)organizações podem desempenhar. Os papéis podem possuir crenças, objetivos, normas que restringem o comportamento do papel e subpapéis.

***Belief*:** crença que o agente possui sobre o sistema, os agentes do sistema e ele mesmo.

***Goal*:** objetivos que devem ser alcançados pelos agentes do sistema.

***Action*:** ações executadas pelos agentes do sistema. Uma ação pode promover e rebaixar valores do agente quando é executada.

***Plan*:** plano que define um conjunto de ações para alcançar um objetivo.

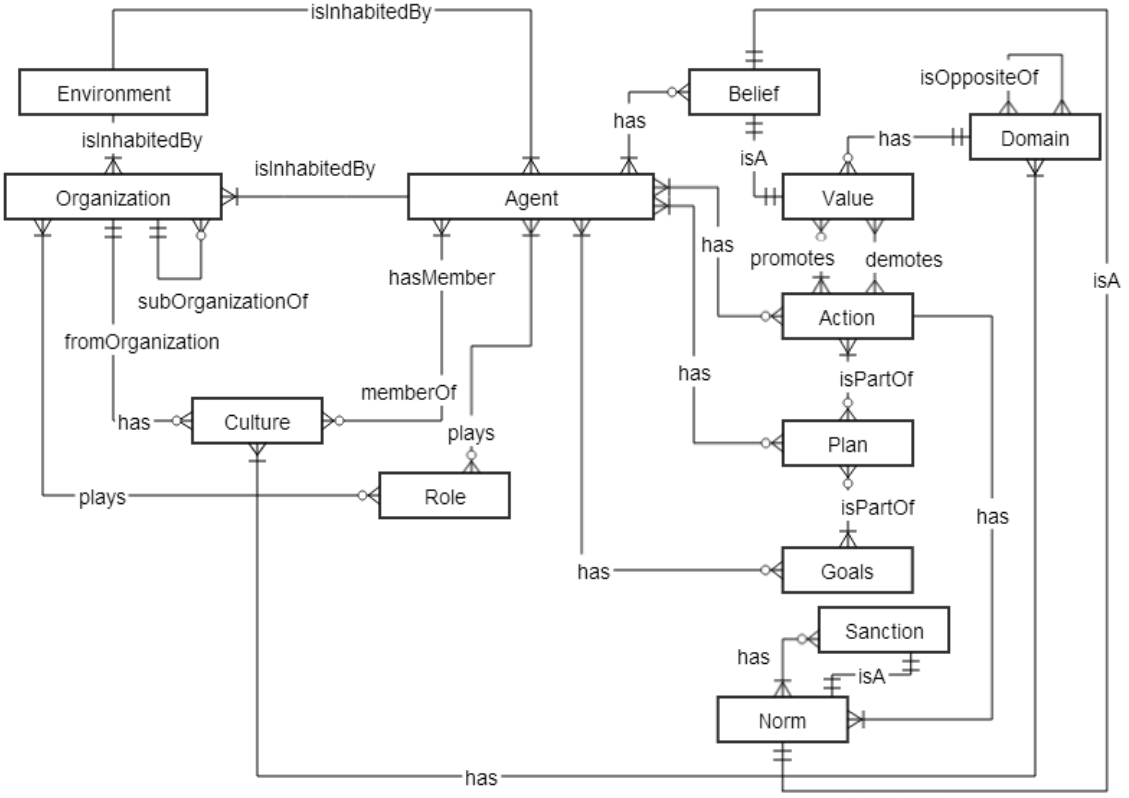


Figura 1. Ontologia para a Representação de SMA Culturais

***Culture*:** cultura construída pelas entidades de uma organização. A cultura compõe um sistema dinâmico com o intuito de garantir a sobrevivência das entidades e a perpetuação de crenças, valores e normas compartilhadas [Keesing 1974]. Os agentes são membros das culturas das organizações que habitam.

***Value*:** valor pessoal de um agente. Os valores são crenças ordenadas por uma importância relativa que guiam a seleção ou avaliação de comportamentos ou eventos [Schwartz e Bilsky 1987]. Ainda segundo a Teoria Universal dos Valores de Schwartz e Bilsky (1987), os valores podem ter um interesse individualista e/ou coletivista, devem pertencer a um domínio e serem dos tipos instrumentais (valores que podem ser

alcançados diretamente através da execução de ações) ou terminais (valores que podem ser alcançados indiretamente através dos valores instrumentais) [Rokeach 1979].

***Domain*:** domínio motivacional que estrutura um conjunto de valores. Os domínios são facetas de uma cultura e portanto suas estruturas e relações podem mudar de cultura para cultura. Os domínios podem agrupar valores individualistas, coletivistas ou mistos e possuem pelo menos um domínio em sua oposição, e.g. *entretenimento* é oposto aos domínios *maturidade* e *segurança* [Schwartz e Bilsky 1987].

***Norm*:** crença que regula ações dos agentes. As normas definem as ações que podem ser executadas pelos agentes (permissões), as ações que devem ser executadas pelos agentes (obrigações) e as ações que não devem ser executadas pelos agentes (proibições). Uma norma que está associada a um papel, se aplica a todos os agentes que desempenham este papel; uma norma que está associada a uma organização ou a um ambiente se aplica a todos os agentes que habitam aquela organização ou ambiente. As normas também podem estar associadas à sanções [Figueiredo 2011].

***Sanction*:** consequência que será aplicada quando uma norma for cumprida ou violada. No caso do cumprimento da norma a sanção é uma recompensa e no caso de uma violação a sanção é uma punição.

2.2. Verificação e Análise da Ontologia

Uma vez descritos os conceitos da ontologia com seus relacionamentos e restrições (e.g. Figura 2), o analista poderá construir o modelo do seu SMA na ferramenta Protégé instanciando as classes da ontologia proposta. Quando o analista tiver terminado o modelo, ele poderá verificar se todas as instâncias do modelo construído estão consistentes com todas as restrições e relacionamentos definidos pela ontologia através da opção *reasoner* da Protégé. Caso encontre alguma inconsistência, uma mensagem de erro será exibida (e.g. Figura 3).

Além de verificar a consistência, a opção *reasoner* também faz inferências sobre as instâncias do modelo a partir das propriedades da ontologia. Desta forma, o *reasoner* constrói automaticamente algumas relações, por exemplo: (i) se um papel possui uma crença e um agente desempenha este papel, então o agente também possui esta crença; (ii) se um agente habita uma organização que é suborganização de uma superorganização, então o agente também habita a superorganização; (iii) se um agente habita uma organização que possui determinada cultura, então o agente está inserido nesta cultura (ver Figura 4); etc.

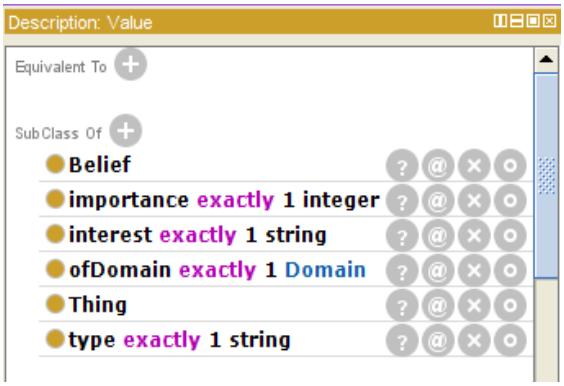


Figura 2. Exemplo de descrição de propriedades - Classe *Value*

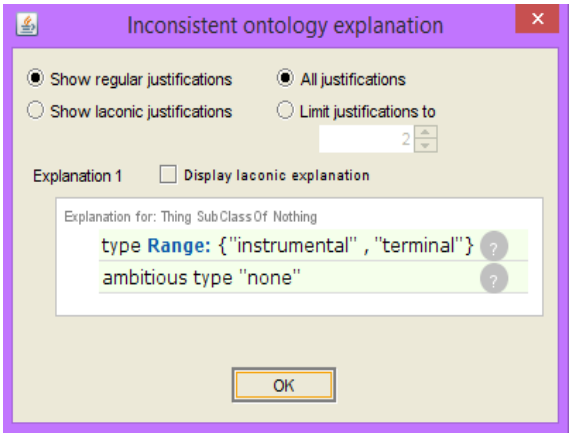


Figura 3. Exemplo de inconsistência: usuário criou uma instância de *Value* sem definir um tipo (*type*) instrumental ou terminal

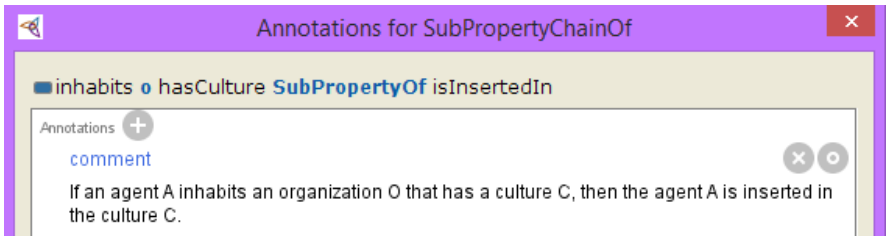


Figura 4. Descrição de uma propriedade da ontologia que será utilizada para inferências

Após a verificação de consistência e a realização de inferências automáticas pelo *reasoner*, o analista pode ainda executar consultas sobre as instâncias do seu modelo na opção *DL Query* da Protégé utilizando a *Manchester Syntax*[2]. Estas consultas são extremamente úteis para o analista, principalmente em modelos grandes com muitas instâncias quando é mais difícil estudar as relações e entidades do modelo. Utilizar as consultas é também uma abordagem interessante para fazer comparações e análises entre culturas e outros elementos culturais. Alguns exemplos de consultas possíveis são:

- *Value and ofCulture value CULTUREINSTANCE1 and (Value and ofCulture value CULTUREINSTANCE2)* – Retorna todos os valores compartilhados entre duas culturas específicas (CULTUREINSTANCE1 e CULTUREINSTANCE2);
- *Belief and ofAgent value AGENTINSTANCE and ofCulture value CULTUREINSTANCE* – Retorna todas as crenças (incluindo normas e valores) compartilhadas por um agente (AGENTINSTANCE) e determinada cultura (CULTUREINSTANCE);
- *Value and ofDomain some Domain and isOppositeTo value DOMAININSTANCE* – Retorna todos os valores de domínios opostos a um determinado domínio (DOMAININSTANCE);
- *Action and promotes value VALUEINSTANCE* – Retorna todas as ações que promovem determinado valor (VALUEINSTANCE).

[2] http://www.w3.org/2007/OWL/wiki/ManchesterSyntax

3. Conclusão

Como foi dito anteriormente, o uso da ontologia é importante para representar domínios de conhecimentos e, através da sua conceituação padrão a intercambiabilidade de modelos entre analistas é facilitada. A proposta desse trabalho é auxiliar a tarefa de representação e análise de SMA culturais através da ontologia apresentada e suporte da ferramenta Protégé. Em nosso trabalho mostramos como é possível criar modelos de instâncias da ontologia, verificar suas consistências e realizar consultas a fim de comparar e estudar os SMA representados.

Este trabalho está em seu estágio inicial, esperamos como próximos passos estender a ontologia incluindo mais conceitos e propriedades e implementar uma ferramenta que faça a verificação de conflitos (por exemplo entre normas, valores, etc) para as instâncias modeladas através da leitura de arquivos do tipo *.*owl* criados pela ferramenta Protégé.

Referências

Antunes, L. (1997) "Towards a model for value-based motivated agentes", In: Proceedings of MASTA97.

Boyd, R. (1988). "Culture and the evolutionary process". University of Chicago Press.

Dechesne, F., Di Tosto, G., Dignum, V. and Dignum, F. (2013) "No smoking here: values, norms and culture in multi-agent systems", In: Artificial Intelligence and Law, v. 21, n. 1, 79 – 107.

Figueiredo, K. (2011) "Modeling and Validating Norms in Multi-Agent Systems", Master's Thesis, Universidade Federal Fluminense, Instituto de Computação.

Figueiredo, K. S., da Silva, V. T. (2013) "Identifying Conflicts between Norms and Values", In: International Workshop on Coordination, Organizations, Institutions and Norms in Agent Systems (COIN@AAMAS 2013), Saint Paul, MN, USA.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge acquisition, 5(2), 199-220.

Heinrich, S., Eberling, M., and Wermter, S. (2011). Determining Cooperation in Multiagent Systems with Cultural Traits. In: ICAART (2), 173 – 180.

Keesing, R. M. (1974) "Theories of culture", In: Annual Review of Anthropology 3, p.73 – 97.

McBreen, J., Di Tosto, G., Dignum, F., and Hofstede, G. J. (2011). Linking norms and culture. In: Culture and Computing (Culture Computing), 2011 Second International Conference, IEEE, 9 – 14.

Rokeach, M. (1979) "The Nature of Human Values", The Free Press. NY: Free Press.

Schwartz, S. and Bilsky, W. (1987) "Toward a universal psychological structure of human values", In: Journal of Personality and Social Psychology 53(3).

Uma Proposta de Resolução de Conflitos entre Normas e Valores

Jean Carlos Oliveira Santos[1], Karen da Silva Figueiredo[2]

Instituto de Computação, Universidade Federal do Mato Grosso, Cuiabá- MT, Brasil

jcoliveira93@gmail.com[1], karen@ic.ufmt.br[2]

Abstract. *Agents are autonomous entities and they have different interests. Therefore, in an open multi-agent system, conflicts can arise between the norms of the system and personal values of the agents. In this paper, we propose an algorithm to help the agent to resolve those conflict cases based on the value's influence of the norms.*

Resumo. *Agentes são entidades autônomas e têm interesses diferentes. Portanto, em um sistema multiagente aberto, podem surgir conflitos entre as normas do sistema e os valores pessoais dos agentes. Neste trabalho, propomos um algoritmo para ajudar o agente a resolver os casos de conflito entre normas e valores com base na influência dos valores nas normas.*

1. Introdução

Cada vez mais, pesquisas de sistemas multiagentes (SMAs) estão se concentrando nos aspectos culturais das organizações de agentes, tais como normas e valores, e.g. [Dechesne et al. 2013, van der Weide et al. 2010]. A cultura pode ser vista como uma coleção social dos padrões de comportamento aprendidos, compartilhados e transmitidos com o objetivo de adaptar os indivíduos de um grupo [Keesing 1974].

As normas são padrões de comportamento compartilhados por um grupo e usados para regular ações que devem ser executadas pelos agentes (obrigações) e as ações que não devem ser executadas pelos agentes (proibições), a fim de lidar com a heterogeneidade dos agentes.

Os valores são outros padrões de comportamento cultural, ajudando os agentes a avaliar e decidir o que fazer com base no que eles acreditam que é certo para si e para seu grupo. Segundo Schwartz e Bilsky (1987), os valores são princípios ordenados por uma importância relativa que guiam a seleção ou avaliação de comportamentos ou eventos. Quando um agente executa ações, estas ações irão promover ou rebaixar seus valores, de acordo com a importância dos mesmos.

Como as normas são utilizadas para regular as ações de um agente e os valores do agente são promovidos ou rebaixados pelas ações, conflitos podem surgir entre as normas do sistema e os valores do agente. Por exemplo, uma norma pode obrigar um agente a executar uma ação que rebaixa um ou mais de seus valores [Figueiredo e da Silva 2013].

Neste trabalho, apontamos os casos de conflitos que podem surgir entre normas e valores e propomos um algoritmo para ajudar o agente a resolver esses casos. Este

trabalho é uma extensão do algoritmo preliminar apresentado por Figueiredo e da Silva (2013) que é capaz somente de identificar os casos de conflito, sem propor uma forma de resolução. Até onde sabemos, este é o primeiro trabalho em SMAs que implementa uma solução para este tipo de conflito.

2. Casos de Conflito entre Normas e Valores

As *normas* regulam a execução de *ações* ao descrever **obrigações** e **proibições** para os agentes. As *ações*, por sua vez, podem **promover** ou **rebaixar** os *valores* do agente. Desse modo, as ações são um elemento central na ligação entre normas e valores. Assim, é necessário analisar a ação de uma norma para avaliar se existe ou não conflito entre a norma e os valores do agente. Também é necessário considerar os casos em que uma ação é a especialização de outra (relacionamento de generalização) e quando uma ação é parte de outra ação (relacionamento de composição), pois o relacionamento entre as ações afetam a análise dos valores que estão sendo promovidos e rebaixados.

Considerando o exposto, os casos em que uma norma estará em conflito com os valores de um agente são[1]:

1. *A norma define a obrigação de executar uma ação que rebaixa um ou mais valores importantes para o agente.*

2. *A norma define uma proibição de executar uma ação que promove um ou mais valores importantes para o agente.*

3. *A norma define uma obrigação de executar uma superação e todas as suas subações rebaixam pelo menos um valor importante para o agente.*

4. *A norma define uma proibição de executar uma superação e pelo menos uma de suas subações promove um ou mais valores importante para o agente.*

5. *A norma define uma obrigação de executar uma ação composta e a ação composta ou pelo menos uma das ações da composição rebaixam um ou mais valores importantes para o agente.*

6. *A norma define uma proibição de executar uma ação composta e a ação composta ou pelo menos uma das ações da composição promovem um ou mais valores importantes para o agente.*

Em todos os casos acima, a norma define um comportamento que vai contra os interesses pessoais do agente, constituindo uma situação conflitante para o agente.

3. Resolvendo os Casos de Conflito

Nesta seção, propomos um algoritmo para a resolução dos casos de conflitos apontados na seção anterior. Utilizamos a Linguagem Z [Spivey 1988] para a apresentação do algoritmo por ser uma linguagem intuitiva, de fácil entendimento, e já utilizada para descrição de algoritmos arquiteturais para agentes [D'Inverno et al. 1998, dos Santos Neto et al. 2012, Figueiredo e da Silva 2013].

Quando um dos casos de conflito da Seção 2 é identificado, o agente precisará decidir o que fazer: seguir o sistema e cumprir a norma apesar de seus valores, ou violar a norma e se manter com seus valores. O algoritmo que propomos para esta solução se

1 Devido ao pouco espaço os casos de conflitos foram apenas enunciados. Mais informações estão disponíveis no artigo [Figueiredo e da Silva 2013].

baseia no conceito da influência dos valores (*value's influence*, ou *VI*) da norma, que por sua vez é calculado de acordo com a importância que o valor tem para o agente (segundo a Teoria Universal dos Valores [Schwartz e Bilsky 1987]). Para cada caso de conflito, a influência dos valores é calculada recursivamente da seguinte maneira:

VI para o caso de conflito 1. Se a norma é uma *obrigação*, o VI será igual a soma da importância de todos os valores promovidos pela ação menos a soma das importâncias dos valores rebaixados pela ação.

VI para o caso de conflito 2. Se a norma for uma *proibição*, o VI será igual a soma da importância dos valores rebaixados menos a soma da importância dos valores promovidos pela ação.

VI para os casos de conflito 3 e 4. Se a ação regulada pela norma for uma *superação*, o VI da norma será igual ao VI do maior VI encontrado entre cada *subação* da ação regulada pela norma. A subação com o maior VI indica o melhor caso de execução da ação, i.e. o caso em que mais valores são promovidos e/ou menos valores são rebaixados.

VI para os casos de conflito 5 e 6. Se a ação regulada pela norma for uma *ação composta*, o VI da norma será igual a soma do VI de todas as normas derivadas de cada ação que a compõem e do VI da ação regulada pela norma. Como a ação composta exige que todas as ações que a compõem sejam executadas, a soma do VI de todas as ações é necessária.

Para realizar o cálculo do VI da norma descrito acima, o algoritmo proposto utiliza três funções: VP, VD e VI. As funções VP e VD são utilizadas pela função VI, elas recebem uma ação e retornam o somatório da importância dos valores promovidos e rebaixados pela ação, respectivamente.

$VP(Action) \rightarrow \mathbb{N}$
$sum : \mathbb{N}$

---

01. $\forall\, a : Action \bullet$
02. $\quad sum = 0$
03. $\quad \{\forall\, v : a.promotes \bullet sum = sum + v.importance\}$
04. $\quad VP(a) = sum$

$VD(Action) \rightarrow \mathbb{N}$
$sum : \mathbb{N}$

---

01. $\forall\, a : Action \bullet$
02. $\quad sum = 0$
03. $\quad \{\forall\, v : a.demotes \bullet sum = sum + v.importance\}$
04. $\quad VD(a) = sum$

$VI(Norm \times Action) \rightarrow \mathbb{Z}$
$influence : \mathbb{Z}$
$greatest : Action$

---

01. $\forall\, n : Norm, a : Action \bullet$
02. $\quad$ **if** $(a.subactions \neq 0)$ **then**
03. $\quad\quad greatest = \emptyset$
04. $\quad\quad \{\forall\, sub : a.subactions \bullet$
05. $\quad\quad\quad$ **if** $(VI(n, sub) > VI(n, greatest))$ **then**
06. $\quad\quad\quad\quad greatest = sub$
07. $\quad\quad \}$
08. $\quad\quad influence = VI(n, greatest)$
09. $\quad$ **else if** $(a.compositeactions \neq 0)$ **then**
10. $\quad\quad$ **if** $(n.deonticConcept = OBLIGATION)$ **then**
11. $\quad\quad\quad influence = VP(a) - VD(a)$
12. $\quad\quad$ **else if** $(n.deonticConcept = PROHIBITION)$ **then**
13. $\quad\quad\quad influence = VD(a) - VP(a)$
14. $\quad\quad \{\forall\, sub : a.compositeactions \bullet$
15. $\quad\quad\quad influence = influence + VI(n, sub)$
16. $\quad\quad \}$
17. $\quad$ **else**
18. $\quad\quad$ **if** $(n.deonticConcept = OBLIGATION)$ **then**
19. $\quad\quad\quad influence = VP(a) - VD(a)$
20. $\quad\quad$ **else if** $(n.deonticConcept = PROHIBITION)$ **then**
21. $\quad\quad\quad influence = VD(a) - VP(a)$
22. $\quad VI(n, a) = influence$

A função VI mapeia uma norma e sua ação para um número inteiro que indica o VI conforme os casos previamente apresentados. Primeiramente, a função verifica se a o tipo da ação pela norma. Se a ação é uma *superação*, a função guarda o maior VI encontrado recursivamente cada subação (casos de conflito 3 e 4 – linhas 2 a 8). Se a ação é uma *ação composta*, a função guarda recursivamente o somatório do VIs de cada ação da composição mais o VI da própria ação composta (casos de conflito 5 e 6 – linhas 9 até 16). Finalmente, se a ação for simples, a função guarda o VI calculado a partir do tipo da norma (obrigação ou proibição) utilizando as funções VP e VD (casos de conflito 1 e 2 – linhas 17 a 21). Ao final das iterações, a variável *influence* é retornada pela função com o VI calculado armazenado (linha 22).

A função *ComplyWithNorm* é a função principal do algoritmo de resolução proposto. Ela é utilizada pelo agente para checar se existem conflitos entra uma norma e seus valores e então resolvê-los. A função *ComplyWithNorm* mapeia uma norma e sua ação para um booleano, que indica se o agente deve cumprir ou não a norma.

$$ComplyWithNorm(Norm \times Action) \rightarrow Boolean$$

```
01. ∀ n : Norm, a : Action •
02.     if (checkNormValueConflicts(n, a)) then
03.         ComplyWithNorm(n, a) = VI(n, a) ≥ 0
04.     else
05.         ComplyWithNorm(n, a) = TRUE
```

Primeiramente, a função verifica a existência de algum dos casos de conflito da Seção 2 através da função *checkNormValueConflict*[2] (linha 2). Caso haja conflito, a função chama a função VI: se o resultado for maior ou igual a zero, a função *ComplyWithNorm* retorna ***verdadeiro***; caso contrário, retorna ***falso*** (linhas 2 a 3). Ou seja, se o valor do VI da norma for maior ou igual a zero, considerando todos os tipos de ações, o ganho do valores promovidos compensa a perda dos valores rebaixados no caso de obrigações, ou o contrário no caso de proibições. Assim, o agente preferirá cumprir a norma. Mas, se o VI da norma for menor que zero, é melhor para o agente seguir seus valores, mesmo que isso signifique violar a norma.

Caso não haja conflito, *ComplyWithNorm* também retorna verdadeiro (linhas 4 e 5), indicando que o agente não tem razão para violar a norma, de acordo com seus valores.

4. Exemplificando a Proposta

Nesta seção, apresentamos um exemplo para demonstração do algoritmo proposto. Para este exemplo, considere um grupo de agentes robôs de resgate (compostos por módulos como lanterna, sensores de calor, comunicação, etc.) que realizam buscas de pessoas feridas em escombros e destroços, por exemplo, em desabamentos. Considere que estes robôs possuem a seguinte norma regulando o seu comportamento:

Norma 1: Todos os robôs são proibidos de desligar um módulo durante uma missão.

2 A função *checkNormValueConflicts* que indica se há ou não um conflito entre as normas e os valores é apresentada em nossos trabalho anterior [Figueiredo e da Silva 2013] e omitida neste trabalho por falta de espaço.

Considere ainda que a ação da norma acima (desligar módulo - *turnOffModule*) possui a seguinte estrutura:

turnOffModule ({subactions: turnOffLights, turnOffSensors, turnOffComunication}, {promotes: power})
turnOffLights ({demotes: security})
turnOffSensors ({demotes: efficiency})
turnOffComunication ({demotes: security, cooperation})

A ação *turnOffModule* pode ser benéfica em algumas situações, como no caso de um módulo desnecessário para a situação estar gastando energia que poderia ser usada para estender o tempo de funcionamento do robô em uma missão, entretanto ela também pode comprometer a segurança do próprio robô, por exemplo. Por se tratar da proibição de uma *superação* que promove um valor do agente, esta norma está em conflito com seus valores (caso de conflito 4).

Dado o presente cenário, imagine que existem dois robôs de resgate (Robô 01 e Robô 02) com os conjuntos de valores pessoais distintos descritos abaixo que precisam avaliar se irão cumprir ou não a Norma 1 através do nosso algoritmo.

Conjunto de Valores do Robô 01:
power (importance: 2)
security (importance: 3)
efficiency (importance: 4)
cooperation (importance: 2)

Conjunto de Valores do Robô 02:
power (importance: 3)
security (importance: 2)
efficiency (importance: 2)
cooperation (importance: 0)

Quando a função *ComplyWithNorm* é chamada, ela identifica que existe um caso de conflito entre a Norma 1 e os valores do Robôs 01 e 02 (função *checkNormValueConflicts*) e então calcula o VI da Norma 1 através da função *VI*. Por se tratar de uma *superação*, a função *VI* irá calcular recursivamente o VI para cada uma das *subações* e irá retornar o maior VI para o caso.

VI (Norm1, turnOffModule) = greatest(VI(turnOffLights), VI(turnOffSensors), VI(Norm1,turnOffComunication))
VI (Norm1, turnOffLights) = importance(security) – importance(power)
VI (Norm1, turnOffSensors) = importance(efficiency) – importance(power)
VI (Norm1, turnOffComunication) = (importance(security) + importance (cooperation)) – importance(power)

Se o VI retornado pela função for maior ou igual a zero os robôs irão seguir a norma. Caso contrário, os robôs irão violar a norma. Como cada robô possui um conjunto de valores diferentes, vamos analisar a saída do algoritmo para cada caso.

Cálculo do VI para o Robô 01:
VI (Norm1, turnOffLights) = 3 – 2 = 1
VI (Norm1, turnOffSensors) = 4 – 2 = 2
VI (Norm1, turnOffComunication) = (3 + 2) – 2 = 3
VI (Norm1, turnOffModule) = 3

Cálculo do VI para o Robô 01:
VI (Norm1, turnOffLights) = 2 – 3 = -1
VI (Norm1, turnOffSensors) = 2 – 3 = -1
VI (Norm1, turnOffComunication) = (2 + 0) – 3 = -1
VI (Norm1, turnOffModule) = -1

Deve o Robô 01 cumprir com a norma? Neste caso, o VI da proibição de *turnOffModule* será maior que zero, portanto a função *ComplyWithNorm* entende que os valores rebaixados ao executar a ação que está proibida pela norma não compensam os

valores que seriam promovidos. Assim, o Robô 01 irá cumprir com a norma e manter seus módulos sempre ligados.

Deve o Robô 02 cumprir com a norma? Neste caso, o VI da proibição será um resultado negativo (-1), portanto a função *ComplyWithNorm* entende que os valores promovidos ao executar a ação da norma compensam a violação da proibição. Assim, o Robô 02 escolherá desligar um módulo, diminuindo suas funções e sua segurança, mas funcionando por mais tempo e atendendo a missão por um período maior.

5. Conclusão e Trabalhos Futuros

A resolução de conflitos entre normas e valores em tempo de execução é importante pelo fato que novos conflitos podem surgir a qualquer momento e o agente precisa saber lidar com essas situações. Primeiramente, as normas são constantemente modificadas devido a dinâmica do SMAs, e segundo, os valores dos agentes são alterados devido a convivência e influências culturais [Schwartz e Bilsky 1987].

Este trabalho propôs um algoritmo para a ajudar o agente na resolução dos conflitos entre normas e valores baseado na importância dos valores pessoais e sua influência nas normas. Nesta resolução consideramos não só ações simples, mas também as ações mais complexas com especializações e composições. Nosso próximo passo é estender o algoritmo para que este inclua no processo decisório a análise das sanções (recompensas e punições) aplicadas ao cumprir ou violar uma norma.

Referências

Dechesne, F., Di Tosto, G., Dignum, V. and Dignum, F. (2013) "No smoking here: values, norms and culture in multi-agent systems", In: Artificial Intelligence and Law, v. 21, n. 1, 79 – 107.

D'Inverno, M., Kinny, D., Luck, M. and Wooldridge, M. (1998) "A Formal Specification of dMARS", In: Proceedings of the 4th International Workshop on Intelligent Agents IV, London, UK, Springer-Verlag, 155 – 176.

dos Santos Neto, B., da Silva, V. and de Lucena, C. (2012) "An architectural model for autonomous normative agentes", In: Advances in Artificial Intelligence-SBIA 2012, Springer Berlin Heidelberg, 152 – 161.

Figueiredo, K. S., da Silva, V. T. (2013) "Identifying Conflicts between Norms and Values", In: International Workshop on Coordination, Organizations, Institutions and Norms in Agent Systems (COIN@AAMAS 2013), Saint Paul, MN, USA.

Keesing, R. M. (1974) "Theories of culture", In: Annual Review of Anthropology 3, p.73 – 97.

Schwartz, S. and Bilsky, W. (1987) "Toward a universal psychological structure of human values", In: Journal of Personality and Social Psychology 53(3).

Spivey, J. M. (1988) "Understanding Z: a specification language and its formal semantics", In: Cambridge University Press, New York, NY, USA.

van der Weide, T., Dignum, F., Meyer, J., Prakken, H. and Vreeswijk, G. (2010) "Practical reasoning using values", In: Argumentation in Multi-Agent Systems, Springer Berlin Heidelberg, 79 – 93.

Charruas: Caçadores e Coletores
Um cenário para avaliação de agentes normativos

Tiago L. Schmitz[1], **Jomi F. Hübner**[1]

[1]Universidade Federal de Santa Catarina
C.P. 476, 88040-900 Florianópolis - SC, Brasil

tiagolschmitz@gmail.com, jomi@das.ufsc.br

Abstract. *In the last decades several agents model were created to reason about norms. However we have not found a scenario able to experiment and evaluate the commitment and the efficiency of agent behavior. In this sense, this paper offers a scenario where the agent needs to deliberate about norms and desires, considering available resources.*

Resumo. *Nas últimas décadas surgiram várias linguagens e modelos de agentes capazes de raciocinar sobre normas. Contudo não encontramos um cenário que permita experimentar e avaliar o comprometimento do agente e a eficiência do seu comportamento. Neste sentido propomos um cenário no qual o agente deve deliberar sobre normas e desejos, considerando recursos disponíveis.*

1. Introdução

Sistemas multi-agentes é uma abordagem para uma sociedade virtual, na qual é possível utilizar uma organização para coordenar os agentes. Um importante componente destas sociedades é o sistema normativo [Boella and van der Torre 2004]. Um sistema normativo é um conjunto de normas que guiam o comportamento dos agentes, com o intuito de cumprir os objetivos da sociedade. A capacidade de entender e interpretar o sistema normativo é uma habilidade desejável para os agentes que compõem estas sociedades. Para que um agente tenha esta habilidade de entender e interpretar foram criadas nas últimas duas décadas diversos modelos [Carabelea et al. 2005, Gaertner and Toni 2008, Meneguzzi and Luck 2009, Pacheco 2012, Alechina et al. 2012]. Na apresentação da maioria desses modelos observa-se a ausência de um cenário capaz de comparar especificamente o raciocínio normativo.

Considerando que um agente normativo é capaz de comprometer-se ou abandonar os desejos e normas com base nos mais variados critérios (necessidade, intenção, sanção, recompensa e recursos), surgem as seguintes perguntas: Como avaliar o funcionamento desses agentes? Como comparar qual é o melhor modelo dados os diferentes parâmetros? Para auxiliar na resposta destas questões este artigo propõe, na seção 2, um cenário que contemple as situações de violação de normas, não atendimento de desejos, considerando os critérios descritos anteriormente.

2. Cenário

O cenário apresentado neste artigo é uma pequena tribo de índios buscando suprimentos para o inverno. Nesta tribo existem três demandas: pedras para as construções, carne para a alimentação e lenha para aquecer. Para obter estes itens os índios devem percorrer uma região e caçar animais e coletar pedras e lenha. Ao final de uma estação a tribo tem que ter caçado e coletado o máximo para sobreviver ao longo inverno.

2.1. Sistema normativo

Para atingir o objetivo o cacique definiu três papéis: o pedreiro, o caçador e o lenhador. Ao sabor das vontades do cacique foram atribuídas normas a cada um dos papéis e são apresentadas na tabela 1. Destaca-se que estas normas são estáticas em nosso cenário.

Tabela 1. Normas definidas pelo cacique

Pedreiro	caçador	lenhador
Obrigações		
coletar pedras	caçar animais	coletar madeira
Proibições		
não caçar animais	não coletar Pedras	não coletar Pedras
não recolher madeira	não coletar madeira	não caçar animais

O cacique ainda se reserva o direito de atribuir valores entre 0 e 1 para as sanções e recompensas destas normas. Esses valores podem ser alterados dependendo da instância do cenário. Por exemplo, se o cacique for tolerante as sanções impostas pelo descumprimento das normas serão baixas, se aproximando de 0.

2.2. Ambiente

Essa tribo foi muito feliz na escolha do local onde se estabeleceu. O ambiente é um local com aclives e declives suaves dignos das mais belas coxilhas dos pampas. Logo, o terreno não influência na movimentação dos índios por suas terras, permitindo que estes andem em qualquer direção sem nenhum obstáculo. Na figura 1 este terreno é representado por uma matriz e cada um dos elementos que compõem o cenário obedece a legenda da figura.

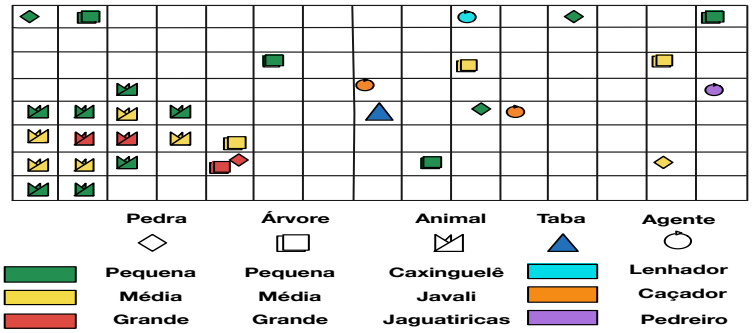


Figura 1. Cenário com todos os elementos

Neste ambiente pode-se encontrar árvores e pedras de tamanho pequeno, médio e grande, necessitando de diferentes tipos e quantidades de recursos para extrair a lenha e pedras. Além disso, existem três tipos de animais no ambiente jaguatiricas, javalis e caxinguelês. Estes animais possuem diferentes graus de dificuldade necessitando de quantidades de recursos diferentes para abatê-los. Neste cenário é importante ressaltar que plantas, pedras e animais, não serão capazes de se mover. Os objetos que compõem o ambiente não podem causar dano ao agente, a não ser que o agente interaja com o objeto e por imperícia se machuque.

Nesse ambiente existe mais um objeto o qual representa a taba. A taba tem três funções: receber os frutos das caçadas e coletas, obter insumos para caçadas e coletas e uma trocar mensagens sobre localização de outros objetos no mapa.

2.3. Agentes [1]

O cacique definiu que nenhum índio pode andar pelo território desacompanhado. Como complemento ele definiu grupos composto por 4 índios que representam uma unidade de ação que pode assumir o papel de pedreiro, lenhador ou caçador.

Cada agente no nosso cenário é representados por um grupo de ação. Esse grupo dispõem de recursos limitados de machados, picaretas, flechas, energia vital e número de índios. Machados, picaretas e flechas são recursos produzidos com pedra lascada e após ser utilizado ele é inutilizado. Um grupo de ação pode dispor de vários machados, picaretas e flechas. O número de índios é um recurso renovável, pois após o grupo de ação concluir um objetivo os índios voltam a ficar disponíveis. A energia vital é representada por um valor entre 0 e 1 e representa saúde do grupo de ação. Caso a energia vital do grupo estiver abaixo de $0,2$ o grupo só consegue se locomover. Quando o grupo de ação retorna para a taba os valores iniciais dos recursos são restaurados.

O grupo de ação deve utilizar estes recursos para atingir 3 objetivos possíveis (coletar lenha e pedra e caçar animais) através de quatro ações, coletar lenha e pedra, caçar animais e andar pelo território. Os grupos de ação tem diferentes aptidões para cumprir os objetivos, com exceção da ação de andar pelo território, a qual todos são aptos. O fato do grupo ter diferentes aptidões implica que ao realizar um objetivo para o qual ele não é totalmente apto causará uma perda de energia vital.

Cada grupo de ação tem custos específicos de recursos para cada ação. Estes custos são determinados no estado inicial do sistema. Um exemplo destes custos são apresentados nas tabelas 2, 3, 4.

Para realizar um objetivo além de observar as restrições do sistema normativo o agente tem que considerar os seus desejos. Esses desejos são compostos por: um objetivo, uma necessidade e uma intensidade. A necessidade e a intensidade são valores entre 0 e 1 que representam respectivamente, o quanto é essencial e o quanto é desejado atingir o objetivo. Por exemplo, a medida que o tempo passa o grupo de ação sente mais necessidade de caçar um animal para se alimentar. Em contrapartida um grupo de ação com aptidão para pedreiro não tem muita intenção de matar um animal. As regras que definem o aumento da necessidade e da intenção são definidos a priori na especificação do cenário.

2.4. Comunicação e percepção

O intuito deste cenário não é focar em comunicação entre agentes, e sim focar no raciocínio do agente sobre os desejos e normas. Dessa maneira a única comunicação entre os agentes é um quadro negro, no qual os grupos de ação informam onde fora avistado árvores, pedras ou animais. O acesso as informações do quadro negro só é feito quando os grupos estão na taba. A percepção dos grupos de ação é em forma de cruz: eles podem observar e agir nos quadrantes imediatamente superior, inferior, à direita e à esquerda.

[1] Este presente trabalho não tem a intenção de descrever o comportamento do agente (elemento a ser avaliado pelo cenário). Apenas são relacionadas as ações que podem ser executadas pelo agente bem como sua função no ambiente.

2.5. Problema decisório

O objetivo deste cenário é colocar o agente em situações nas quais ele tem que escolher entre obedecer as normas ou seguir seus desejos. Por exemplo, um agente pedreiro está a diversos dias coletando pedras sem voltar para a taba. Este período faz com que tanto a intensidade quanto a necessidade de caçar um animal sejam altas, pois o agente está com fome. O agente sabe que a poucos metros (quadrante ao lado) existem alguns caxinguelês. Considerando essa situação, o agente deve deliberar quais normas (coletar pedra e não caçar animais) e desejos (caçar animais) serão atendidos, levando em consideração os recursos disponíveis. Ressaltando que ao caçar caxinguelês o agente estará infringindo uma norma, contudo ele estará atendendo a um desejo.

2.6. Modelo do cenário

O modelo do cenário proposto pode ser representado pelo diagrama de classes da figura 2.6. Este modelo será utilizado futuramente na implementação da plataforma de avaliação de agentes normativos Charruas.

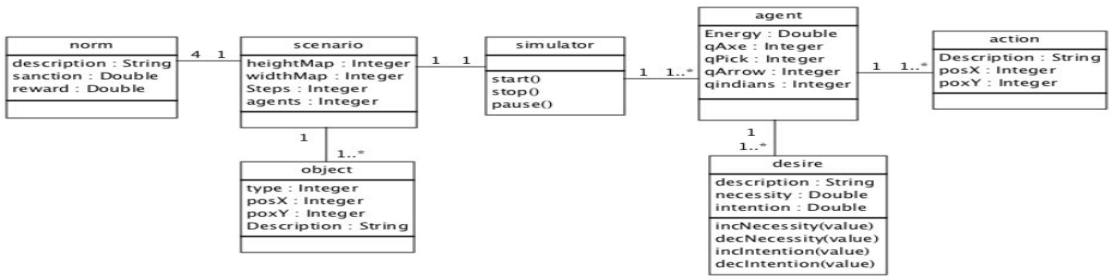


Figura 2. Modelo do cenário

3. Trabalhos relacionados

Existem cenários com finalidades diversas para avaliação de sistemas multiagentes, contudo não encontramos um cenário que aborde especificamente o processo de decisão dos agentes normativos. O Art test bed [Fullam et al. 2005] é um cenário clássico que foca na confiança e reputação entre agentes. Outros cenários são abordados pelo Multi agent programming contest (MAPC)[2], entre eles podemos citar o Cowboys and cows [Behrens et al. 2010] e o Agent on mars [Behrens et al. 2012], tem um enfoque na organização e coordenação de agentes. Também focado na organização e coordenação o Robocup rescue [Kitano et al. 1999] é um cenário mais rico e complexo que os utilizados no MAPC.

Os agentes destes trabalhos citados **não são capazes de infringir normas** e tão pouco são capazes de **deliberar sobre as normas e desejos, observando os recursos disponíveis**. Nesse caminho o cenário Charruas vem suprir esta lacuna e fornecer uma ferramenta de avaliação para agentes normativos. Nesse ferramenta poderá ser observado o grau de comprometimento do agente com as normas e desejos adotados, a eficiência dos agentes em um determinado período de tempo e por fim a relação entre comprometimento e eficiência.

[2]https://multiagentcontest.org

4. Considerações finais

O Charruas é um cenário para avaliação de agentes normativos. O agente possui poucos objetivos e normas. Contudo, o cenário propõem situações onde o agente deve escolher entre obedecer as normas ou satisfazer seus desejos, podendo fazer uso de informações como sanções, recompensas, intenções e necessidades. Diferente dos trabalhos relacionados, o Charruas permite aos agentes infringir normas, ignorar desejos e considerar recursos disponíveis.

Este é um trabalho em andamento e o próximo passo é a implementação desta plataforma. Uma vez concluída esta plataforma, a mesma será utilizada para avaliar o modelo Huggin [Schmitz and Hübner 2013].

Referências

Alechina, N., Dastani, M., and Logan, B. (2012). Programming norm-aware agents. In Conitzer, V., Winikoff, M., Padgham, L., and van der Hoek, W., editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Valencia, Spain.

Behrens, T., Dastani, M., Dix, J., Huebner, J., Koester, M., Novak, P., and Schlesinger, F. (2012). The multi-agent programming contest. *AI Magazine*, 33(4):111–113.

Behrens, T. M., Dastani, M., Dix, J., Köster, M., and Novák, P. (2010). The multi-agent programming contest from 2005-2010 - from gold collecting to herding cows. *Ann. Math. Artif. Intell.*, 59(3-4):277–311.

Boella, G. and van der Torre, L. W. N. (2004). Regulative and constitutive norms in normative multiagent systems. In *KR*, pages 255–266.

Carabelea, C., Boissier, O., Castelfranchi, C., and Etienne, S.-g.-e. M. D. S. (2005). Using Social Power to Enable Agents to Reason About. pages 166–177.

Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K. S., Rosenschein, J. S., Vercouter, L., and Voss, M. (2005). A specification of the agent reputation and trust (art) testbed: Experimentation and competition for trust in agent societies. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 512–518, New York, NY, USA. ACM.

Gaertner, D. and Toni, F. (2008). Preferences and assumption-based argumentation for conflict-free normative agents. In *Proceedings of the 4th international conference on Argumentation in multi-agent systems*, ArgMAS'07, pages 94–113, Berlin, Heidelberg. Springer-Verlag.

Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., and Shimada, S. (1999). Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS*, pages 739–746. IEEE Computer Society.

Meneguzzi, F. and Luck, M. (2009). Norm-based behaviour modification in BDI agents. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 177–184. International Foundation for Autonomous Agents and Multiagent Systems.

Pacheco, N. C. (2012). *Using Nnorms to Control Open Multi-Agent Systems*. Tesis doctoral en informática, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.

Schmitz, T. L. and Hübner, J. F. (2013). Raciocínio normativo organizacional para agentes: Uma abordagem anímica. In *IV Workshop sobre Sistemas de Software Autônomos - AutoSoft 2013*, pages 59–68.

Tabela 2. Grupo de ação pedreiro

Recursos	Coletar						Caçar		
	Lenha			Pedra			Animais		
	Árvore pequena	Árvore média	Árvore grande	Pedra pequena	Pedra média	Pedra grande	Caxinguelês	Javalis	Jaguatiricas
Machados	2	3	4	0	0	0	0	0	0
Picaretas	0	0	0	1	2	3	0	0	0
Flechas	0	0	0	0	0	0	3	10	15
Energia vital	0.2	0.4	0.8	0	0	0	0.1	0.3	0.6
Índios	2	3	4	1	1	2	1	3	4

Tabela 3. Grupo de ação caçador

Recursos	Coletar						Caçar		
	Lenha			Pedra			Animais		
	Árvore pequena	Árvore média	Árvore grande	Pedra pequena	Pedra média	Pedra grande	Caxinguelês	Javalis	Jaguatiricas
Machados	2	3	4	0	0	0	0	0	0
Picaretas	0	0	0	1	2	3	0	0	0
Flechas	0	0	0	0	0	0	3	10	15
Energia vital	0.1	0.4	0.9	0.1	0.5	0.8	0	0	0
Índios	2	3	4	1	1	2	1	3	4

Tabela 4. Grupo de ação lenhador

Recursos	Coletar						Caçar		
	Lenha			Pedra			Animais		
	Árvore pequena	Árvore média	Árvore grande	Pedra pequena	Pedra média	Pedra grande	Caxinguelês	Javalis	Jaguatiricas
Machados	1	2	3	0	0	0	0	0	0
Picaretas	0	0	0	2	3	5	0	0	0
Flechas	0	0	0	0	0	0	3	8	12
Energia vital	0	0	0	0.2	0.4	0.8	0.1	0.3	0.4
Índios	1	2	2	1	1	2	1	3	4

Utilizando DCOP para Modelar o Problema de Alocação de Redes Virtuais

Alexander R. Gularte, Odorico Machado Mendizabal, Raquel de Miranda Barbosa, Diana F. Adamatti

[1]Programa de Pós Graduação em Computação – Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG) – Rio Grande – RS – Brazil

{alexgularte,odoricomendizabal,raquelmbarbosa,dianaadamatti}@furg.br

Abstract. *Distributed Constraint Optimization Problem (DCOP) is a formalism that is widely used for coordination in multiagent systems and has distributed, robust and scalable algorithms. Virtual Networks (VN) offer a flexible and economic approach to deploy customer suited networks. However, defining how resources of a physical network are used to support VNs demands is a NP-hard problem. This work presents a modeling of the VN allocation problem using DCOP with factor graphs.*

Resumo. *Problema de Otimização de Restrição Distribuída (DCOP) é um formalismo amplamente utilizado para coordenação de Sistemas Multiagente e possui algoritmos distribuídos, robustos e escaláveis. Redes Virtuais (RV) oferecem uma forma flexível e econômica para implantar redes adequadas aos clientes. Entretanto, definir como os recursos de uma rede física são usados para suportar demandas das RVs é um problema NP-hard. Este trabalho apresenta uma modelagem do problema de alocação de RVs utilizando DCOP com grafos-fatores.*

1. Introdução

Virtualização de redes consiste em compartilhar recursos de uma rede física entre várias redes virtuais heterogêneas. Nesse processo, os recursos de hardware dos roteadores (ex. CPU, Memória) são compartilhados por nós virtuais. No caso dos links físicos, a banda é compartilhada entre diferentes links virtuais. Conhecido na literatura por *Virtual Network Embedding* (VNE) *Problem*, o mapeamento de redes virtuais em redes físicas é um dos principais desafios da virtualização de redes. Esse problema apresenta uma complexidade *NP-Hard* e a maioria das abordagens presentes na literatura é centralizada, o que gera problemas de escalabilidade, visto que o número de elementos mapeados costuma ser elevado [Fischer et al. 2013]. A alocação de uma mesma rede virtual pode ser realizada por diferentes fornecedores de infraestrutura (InterInP) ou por um mesmo fornecedor (IntraInP).

O formalismo DCOP (Distributed Constraint Optimization Problems) é utilizado para coordenação de Sistemas Multiagente (SMA) e vem ganhando destaque na literatura pela robustez e escalabilidade. Um DCOP é formalmente definido como a tupla (X , D , C , A , α), onde $X = \{x_1, x_2, \ldots, x_n\}$ é um conjunto de n variáveis, $D = \{D(x_1), D(x_2), ..., D(x_n)\}$ um conjunto de domínios discretos no qual cada elemento corresponde ao domínio de uma variável, C um conjunto de funções de utilidade, A o conjunto de agentes e α o

mapeamento de agentes e variáveis. Encontrar uma solução de utilidade máxima para um DCOP também é um problema *NP-Hard* [Modi et al. 2005].

Os dois problemas apresentados, DCOP e alocação de redes virtuais, possuem uma série de características em comum. Ambos são problemas de otimização e, quando abordagens distribuídas são adotadas, o tamanho e o número de mensagens devem ser restritos. Atualmente, a literatura apresenta apenas um algoritmo distribuído para alocação de redes virtuais IntraInP [Houidi et al. 2008]. O trabalho de Houidi et al. utilizou com êxito a abordagem multiagente para resolver problemas de alocação de redes virtuais.

O principal objetivo deste trabalho é modelar o problema de alocação de redes virtuais através de DCOPs. Essa modelagem possibilita estender, para o contexto da alocação de redes virtuais, os recentes avanços que a comunidade científica vem alcançando no desenvolvimento de algoritmos distribuídos para DCOP.

Inicialmente será apresentado na Seção 2, a proposta de Houidi et al. que consiste, até o momento, na única abordagem distribuída para alocação de redes virtuais IntraInP. A Seção 3 apresenta a representação de DCOP com grafos-fatores. Na Seção 4 é demonstrado como o problema de alocação de redes virtuais pode ser modelado por DCOP. Finalmente, a Seção 5 apresenta as conclusões do trabalho.

2. Mapeamento Distribuído IntraInP Baseado em SMA

Em [Houidi et al. 2008] foi proposto o primeiro algoritmo distribuído para mapeamento de redes virtuais e único IntraInP. Essa abordagem é baseada em SMA onde cada nó físico executa um agente que é responsável por implementar o algoritmo de mapeamento. A estratégia adotada se baseia em dividir a rede virtual em uma topologia *hub-and-spoke*. Um *hub* é um nó central que se conecta com múltiplos nós adjacentes denominados *spokes*. Os *spokes* também podem representar *hub* de outros clusters. Então, o mapeamento da rede virtual se resume em mapear sequencialmente os clusters *hub-and-spoke*.

O algoritmo multiagente garante a negociação e sincronia entre os nós físicos. O padrão de comunicação ACL (*Agent Communication Language*) é utilizado para troca de mensagens entre os nós físicos representados por agentes. O framework JADE é utilizado para implementar os agentes responsáveis por executar o algoritmo distribuído.

3. Representação de DCOPs com Grafos-Fatores

Uma das formas utilizadas para representar um DCOP são os grafos-fatores, que consistem em grafos bipartidos. Um grafo bipartido é composto por arestas não direcionais e dois conjuntos de nós. Nesses grafos, cada aresta conecta nós de conjuntos diferentes. No caso dos grafos-fatores, um conjunto de nós representa as variáveis das funções (nós de variáveis), enquanto os outros nós representam as funções (nós de fatores). As arestas conectam as variáveis às funções sempre que uma variável for argumento para uma função. Uma aresta existe entre um nó fator e um nó variável se, e somente se, a variável é um argumento para a função representada pelo nó fator. A Figura 2 apresenta um exemplo de grafo-fator com cinco variáveis ($x_{1,1}, x_{2,2}, x_{1,3}, x_{2,1}, x_{1,2}$) e quatro funções ($f_A$,$f_D$,$f_B$,$f_C$).

A representação baseada em grafos-fatores permite utilizar o algoritmo soma-máxima [Farinelli et al. 2008] e suas evoluções, que vêm recebendo grande atenção por parte da comunidade científica por apresentar boa performance e escalabilidade. Enquanto, a maioria dos algoritmos para DCOP troca um número exponencial de pequenas

mensagens ou poucas mensagens que crescem exponencialmente, no algoritmo soma-máxima, os processos trocam pequenas funções apenas com seus vizinhos diretos.

4. Modelagem com DCOP

O processo de alocação de redes virtuais pode ser dividido em duas etapas: mapeamento de nós e mapeamento de links. Cada etapa produz um DCOP que pode ser resolvido pelo algoritmo soma-máxima. A primeira etapa consiste em alocar os nós virtuais em nós físicos. Em seguida, alocam-se os links virtuais em caminhos físicos.

4.1. Alocação de Nós Virtuais

O processo de alocação de redes virtuais inicia com a descoberta dos nós físicos candidatos a hospedar os nós virtuais[1]. Cada nó virtual n_{v_i} é representado por uma variável x_i, cujo domínio $D(x_i)$ é um conjunto formado pelos nós físicos candidatos a hospedá-lo. Assim, o valor definido para x_i estabelece o(s) nó(s) físico(s) que pode(m) hospedar n_{v_i}. As funções do DCOP representam as restrições no uso dos recursos físicos.

Cada nó físico n_{f_j} tem uma função de utilidade associada f_j, que é inversamente proporcional à quantidade de recursos fornecida por n_{f_j} para atender os seus nós virtuais hospedados. Em outras palavras, quanto menor o stress gerado nos nós físicos, maior será a utilidade produzida. Então, ao aplicar o algoritmo soma-máxima, a solução produzida será aquela que tenta maximizar o somatório das utilidades dos nós físicos, ou seja, distribuir com eficiência os recursos físicos entre os elementos virtuais.

Uma vez definidas as variáveis e funções, pode-se criar o grafo-fator para executar o algoritmo soma-máxima. Os nós virtuais serão representados por nós de variável e os nós físicos por nós de função.

As Figuras 1 e 2 mostram um exemplo dessa modelagem. A Figura 1 mostra um exemplo de problema de alocação de redes virtuais, onde uma rede física com quatro nós recebe duas requisições de redes virtuais com três e dois nós, respectivamente. Conforme ilustra a figura, cada nó virtual possui um conjunto de nós físicos candidatos à hospedagem. Esses candidatos são selecionados com base em restrições de localização geográfica. Por exemplo, o nó 3 da rede virtual 1 pode ser alocado tanto ao nó físico B quanto ao D.

A Figura 2 mostra o grafo-fator resultante do problema apresentado na Figura 1. Cada nó virtual originou um nó de variável cujo domínio é formado pelos nós físicos capazes de alocar o nó virtual. Por exemplo, o nó virtual 3 da primeira rede virtual pode ser alocado ao nó físico B ou D. Então, foi criado um nó de variável $x_{1,3}$ com domínio $\{D, B\}$.

Cada nó físico possui um nó de função correspondente, onde uma função de utilidade descreve o stress produzido sobre o nó físico em relação às possíveis alocações. Por exemplo, o nó físico C recebeu requisição de alocação dos nós virtuais $VN_{2,1}$ e $VN_{1,2}$. Então, foi criado um nó de função f_C com uma função de utilidade que possui como parâmetro as variáveis $x_{2,1}$ e $x_{1,2}$. Dependendo do valor atribuido às variáveis $x_{2,1}$ e $x_{1,2}$ a utilidade resultante da função f_C poderá ser maior ou menor.

[1]Neste trabalho esta etapa é omitida. Assume-se a pré-existência de um método capaz de realizar a seleção de candidatos com base em restrições (ex. restrições geográficas).

Nó Virtual	Nós Físicos Candidatos
$VN_{1,1}$	{A}
$VN_{1,2}$	{C}
$VN_{1,3}$	{B,D}
$VN_{2,1}$	{B,C}
$VN_{2,2}$	{D,A}

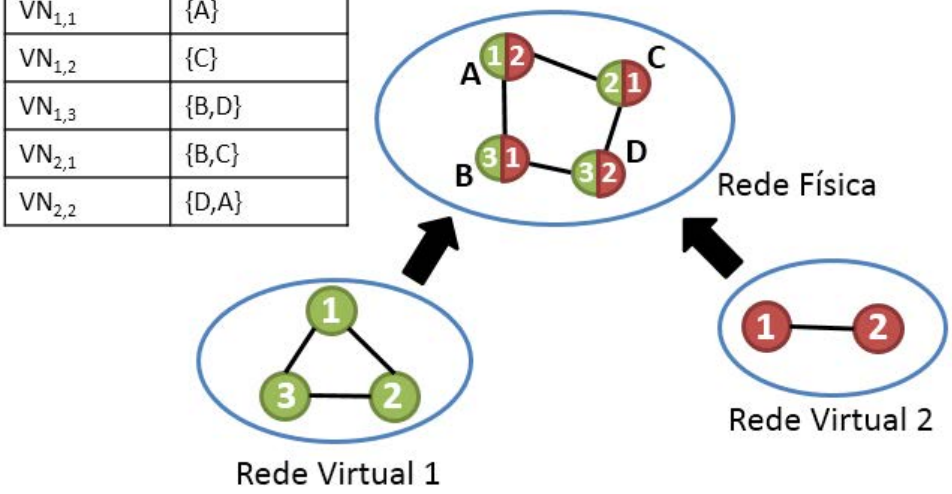


Figura 1. Exemplo de Problema de Alocação de Nós Virtuais

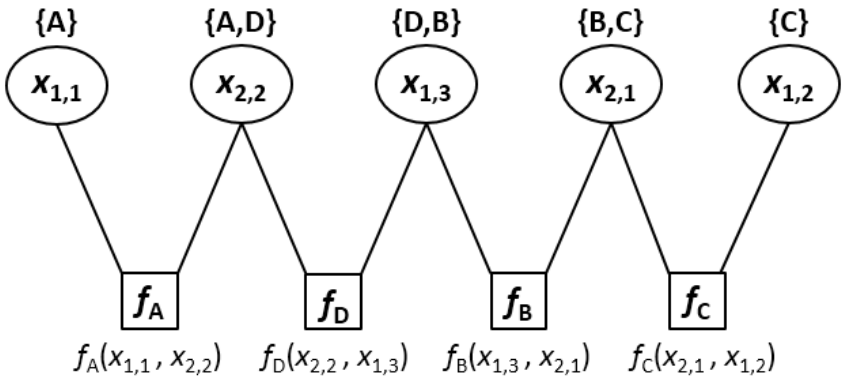


Figura 2. Grafo-fator

4.2. Alocação de Links Virtuais

Os links virtuais devem ser mapeados em caminhos da rede física. Cada caminho é formado por um conjunto acíclico de links físicos que conectam dois nós físicos. Esse problema pode ser modelado através de um DCOP seguindo a mesma lógica utilizada no mapeamento de nós, que é utilizar variáveis para representar elementos virtuais e funções para representar a utilidade dos recursos físicos. Assim, o algoritmo aplicado tentará otimizar o uso de recursos, permitindo que um maior número de elementos virtuais sejam adicionados à mesma rede física.

A Figura 3 mostra um exemplo de problema de alocação de links, formado por uma rede virtual e uma rede física. Como mostra a figura, os nós virtuais 1, 2 e 3 foram alocados aos nós físicos A, C e D, respectivamente. Para atender o link virtual $l_{v_{1,2}}$ só existe um caminho possível na rede física, que é formado pelo link físico $l_{f_{AC}}$. O link virtual $l_{v_{1,3}}$ pode ser alocado em dois caminhos possíveis. O primeiro é formado por dois links físicos: $l_{f_{AC}}$ e $l_{f_{CD}}$. O segundo caminho possível é formado por três links físicos: $l_{f_{AC}}$, $l_{f_{CB}}$ e $l_{f_{BD}}$.

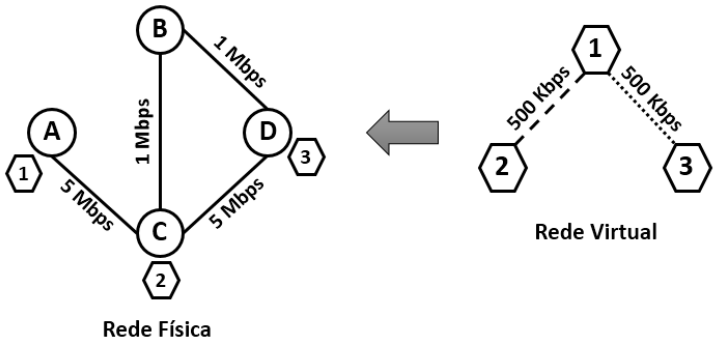


Figura 3. Cenário Exemplo

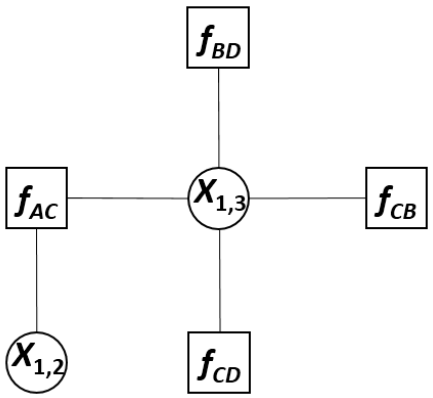


Figura 4. Grafo-fator

A primeira etapa para resolver o problema com DCOP consiste em definir as variáveis. Como a rede virtual é composta por dois links ($l_{v_{1,2}}$ e $l_{v_{1,3}}$), devem ser criadas duas variáveis, uma para cada link: $x_{1,2}$ e $x_{1,3}$. Os domínios das variáveis são definidos a partir do conjunto de possíveis caminhos: $x_{1,2} = \{AC\}$ e $x_{1,3} = \{ACD, ACBD\}$. Em seguida, define-se uma função de utilidade para cada link físico: $f_{AC}(x_{1,2}, x_{1,3})$, $f_{CD}(x_{1,3})$, $f_{CB}(x_{1,3})$, $f_{BD}(x_{1,3})$. A função f_{AC} possui como parâmetro as variáveis que representam links virtuais alocáveis a pelo menos um caminho formado pelo link físico $l_{f_{AC}}$. As demais funções são definidas seguindo essa mesma lógica. A Figura 4 mostra o grafo-fator do DCOP gerado.

5. Conclusão

Um SMA é composto por um conjunto de agentes que trabalham de forma coordenada. Nesses sistemas, um problema pode ser resolvido de maneira distribuída com cada agente sendo capaz de resolver um subproblema. Para que isso ocorra os agentes precisam coordenar a realização de suas tarefas em conjunto. Uma das formas de realizar a coordenação dos agentes é através de abordagens baseadas em DCOP.

Neste trabalho foi apresentada uma abordagem para resolver o problema de alocação de redes virtuais baseada em DCOP, onde o mapeamento de nós e links é realizado em duas etapas. Inicialmente, os nós virtuais, físicos e as relações de candidatos são utilizados para criar uma representação em grafo-fator para o problema. Essa

representação permite resolver o problema através do algoritmo soma-máxima. Em seguida, a solução é utilizada como entrada para a fase de mapeamento de links, que também pode ser resolvido através da modelagem por grafos-fatores.

Diversos trabalhos vem sendo desenvolvidos para solucionar o problema de alocação de redes virtuais em duas etapas separadas [Botero et al. 2012] , [Nogueira et al. 2011], [Chowdhury et al. 2012]. Contudo, a maioria deles baseia-se em métodos centralizados que acabam sofrendo problemas de escalabilidade . O trabalho de Houidi et al. [Houidi et al. 2008] consiste na única abordagem capaz de realizar alocação distribuída de redes virtuais em uma mesma rede física. Apesar de distribuída, essa abordagem ainda apresenta pouca escalabilidade uma vez que os nós precisam se comunicar com todos os demais para manter um conhecimento global do sistema atualizado.

Em [Gularte 2014] a modelagem apresentada neste trabalho foi formalmente verificada utilizando a técnica de verificação de modelos. Os resultados obtidos permitiram concluir que o método atende um conjunto de propriedades capazes de descrever o adequado comportamento de algoritmo de alocação de nós virtuais.

Referências

Botero, J., Hesselbach, X., Fischer, A., and Meer, H. (2012). Optimal mapping of virtual networks with hidden hops. *Telecommunication Systems*, 51(4):273–282.

Chowdhury, M., Rahman, M. R., and Boutaba, R. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. Netw.*, 20(1):206–219.

Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. R. (2008). Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, AAMAS '08, pages 639–646, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Fischer, A., Botero, J., Beck, M., De Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, PP(99):1–19.

Gularte, A. R. (2014). Alocação de redes virtuais baseada em otimização de restrição distribuída. Master's thesis, Universidade Federal do Rio Grande, Rio Grande, Rio Grande do Sul, Brasil.

Houidi, I., Louati, W., and Zeghlache, D. (2008). A distributed virtual network mapping algorithm. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 5634–5640.

Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180.

Nogueira, J., Melo, M., Carapinha, J., and Sargento, S. (2011). Virtual network mapping into heterogeneous substrate networks. In *Proceedings of the 2011 IEEE Symposium on Computers and Communications*, ISCC '11, pages 438–444, Washington, DC, USA. IEEE Computer Society.

Generating arguments based on Data-oriented Belief Revision model

Mariela Morveli-Espinoza[1], **Cesar A. Tacla**[1]

[1]CPGEI – Universidade Tecnológica Federal do Parana (UTFPR)
Curitiba – PR – Brazil

morveli.espinoza@gmail.com, tacla@utfpr.edu.br

Abstract. *Data-Oriented Belief Revision (DBR) is a relatively recent approach that claims the difference between pieces of information gathered and stored by agent (data) and information revised and considered reliable (beliefs). Data structure proposed in DBR can also be used to implement the structure of an argument. It's worth noting that this approach can be considered as the first step to integrate both belief revision and argumentation areas and allowing them to be modeled in the same framework. Our objective is to continue making progress on this research from a computational view. This article is our first step, here we propose two ways to generate arguments and counter-arguments in a monological argumentation based on the DBR model's data structure.*

1. Introduction

Argumentation is a reasoning model based on the construction of arguments in favor of or against some statement and then to select the most acceptable of them [Amgoud 2005]. On the other hand, belief revision is focused on investigating knowledge bases in change. Proposed approaches give different interpretations to the term change. Gardenfors [Gardenfors 1988] identified three fundamental types of belief change: revision, expansion and update. Katsuno and Meldelzon [Katsuno and Mendelzon 1991] recommend updating to handle knowledge in a changing world. In general, belief revision means the process of adapting some set of beliefs to new data.

In an agent environment, an agent collates knowledge to construct arguments for and against a particular claim and after arguments construction the agent draws a conclusion. Most works on argument construction, also known as argument generation, propose techniques and algorithms that focus on how to deal with contradicting arguments or how to generate adequate arguments[Ontanon and Plaza 2007][Besnard et al. 2010]. However, the matter of the dynamics of the knowledge base (KB) is neglected, i.e. it is not taken into account that during argument generation process, an agent can also receive new data that may change its beliefs and can alter the final conclusion.

[Paglieri 2004] has proposed DBR, a belief revision model that not just best characterizes the epistemic dynamics but it also defines a data structure where the squema of argumentation is liable of immediate implementation [Paglieri and Castelfranchi 2006]; in other words, it allows to integrate argumentation and belief revision in a single framework. DBR model also proposes to separate the knowledge of the agent in data and beliefs. Data are pieces of information gathered and stored by the agent and beliefs are pieces of information revised and considered reliable by the agent. This proposal is supported by the fact that not all incoming data are to be

believed, but it doesn't mean that not believed factual data are forgotten. Another reason to work with DBR model is that both data and belief states are finite and deductively open, and it is quite difficult that contradicting data become beliefs, since selection process is based on the measure of data credibility, and the credibility of contradicting data is conversely proportional.

In this paper, we propose, based on the DBR model, two ways for generating arguments from a set of data an agent has in advance or from new data. In our proposal, we take into account both the necessity of finding supports for a claim and the autonomous generation of arguments from new data.

This article is organized as follows: Section 2 shows DBR principal definitions used in our proposal. A motivational example and our proposal are presented in Section 3. Finally, some conclusions and future works are detailed in Section 4.

2. Data-oriented belief revision model and argumentation

Data-oriented Belief Revision (DBR) is a belief revision model alternative to the AGM approach [Gardenfors 1988]. As mentioned before, two basic informational categories are put forward (data and beliefs). Data are selected (or rejected) as beliefs on the basis of their properties, i.e. the cognitive reasons to believe them. These properties are:

1. **Relevance:** a measure of the pragmatic utility of the datum, i.e. the number and values of the (pursued) goals that depends on that datum;
2. **Credibility:** a measure of the number and values of all supporting data, contrasted with all conflicting data, both external and internal sources;
3. **Importance:** a measure of the epistemic connectivity of the datum, i.e. the number and values of the data that the agent will have to revise, should he revise that single one;
4. **Likeability:** a measure of the motivational appeal of the datum, i.e. the number and values of the (pursued) goals that are directly fulfilled by that datum.

In DBR, the data structure is conceived as network of nodes (data or beliefs), linked together by characteristic relations. This proposed form of arrange data lets to represent a net of arguments easily. The following are the three different types of data relations:

1. **Support:** ϕ supports ψ (in symbols: $\phi \Rightarrow \psi$) iff $c^{\psi} \propto c^{\phi}$, the credibility of ψ is directly proportional to the credibility of ϕ.
2. **Contrast:** ϕ contrasts ψ (in symbols: $\phi \perp \psi$) iff $c^{\psi} \propto 1/c^{\phi}$, the credibility of ψ is conversely proportional to the credibility of ϕ.
3. **Union:** ϕ and ψ are united (in symbols: ϕ & ψ) iff c^{ψ} and c^{ϕ} jointly (not separately) determine the credibility of another datum γ.

Where ϕ, ψ and γ are data or beliefs that are in the network.

3. Proposal

In this proposal, only relevance and credibility are used to assess data and only support and contrast relations are taken into account. Importance, likeability and union relation will be studied in future works.

Following definitions are necessary before continuing:

Definition 1. (*Argument*) An argument is a tuple $Ar = \langle C, S_C, K_C \rangle$ where:

- C is the claim
- S_C is the set of supports of claim C
- K_C is the set of contrasts (attacks) of claim C

Definition 2. (*Agent*) An agent is a tuple $Ag = \langle B_{Ag}, G_{Ag}, D_{Ag}, R_{Ag}, A_{Ag} \rangle$ where:

- B_{Ag} is the set of its beliefs
- G_{Ag} is the set of its goals
- D_{Ag} is the set of its data
- R_{Ag} is the set of relations among data or beliefs: support, union and contrast
- A_{Ag} is the set of accepted arguments.

3.1. Example

This example consists of two parts, in the first one the agent has existing data from which it constructs some arguments and reaches a preliminary conclusion. In the second part, the agent evaluates data that has arrived while it was in its construction process and it also looks for more data from other sources. Finally the agent reaches a final conclusion. See below how generation arguments process is carried out and which will be the conclusion of the agent.

C is an agent interested in investing in short-term trading stocks of company ACME ($f1$). Let's suppose that it already has related data that support its desire and others that don't. According to supporting data, it is recommended to invest in short-term trading stocks of company ACME because low capital is needed ($f3$), the reinvestment is quick ($f5$), leverage has no risks ($f6$) and C would earn more money than in long-term investing ($f7$). On the other hand, according to contrasting data, it is not recommended to invest in short-term trading stocks of company ACME because the stock market is too much volatile ($f8$), there is not enough time to take decisions ($f4$) and C would spend too much in bank rates ($f2$). Both support and contrast relations are denoted as follow:

- $f3 \Rightarrow f1$
- $f5 \Rightarrow f1$
- $f6 \Rightarrow f1$
- $f7 \Rightarrow f1$
- $f8 \bot f1$
- $f4 \bot f1$
- $f2 \bot f1$

Therefore $S_{f1} = \{f3, f5, f6, f7\}$ and $K_{f1} = \{f4, f8, f2\}$

3.2. Generation of arguments

3.2.1. Generating arguments out of necessity

It is triggered when a new goal is added to the list of goals of the agent. It can be generated from self claims or from contrasts the agent wants to defeat: $G_{Ag} = G_{Ag} \cup \{\alpha\}$, where α is the new goal. Agent looks for information that support its goals, it can obtain this by communication, perception or exists the possibility that agent already has some information in its data net that has not been evaluated.

Steps followed by agent to assess a datum or a set of data are:

1. The first step is to evaluate the relevancy of new entry data or data from the data network.
2. The agent must evaluate the credibility of those data considered relevant. The formula[1] used for this is:

   $$c^{\alpha} = [1 - \prod_{\mu \in S_{\alpha}} (1 - c^{\mu})] \times \prod_{\chi \in K_{\alpha}} (1 - c^{\chi})$$

   Besides supporting and contrasting data, the reliability of the source of some data is taken into account. This is considered as part of the set of supports when applying the formula. There are cases where a datum has neither supports nor contrasts, it only has information about its source. To calculate the credibility of a datum based upon its source, we use: $c^{\psi} = rel(src_{\psi})$ where, c^{ψ} is the credibility of datum ψ and $rel(src_{\psi})$ is the reliability of the source of datum ψ.
3. In order to know which data will become beliefs, the agent must evaluate data with the following condition:

   If $C : c^{\phi} > k$ then $B^{s} \in B$, where: $k = 0.5$ is the threshold.

   It is important to mention that this threshold value is just an example, since the value of threshold can vary from 0 to 1. This example setting expresses a thoroughly realistic attitude.

Following with the example, our agent C wonders whether it should or not invest in short-term trading, therefore $f1$ becomes its first goal: $G_C = \{f1\}$. Now, agent is going to try to find supports to its goal (this doesn't mean it can't find contrasts too). Since it already has data, it will begin to assess these ones.

It is important to clarify that, for the moment, reliability was arbitrarily set. The result of the evaluation of data (supports and contrasts) credibility is:

- $f2$: src_{f2}= university friend, $rel(src_{f2}) = 0.2$, therefore $c^{f2} = 0.2$.
- $f3$: src_{f3}= site specialized in investments, $rel(src_{f3}) = 0.9$, therefore $c^{f3} = 0.9$.
- $f4$: src_{f4}= investment blog, $rel(src_{f4}) = 0.3$, therefore $c^{f4} = 0.3$.
- $f5$: src_{f5} =investment magazine, $rel(src_{f5}) = 0.6$, therefore $c^{f5} = 0.6$.
- $f6$: src_{f6} =top investment magazine, $rel(src_{f6}) = 0.8$, therefore $c^{f6} = 0.8$.
- $f7$: src_{f7} =university friend, $rel(src_{f7}) = 0.2$, therefore $c^{f7} = 0.2$.
- $f8$: src_{f8} =investment magazine, $rel(src_{f8}) = 0.6$, therefore $c^{f8} = 0.6$.

The result of evaluating credibility of $f1$ is: $c^{f1} = (1 - 0.0064) \times 0.224 = 0.222$

Therefore, for now, $f1$ doesn't become a belief because its credibility is under the threshold, so C doesn't decide to invest yet. Updating beliefs: $B_C = \{f3, f5, f8, f6\}$ where $f3$, $f5$ and $f6$ are supports and $f8$ is a contrast. Figure 1(a) shows the configuration of the data and beliefs network of agent C.

Now, C will try to defeat $f8$ finding some contrasts for it, therefore $\neg f8$ becomes a new goal: $G_C = \{f1, \neg f8\}$. Agent C will take advice from a stock trading expert and he says that company ACME has demonstrated historically low volatility. This will be a new datum: $f9$.

- $f9$: src_{f9} = stock trading expert, $rel(src_{f9}) = 0.9$, therefore $c^{f9} = 0.9$.

[1] Both this formula and that used in point 3 were extracted from [Paglieri 2004].

Since $f8$ has a new contrast datum, its credibility must recalculated, the result is: $c^{f8} = [1 - (1 - 0.6)] \times (1 - 0.9) = 0.06$. Due to its credibility is now less than the threshold value, $f8$ is no longer a belief.

And $f1$ credibility must be also recalculated: $c^{f1} = (1 - 0.0064) \times 0.5264 = 0.5230$.

Taking into account this result, $f1$ becomes a belief and therefore agent C decides to invest in short-term trading in stocks of company ACME.

After these calculations, the state of agent C is: $B_C = \{f3, f5, f8, f6, f1\}$, $G_C = \{\}$, $A_C = \{f3 \Rightarrow f1, f5 \Rightarrow f1, f6 \Rightarrow f1, f8 \perp f1, f9 \perp f8\}$

3.2.2. Generating arguments from new entry information

It is triggered when new relevant data arrives. Since these data are relevant, the agent calculates their credibility. Some data can become beliefs and some arguments can be generated. These arguments are saved in A_C waiting for being used at any moment.

Resuming the example, agent C, during its decision process, receives the following data: dollar price will remain low during this year ($f10$) and this will have a positive effect on bank loan interests ($f11$). Hitherto, these data don't have any relation with its goals, hence they are not relevant. But, agent C also gets another data: if loan bank interests are low ($f11$) then leverage is really riskless ($f6$). Now, the former data becomes relevant because it has relation with one of its beliefs. Since all these new data are relevant, then, agent must calculate their credibility:

- $f10$: src_{f10}= economy channel, $rel(src_{f10}) = 0.6$, therefore $c^{f10} = 0.6$.
- $f11$: src_{f11}= economy channel, $rel(src_{f11}) = 0.6$, therefore $c^{f11} = 0.6$.

Since both are greater than the threshold, the agent updates its beliefs and recalculates credibility of $f6$ and $f1$ because $f11$ is a new support for $f6$.

The final state of our agent C is: $B_C = \{f3, f5, f6, f1, f10, f11\}$ e $A_C = \{f3 \Rightarrow f1, f5 \Rightarrow f1, f6 \Rightarrow f1, f10 \Rightarrow f11, f11 \Rightarrow f6\}$, with:

$c^{f6} = 1 - (1 - 0.6)(1 - 0.8) = 0.92$

$c^{f1} = (1 - 0.0032) \times 0.5264 = 0.5247$.

It can be noticed that $f1$ value increases slightly. Which means, C will really invest in short-term trading in stocks of company ACME. Figure 1(b) shows the final configuration of the data and beliefs network of agent C.

4. Conclusions and Future Work

It is worthwhile to note that the basic process of belief revision for integrating new data or update existing one is automatic. After revision, beliefs are automatically ready to form possible arguments, which facilitates the task of generation of arguments either by necessity or arrival of new data. It's also clear that DBR model gives support for the integration of belief revision and argumentation.

Since arguments are constructed from both directions top-down (a new claim needs supports) and bottom-up (new relevant data arrive), we can claim that using DBR

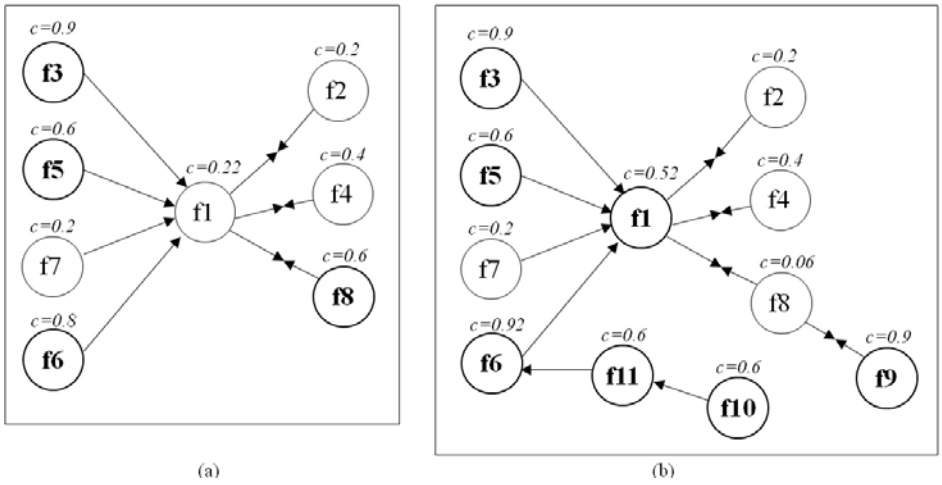


Figure 1. (a) Data an beliefs network configuration after the first credibility calculus. (b) Final data an beliefs network configuration. Beliefs are represented with bold line circles and data with normal line circles. Support relations are represented with normal arrows and contrast relations with contrasting arrows.

model in argumentation accelerates the process of argument generation and allows to generate arguments with more credibility.

Our next step is to make experiments and compare our proposal with other approaches to demonstrate that DBR model really helps to improve the process of argument generation. Others future works are argument selection and to use this proposal in persuasive dialogues between two or more agents.

References

Amgoud, L. (2005). A unified setting for inference and decision: an argumentation-based approach. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 26–33.

Besnard, P., Gregoire, E., Piette, C., and Raddaoui, B. (2010). Mus-based generation of arguments and counter-arguments. pages 239–244. Information Reuse and Integration (IRI).

Gardenfors, P. (1988). Knowledge in flux: Modeling the dynamics of epistemic states. The MIT press.

Katsuno, H. and Mendelzon, A. (1991). Propositional knowledge base revision and minimal change. pages 253–294. Artificial Intelligence.

Ontanon, S. and Plaza, E. (2007). Arguments and counterexamples in case-based joint deliberation. In *Argumentation in Multi Agent Systems. Springer Berlin Heidelberg*, pages 36–53.

Paglieri, F. (2004). Data-oriented belief revision: Towards a unified theory of epistemic processing. In *Proceedings of STAIRS'04*, pages 179–190.

Paglieri, F. and Castelfranchi, C. (2006). The toulmin test: Framing argumentation within belief revision theories. In *Arguing on the Toulmin model. Springer Netherlands*, pages 359–377.

Trabalhando Reputação em SMA com a utilização de Artefatos Normativos

Henrique Donâncio N. Rodrigues[1], **Glenda Dimuro**[2], **Graçaliz P. Dimuro**[1], **Diana F. Adamatti**[1], **Esteban de Manuel Jerez**[2]

[1]Universidade Federal do Rio Grande - FURG

[2]Depto de Expresión Gráfica Arquitectónica, Universidad de Sevilla, Sevilla, Espanha

{henriquedonancio,gracaliz,dianaada}@gmail.com

Abstract. *This paper presents the modeling and simulation of a multiagent system applied to the process of reputation join with normative political social organization of Urban Horta San Jerónimo in Seville, Spain. For that, the rules of the organization were adapted by MSPP framework (Modeling and Simulation of Public Policies), developed for modeling and simulation of public policy at Jacamo platform*

Resumo. *Este artigo apresenta a modelagem e simulação de um sistema multiagente aplicado ao processo de reputação aliado política normativa da organização social da Horta Urbana Sán Jerónimo, em Sevilha na Espanha. Para tanto, foram adaptadas as normas da organização através do arcabouço MSPP (Modeling and Simulation of Public Policies), desenvolvido para modelagem e simulação de polícas públicas na plataforma JaCaMo.*

1. Introdução

Este trabalho é parte de um projeto que tem como objetivo geral o desenvolvimento de ferramentas SMA para simulação de processos de produção e gestão social de ecossistemas urbanos, em particular, o projeto social da Horta Urbana "San Jerónimo", localizada em Sevilha, Espanha, coordenado pela associação "Ecologistas en Acción" (EA).

O projeto tem o intuito de fornecer ferramentas de simulação para análise/avaliação/experimentação desses processos no sentido de auxiliar no fomento à participação social em práticas de agricultura orgânica urbana coletiva.

Este artigo apresenta um SMA para simular reputação de agentes baseado no modelo proposto em [Hübner et al. 2009], por meio da aplicação das políticas internas ou normas regulamentares da organização social da Horta San Jerónimo, utilizando a plataforma Jason [Bordini et al. 2007], um interpretador da linguagem AgentSpeak(L), baseada na arquitetura BDI [Rao 1996, Rao and Georgeff 1992], juntamente com o arcabouço CArtAgO [Ricci et al. 2013] e o arcabouço para prover a simulação de políticas públicas MSPP (Modeling and Simulation of Public Policies) [Santos and Rocha 2012, Santos et al. 2012].

O artigo está organizado como descrito a seguir: A Seção 2 aborda uma síntese sobre Sistemas Multiagentes, agentes BDI e a plataforma JaCaMo. A Seção 3 irá explanar

sobre o ambiente em que as políticas públicas foram inseridas. A Seção 4 aborda características do arcabouço utilizado para inserção de políticas públicas. A Seção 5 apresenta alguns modelos de reputação como também apresenta a implementação de um modelo para a Horta San Jerónimo e a Seção 6 expõe a conclusão sobre o trabalho, e perspectivas futuras.

2. A Horta San Jerónimo

A Horta San Jerónimo é um projeto social coordenado pela associação *Ecologistas en Acción* (EA), na cidade de Sevilha, Espanha, e foi escolhida para este trabalho por ter um regulamento próprio que busca o melhor convívio e participação entre os seus agentes, além de resguardar seus direitos e atribuir-lhes restrições.

O regulamento da horta (tabela de normas) é um conjunto de quarenta normas no total. Nele estão incluídos quatro diferentes tipos de normas: as normas de *direito*, que concedem ao agente o direito de determinada ação ser executada sem que haja restrições e desde que não infrinja outras normas; as normas de *permissão* as quais o agente necessita requisitar junto a outros agentes a possibilidade de executar a ação prevista; normas de *obrigação* que são aquelas onde o agente é em determinado período obrigado a executar a ação; e normas de *proibição*, as quais restrigem ações que os agentes possam vir a executar.

Nesse contexto, baseando-se em [Santos et al. 2012] foram identificados dentre os papéis do projeto social, características que se adéquam aos papéis de agentes sociais, governamentais e também o agente emissor de políticas públicas.

3. Sistemas Multiagentes e a plataforma JaCaMo

Os Sistemas Multiagentes (SMA) são ambientes *habitados* por vários agentes que são capazes de interagir, trocar informações, são sensíveis a percepções, se adaptam às mudanças, tem conhecimentos sobre o ambiente (pleno ou parcial) e podem tomar ações (coordenadas entre si ou não) para modificá-lo dentro da chamada *esfera de influência*.

O modelo de agente BDI *(Believe, Desires and Intentions)* é caracterizado pelo aspecto cognitivo, apresentando crenças, desejos e intenções. Crenças representam a informação que o agente tem sobre outros agentes e sobre o ambiente, já os desejos expressam os objetivos que esse agente tenciona atingir, e as intenções são metas que o agente se comprometeu a cumprir.

A plataforma JaCaMo [Bordini and Hübner] é um arcabouço para programação de Sistemas Multiagentes constituído de três ferramentas.

3.1. Jason

O Jason[Bordini et al. 2007] é um interpretador da linguagem AgentSpeak(L), baseada na arquitetura BDI. Um aspecto importante dessa plataforma é sua implementação em Java, e portanto multi-plataforma. A comunicação entre agentes no Jason é baseada na teoria de atos de fala. Os agentes ao se comunicarem, geram crenças e estas por sua vez podem desencadear planos.

3.2. CArtAgO

O arcabouço CArtAgO (Common ARTifact infrastructure for AGents Open environments) [Ricci et al. 2013] é baseado no modelo Agentes e Artefatos (A & A) para modelar e projetar Sistemas Multiagente. Com essa ferramenta é possível criar artefatos estruturados em espaços abertos onde agentes podem se unir de forma a trabalhar em conjunto.

Os Artefatos são recursos e ferramentas construídas de forma dinâmica, usados e manipulados por agentes para apoiar/realizar suas atividades individuais e coletivas. O ambiente como também os recursos disponíveis no mesmo podem ser modelados na forma de um artefato CArtAgO.

3.3. Moise+

O modelo organizacional MOISE+ [Hübner 2003] é uma ferramenta com intuito de modelar a organização de SMA. Consiste na especificação de três dimensões: a estrutural, onde definem-se papéis e ligações de heranças e grupos; a funcional, onde é estabelecido um conjunto de planos globais e missões para que as metas sejam atingidas; e a deôntica, que é a dimensão responsável pela definição de qual papel tem obrigação ou permissão para realizar cada missão.

4. O arcabouço MSPP (Modeling and Simulation of Public Policies) e sua aplicação na HSJ

O arcabouço MSPP [Santos and Rocha 2012, Santos et al. 2012] para inserção de políticas públicas concretiza-se no formato de artefatos no modelo CArtAgO. Estão incluídos neste arcabouço dois tipos de artefatos normativos: NormObrig e NormPrb, modelando normas de obrigação e proibição, respectivamente.

Além dos artefatos, estão previamente inseridos agentes para executar/verificar tais normas. São eles o agente governamental, responsável por emitir as normas, os agentes sociais que estão submetidos às normatizações e buscam atingir objetivos próprios, e também os agentes governamentais detectores/efetuadores responsáveis por detectar o cumprimento das normas da política como também características e recursos do ambiente, aplicar possíveis sanções a ações que caracterizarem o descumprimentos de normas e regularizar os recursos disponíveis no ambiente.

Em um estudo preliminar, verificou-se que o arcabouço MSPP poderia atender as necessidades normativas para o estudo de caso da Horta San Jerónimo, salve algumas modificações discutidas em [Rodrigues et al. 2013]. Dentre as modificações feitas para adequar a estrutura das normas ao caso, está adaptação de seus parâmetros, uma vez que o arcabouço MSPP é maleável nesse sentido.

5. Reputação dos agentes

Reputação é inserida neste trabalho como a informação que um grupo tem sobre um indivíduo, tendo como aspectos avaliativos os resultados que este indivíduo obtêm dentro da organização, a sua obediência em detrimento das regras que lhe foram atribuídas e sua participação no grupo.

Estas informações são armazenadas no "Artefato Reputação", uma unidade centralizada responsável por guardar informações relativas ao desempenho dos agentes sociais em detrimento da política normativa.

5.1. Modelos de reputação

Pode-se dividir modelos de reputação em dois grupos: Modelos de reputação centralizados, onde a informação em relação a um agente geralmente é obtida de uma fonte única, e modelos descentralizados, onde a avaliação geralmente dependem da interação entre os agentes envolvidos no processo.

O modelo de reputação centralizado é comumente utilizado em sistemas on-line de comércio eletrônico onde toda a informação sobre a reputação é gerenciada de forma centralizada. Uma aplicação dessa abordagem é o modelo SPORAS [Zacharia and P. 2000].

O modelo de reputação descentralizado oferece a cada agente o poder de realizar sua própria avaliação sobre a reputação de outros agente, sem depender de uma unidade central. Alguns mecanismos adotam esse modelo, como Jurca e Faltings [Jurca and Faltings 2002], Regret [J. 2003, Sabater and Sierra 2001] e TRAVOS [Teacy et al. 2005].

5.2. Artefato de Reputação

O Artefato de Reputação é baseado no modelo CArtAgO de artefato. Este artefato, guarda valores tais como a participação do agente no grupo (quantidade de reuniões e palestras que ele frequentou), sua obediência (quantidade de obrigações cumpridas) e resultados.

O agente Admin (agente detector/efetuador das normas) é responsável por inserir estas informações, ou seja, fazer operações de atualizações sobre o desempenho dos agentes sociais, por se tratar de um agente governamental responsável por verificar o cumprimento das normas.

Cada agente tem como crença um respectivo valor referente ao grau de importância que ele atribui a estes aspectos que são compartilhados entre o grupo. A avaliação individual dada por cada individuo é dada pela Equação (1) [Hübner et al. 2009]:

$$\frac{\gamma p(\alpha) + \delta o(\alpha) + \epsilon r(\alpha)}{\gamma + \delta + \epsilon} \tag{1}$$

Onde os fatores γ, δ e ϵ definem a importância da participação, obediência e resultados, respectivamente.

Esses fatores são independentes entre os agentes sociais, podendo assumir valores por convenção entre 1 a 10, definindo assim o grau de relevância que o agente determina para o atributo multiplicado pelo fator. Dessa forma, o valor mínimo dado para reputação de determinado agente é 0 e o valor máximo é 1.

Ao início de uma nova simulação, cada agente é responsável por inserir seu registro no Artefato de Reputação, que então define uma "propriedade observável" (crença) aos

agentes para consulta sobre os valores de participação, obediência e resultados. Nota-se que quando o agente se inscreve no Artefato de Reputação, assim como no modelo SPORAS, os valores são os mínimos possíveis para a reputação. Por convenção adotamos 0 para os três atributos.

A cada novo evento relacionado as normas, o valor da reputação é alterado e a base de crença dos agentes é atualizado. Caso algum agente seja submetido a assembleia, estes valores são buscados na base de crença dos agentes votantes e multiplicados cada qual pelo seu fator de importância. Dependendo do resultado dessa composição, os agentes podem votar a favor ou contra a permanência do agente votado.

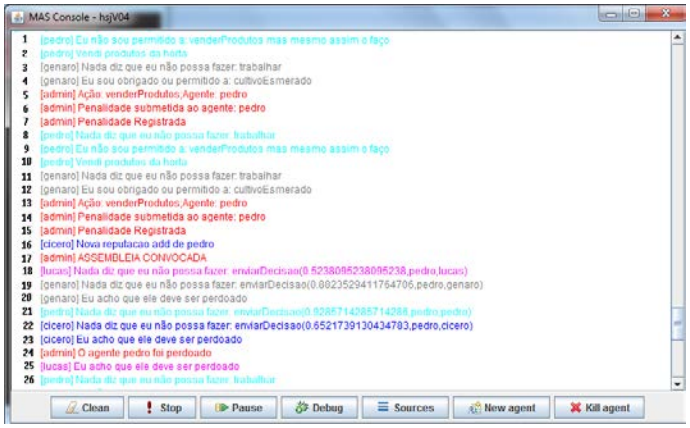


Figura 1. Simulação da política normativa e votação baseada na reputação

No exemplo da Figura 1, o agente "pedro" possui o critério de analisar que a ação "vender produtos" é uma ação que corresponde a uma infração das normas do regulamento do projeto, mas mesmo assim o decide executar (linha 1). Quando o agente reincide na infração pela sua terceira vez (linhas 13 e 14), o agente Admin convoca uma assembleia para decidir sua permanência (linha 17). Todos os agentes sociais votam, exceto o agente que esta sendo votado. O valor da composição dodo pela Equação (1) é enviado pelo agente (linhas 18 e 22) e sua decisão obedece o mínimo valor estabelecido para o agente permanecer.

6. Conclusões e trabalhos futuros

Nessa implementação baseada no modelo proposto em [Hübner et al. 2009] e utilizando o arcabouço MSPP, entende-se que a utilização da reputação como fator de decisão e relacionamento entre os agentes foi satisfatória, visto que o processo de decisão na assembleia em uma primeira abordagem do problema em [Rodrigues et al. 2013] era randômico e portanto imprevisível. Foi introduzida modificações em relação ao modelo de reputação originalmente proposto. Os fatores de importância que compõem o desempenho do agente são independentes entre cada agente, tornando assim a avaliação dos agentes descentralizada.

Em trabalhos futuros pretende-se atribuir pesos as ações dos agentes, assim como o modelo Regret [J. 2003, Sabater and Sierra 2001], para que ações recentes tenham mais relevância na avaliação da reputação. Espera-se também adotar uma classificação para as interações entre os agentes sociais, assim como o modelo TRAVOS [Teacy et al. 2005].

Agradecimentos: Este trabalho é suportado pelo CNPq (Proc. 305131/2010-9, 481283/2013-7, 306970/2013-9) e pelo Projeto RS-SOC (FAPERGS Proc. 10/0049-7).

Referências

Bordini, R. H. and Hübner, J. F. JaCaMo project. Available at http://jacamo.sourceforge.net/, accessed in September 2012.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley, New Jersey.

Hübner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Universidade de São Paulo.

Hübner, J. F., Vercouter, L., and Boissier, O. (2009). *Instrumenting Multi-Agent Organisations with Artifacts to Support Reputation Processes*. Springer.

J., S. (2003). *Trust and Reputation for Agent Societies*. PhD thesis, Universitat Autònoma de Barcelona, Barcelona.

Jurca, R. and Faltings, B. (2002). Towards incentive-compatible reputation management. In *Trust, Reputation, and Security: Theories and Practice*, Lecture Notes in Computer Science, pages 138–147.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. pages 42–55.

Rao, A. S. and Georgeff, M. P. (1992). An abstract architecture for rational agents. pages 439–449.

Ricci, A., Santi, A., and Piunti, M. (2013). CArtAgO (common atifact infrastructure for agents open environments).

Rodrigues, H. D. N., Santos, F. C. P., Dimuro, G., Adamatti, D. F., Jerez, E. M., and Dimuro, G. P. (2013). A mas for the simulation of normative policies of the urban vegetable garden of san jerônimo, seville, spain. In *Proceedings of the Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações - WESAAC 2013*, pages 79–84.

Sabater, J. and Sierra, C. (2001). Reputation model for gregarious societies. In *Proceedings of the Fourth Workshop on deception Fraud and Trust in Agent Societies*, pages 61–70.

Santos, I. and Rocha, A. C. R. (2012). Toward a framework for simulating agent-based models of public policy processes on the jason-cartago platform.

Santos, I. A. S., P., M. F., Costa, A. C. R., and Dimuro, G. P. (2012). Um framework para simulação de políticas públicas aplicado ao caso da piracema, sob o olhar da teoria dos jogos.

Teacy, W. T. L., Patel, J., Jennings, N. R., and Luck, M. (2005). Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 997–1004.

Zacharia, G. and P., M. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence Journal*, 14(9):881–908.

Modelo de VANT Autônomo Baseado em uma Arquitetura BDI

Fernando Rodrigues Santos[1], **Jomi Fred Hübner**[1], **Leandro Buss Becker**[1]

[1]Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brazil

fernando.rod.santos@gmail.com, jomi.hubner@ufsc.br, lbecker@das.ufsc.br

Abstract. *In recent decades several applications for Unmanned Aerial Vehicles (UAVs) have emerged. There are advances in order to provide the UAV more autonomy, but the models currently used for these applications have a programming style which the application code and autonomous coding are mixed. The objective of this work is to develop a model of autonomous behavior for UAVs via a BDI architecture, exploring the potential of this architecture in solving the problem. We also plan to analyse the differences between the existing implementation techniques and the proposed model.*

Resumo. *Nas últimas décadas surgiram várias aplicações para Veículos Aéreos Não-Tripulados (VANTs). Existem avanços no sentido de dotar o VANT de mais autonomia, mas os modelos utilizados atualmente para essas aplicações possuem um estilo de programação onde o código da aplicação e da autonomia são misturados. O objetivo deste trabalho é desenvolver um modelo de comportamento autônomo para VANTs através de uma arquitetura BDI, explorando as potencialidades desta arquitetura na resolução do problema. Também pretendemos analisar as diferenças em relação às técnicas de implementação existentes.*

1. Introdução

As possibilidades de aplicações utilizando Veículo Aéreo Não-Tripulado (VANT) são inúmeras, indo desde monitoramento de regiões por times de VANTs até filmagens de shows e entrega de pizzas, como abordado em [Fahlstrom and Gleason 2012]. Na maioria das aplicações um operador humano é responsável pela operação remota destes equipamentos. Em alguns casos, porém, é necessário, ou desejável, que o veículo possua autonomia para realizar algumas tarefas. E o controle do VANT por rádio frequência limita o alcance das missões. Um veículo com maior autonomia poderia tanto dispensar o operador de atividades cansativas e sujeitas a erro, quanto possuir mecanismos automáticos mais precisos e otimizados. Neste contexto, considera-se autônomo um veículo que não precise da operação constante de um humano e para o qual possa ser atribuídos objetivos. Assim, o veículo tem autonomia para escolher um plano de ações para atingir um objetivo, dado por um operador humano, porém deve se comprometer com tal objetivo, ou seja, ele não tem autonomia para ignorá-lo. O operador humano, no lugar de controlar diretamente os motores e flaps, ele passaria a controlar o VANT por meio da delegação de objetivos, tais como passar por determinados lugares (waypoints) ou encontrar uma pessoa perdida em uma floresta.

O tema autonomia é estudado há vários anos na área de Sistemas Multi-Agentes (SMA), sendo produzidas teorias, arquiteturas de software e mesmo linguagens de programação especificamente voltadas para o desenvolvimento deste tipo de software, chamado de agente autônomo, segundo [Wooldridge 2002]. Uma das vantagens deste tipo de linguagem é seu alto nível de abstração, com primitivas como objetivos, planos e ações que permitem ao desenvolvedor definir o comportamento e a autonomia do agente. Destacam-se na área as linguagens baseadas na arquitetura BDI (Belief, Desire, Intention) que facilitam o desenvolvimento de agentes que apresentam capacidade de reagir rapidamente a mudanças em seu ambiente e possuir objetivos de longo prazo, conforme [Rao and Georgeff 1995] e [Bratman 1987].

Evidentemente, linguagens de programação convencionais podem ser usadas para desenvolver agentes e veículos autônomos, porém o código final é muitas vezes mais complexo, misturando programação da aplicação com a programação do comportamento e autonomia. Por outro lado, linguagens BDI são ainda novas, pouco testadas em aplicações reais e seus interpretadores são pouco otimizados.

Neste artigo apresentamos um projeto para o desenvolvimento de um modelo de comportamento autônomo para tomada de decisão de um VANT, através de uma abordagem com arquitetura de agentes BDI, além de sua integração com o sistema embarcado do VANT. Como se trata de um trabalho em andamento, ainda não temos resultados.

2. Projeto ProVant

O presente trabalho faz parte do projeto ProVant[1], que propõe o desenvolvimento de um VANT de baixo custo, de caráter geral, voltado para aplicações civis. O desenvolvimento prevê a modelagem, o controle, a concepção e a construção. Esta é uma proposta diferenciada em termos de desenho e de custo reduzido em relação ao que existe no mercado. Ao invés de se trabalhar com veículos aéreos em escala reduzida, como aeronaves de asas fixas ou aeronaves de asas rotativas com quatro rotores, o VANT deste projeto possui apenas dois rotores, cada um associado a um servomotor para realizar a rotação longitudinal dos rotores, constituindo assim uma aeronave na configuração Tiltrotor, conforme ilustrado na Figura 1.

3. Hardware-In-the-Loop (HIL)

A simulação Hardware-in-the-loop (HIL) é descrita como a técnica aplicada ao desenvolvimento, teste e validação de sistemas de tempo real embarcados complexos. Esta técnica tem por objetivo a união entre o ambiente simulado e o hardware real, de forma a abstrair os riscos físicos da modelagem, assim como, a aplicação dos sistemas diretamente na estrutura final da aplicação, conforme [Louall et al. 2011].

Uma simulação HIL inclui a representação dos sensores e atuadores e os sinais gerados estabelecem uma interface entre a planta simulada e o sistema embarcado. Os dados da simulação são encaminhados como leitura para o sistema embarcado. A partir desta troca de informações os subsistemas são alimentados e, de acordo com os testes propostos, o comportamento tanto do sistema embarcado quanto do modelo computacional pode ser verificado.

[1] Site: http://provant.das.ufsc.br

Figura 1. Protótipo do VANT

4. Trabalhos Relacionados

O trabalho de [Hama 2012] aborda a aplicação do paradigma da programação orientada a agentes para controle inteligente de comportamento de VANTs, com a concepção de um framework, que visa prover uma abstração para a programação de comportamentos inteligentes em VANTs. Neste trabalho é proposto um modelo UAVAS, que é um framework de programação de comportamentos para VANTs, executado dentro do sistema operacional do hardware embarcado do VANT. A arquitetura do modelo baseia-se em um esquema de fluxo e conversão de dados, onde o agente possui dois tipos de interação com o ambiente: ação ou percepção.

No trabalho de [Chaves 2013] é proposto um modelo de VANTs cooperativos que combina mecanismos de coordenação multi-agente, algoritmos de navegação e padrões de busca e salvamento. A arquitetura geral do VANT possui dois mecanismos, um reativo e outro deliberativo. O controle reativo é responsável por manter a estabilidade da aeronave e a rota, além de responder a situações de emergência. Já o controle deliberativo tem as funções de planejamento de trajetória e de coordenação com outros VANTs, também busca evitar colisões por meio do planejamento da rota.

E o trabalho de [Selecký and Meiser 2012] apresenta algumas soluções para os problemas no processo de integração de VANTs de hardware em um sistema de simulação multi-agente com VANTs virtuais adicionais, compondo em um sistema de realidade mista, onde VANTS de hardware e VANTs virtuais podem coexistir, coordenar o seu voo e cooperar em tarefas comuns. Esses VANTs de hardware são capazes de realizar planejamento on-board e raciocínio, podem cooperar e coordenar seus movimentos uns com os outros, e também com os VANTs virtuais. Os VANTs de hardware foram integrados ao sistema AgentFly e o sistema foi modificado para permitir que tanto VANTs de hardware e VANTs virtuais pudessem agir e interferir em um ambiente comum de realidade mista.

Nota-se que alguns trabalhos modelam um VANT como um SMA, outros trazem apenas resultados em ambiente simulado, enquanto outro realizou testes com um VANT real acoplado a um simulador. Esses trabalhos exploram as potencialidades do SMA para tomada de decisão em grupo de VANTs e outras técnicas computacionais ou específicas da

área do problema. Porém, esses trabalhos não exploram as potencialidades da arquitetura BDI, como capacidade de raciocínio rápido e de ter objetivos de longo prazo, para dotar o VANT de autonomia em suas aplicações.

5. Proposta de Trabalho

O presente projeto pretende desenvolver um modelo do comportamento do VANT baseado no trabalho de [Hama 2012], porém não será utilizado um SMA em um VANT e sim um agente BDI para um VANT, e assim, explorar as potencialidades da arquitetura BDI. A arquitetura do projeto será baseada no trabalho de [Chaves 2013], com uma camada de baixo nível com os controles de estabilidade e de trajetória, e uma camada de alto nível de tomada de decisão. Os testes serão inspirados no trabalho de [Selecký and Meiser 2012] e seu modelo de integração, mas em vez de integração com o VANT de hardware, pretende-se realizar uma integração da plataforma embarcada e a simulação de um VANT, com o uso da técnica HIL.

Além disso, uma abordagem com sistemas híbridos com agentes e outras técnicas computacionais para solução de problemas com VANTs, como nos trabalhos de [Han et al. 2013] e [Chen et al. 2013], pode ser uma estratégia utilizada ao longo do desenvolvimento da solução para este projeto.

5.1. Objetivo

O objetivo deste trabalho é desenvolver um modelo de comportamento para tomada de decisão de um VANT autônomo, utilisando uma abordagem com arquitetura de agentes BDI, e sua integração no sistema embarcado do VANT.

5.2. Cenário de Aplicação

O cenário inicialmente escolhido para avaliar a proposta é o de coleta de dados em uma rede de sensores sem fio, através do seguimento de trajetória que passa pelos pontos onde se encontram os sensores.

A missão do VANT será realizar a coleta de informações dos sensores de uma "região x", sendo essa região conhecida pelo sistema e tendo a referência da posição de cada um dos sensores da rede. O VANT parte de uma estação-base e percorre a região pelos nodos sensores que estão mais próximos. Ele deve se aproximar de cada nodo até que o sinal com a informação do sensor seja coletado ou seguir até o ponto exato do sensor, visto que, sensores sem fio utilizam bateria e esta pode descarregar e o sensor parar de funcionar. Caso o VANT consiga captar o sinal do sensor antes de chegar até ao ponto coordenado em que o nodo se situa, o VANT pode passar para o próximo sensor em sua trajetória. Caso o VANT se desloque até o ponto exato onde se encontra o sensor e não capte o sinal, o VANT deve considerar que aquele sensor está sem bateria e passa para o próximo ponto da trajetória.

O VANT deve considerar o *ponto de não retorno*[2]. Durante o voo, o VANT deve monitorar a carga disponível em sua bateira, a distância até a estação base e o seu consumo médio (carga/metro). Assim, caso identifique que não conseguirá percorrer todos

[2]Trecho da rota de um voo em que o combustível restante na aeronave não é suficiente para que o avião retorne ao aeroporto de partida em caso de emergência.

os pontos da rede de sensores devido a sua carga de bateria, o VANT deve retornar para estação-base para recarregar a bateria, informar a estação-base que não concluirá a missão de coleta de dados da região e dessa forma solicitar a sua substituição por outro VANT.

6. Metodologia

A metodologia utilizada para este projeto consiste no desenvolvimento do modelo proposto, a realização de testes de simulação, a comparação com outros modelos e a integração deste no sistema embarcado do VANT.

Inicialmente, será realizada a investigação sobre o uso do software Jason[3], que é uma linguagem de programação de agentes baseado na arquitetura BDI, em uma plataforma embarcada, a qual será integrada ao sistema embarcado do VANT do ProVant.

Em seguida, iniciará o desenvolvimento de um modelo do comportamento de VANTs autônomos com a arquitetura de agentes BDI. Este modelo deve ser capaz de dotar o VANT das capacidades necessárias para realização da missão, de modo a garantir a autonomia desejada.

Para se integrar o modelo gerado em Jason no VANT do ProVant, será necessário o desenvolvimento de uma interface entre o Jason e o sistema embarcado do VANT, de modo a executar as ações do sistema de tomada de decisão no VANT.

Após isso, serão realizados testes de simulação, utilizando a técnica HIL, para avaliação do modelo gerado em Jason com a simulação do VANT do ProVant. A simulação que representa o VANT real já foi desenvolvida no ProVant e será utilizada nesta parte do trabalho.

Por fim, pretende-se comparar o modelo proposto e os modelos utilizados para projeto de VANTs autônomos existentes. Para isso, será realizada a análise dos resultados por meio de simulação ou em sistema real. Além de observar a qualidade do código gerado por ambas abordagens, quanto ao tamanho do código final e facilidade de se alterar ou inserir novas funcionalidades ao modelo final.

7. Considerações Finais

A arquitetura do sistema embarcado do VANT do ProVant será composta por uma camada de baixo nível, na qual se encontram os controles de estabilidade e de seguimento de trajetória da aeronave, e por uma camada de alto nível, onde se encontra o modelo de comportamento em agentes BDI para tomada de decisões do VANT. O trabalho se concetra no desenvolvimento desse modelo e do mecanismo de interface para integração entre essas duas camadas do sistema, como pode ser observado na Figura 2

O modelo proposto será testado no cenário de aplicação apresentado na proposta do trabalho, porém, outros cenários de aplicação podem ser estabelecidos ao longo do desenvolvimento do projeto, visando explorar as potencialidades do modelo frente a outras abordagens utilizadas em VANTs autônomos.

Ao final do projeto, pretende-se saber quais as vantagens e desvantagens que um modelo baseado em agentes BDI pode trazer ao desenvolvimento de aplicações com VANTs.

[3][Bordini et al. 2007] e Site: http://jason.sourceforge.net/wp/

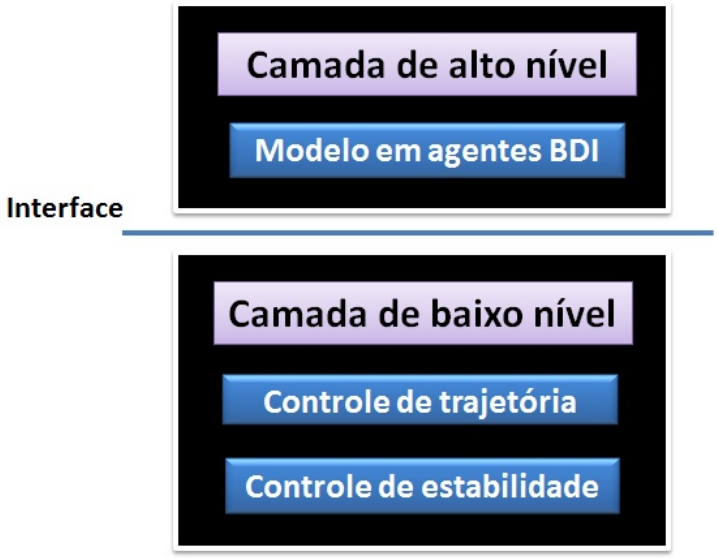


Figura 2. Arquitetura do VANT

Como resultado do projeto, teremos um modelo de comportamento para VANTs com alto nível de abstração, que seja mais intuitivo alterar e inserir novas funcionalidades ao código final.

Referências

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. Wiley-Interscience.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press.

Chaves, q. N. (2013). Proposta de modelo de veículos aéreos não tripulados (vants) cooperativos aplicados a operações de busca. Master's thesis, USP - Universidade de São Paulo.

Chen, X., He, W., and Wu, Z. (2013). Research of uav's multiple routes planning based on multi-agent particle swarm optimization. *Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*.

Fahlstrom, P. G. and Gleason, T. J. (2012). *Introduction to UAV Systems*. Wiley.

Hama, M. T. (2012). Uma plataforma orientada a agentes para o desenvolvimento de software em veículos aéreos não-tripulados. Master's thesis, UFRGS - Universidade Federal do Rio Grande do Sul.

Han, J., Wang, C.-h., and Yi, G.-x. (2013). Cooperative control of uav based on multi-agent system. *IEEE*.

Louall, R., Belloula, A., Djouadi, M., and Bouaziz, S. (2011). Real-time characterization of microsoft flight simulator 2004 for integration into hardware in the loop architecture. *Control Automation (MED) - 19th Mediterranean Conference*.

Rao, A. S. and Georgeff, M. P. (1995). Bdi agents: from theory to practice. *Proceedings of the First International Conference on MultiAgent Systems*.

Selecký, M. and Meiser, T. (2012). Integration of autonomous uavs into multi-agent simulation. *Acta Polytechnica*, 52(5).

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. Chichester: John Wiley and Sons Ltd.

A Multi-agent approach for devices management and control in IoT environments

Renato L. Cagnin[1], Ivan R. Guilherme[1], Jonas F. P. Queiróz[1], Rodrigo C. Antonialli[1]

[1]UNESP – DEMAC/IGCE
Caixa Postal 178 – 13.506-900 – Rio Claro – SP – Brazil

{r_cagnin,ivan}@rc.unesp.br, {jonnas.queiroz, rcantonialli}@gmail.com,

Abstract. *The Internet of Things (IoT) envisions millions of heterogeneous smart devices in an open, distributed and dynamic environment. IoT addresses the complexity involved in connecting, discovering and accessing all these devices, and also integrate and analyze the amount of data produced by them. In this paper is presented a proposal for the development of an architecture based on Multi-agent systems, Service-Oriented Architecture and Semantic Web technologies. The architecture aims to support the design and development of robust, flexible and scalable application to automate devices management and coordination in IoT based environments.*

1. Introduction

The Internet of Things (IoT) envisions a network of heterogeneous devices communicating with each other in an open and dynamic environment. These devices, also called smart devices, represent sensors and actuators connected in the Internet, providing data and other resources from their environment for other systems. The advances in hardware and network infrastructure have contributed more and more to the development of cheaper, smaller and more powerful devices, endowed with more processing capabilities, and also wireless network interfaces. These network interfaces provide the devices with the communication capabilities, necessary to connect to the network, publish and provide their resources, and interact with each other and with their surrounding environment [Karnouskos and Tariq 2009].

Smart devices have been used in a wide range of environments, also called smart environments, such as home, industry and also open and urban areas, for a variety domain of application [Ruta, M. et al., 2013]. Most often such applications aim to provide users with information to monitor and control the environment. However, just integrate and provide users with data measured from the environment may not be enough to address user intends. It means that in a smart environment the application can integrate and coordinate devices, providing high level information and resources in order to support users in the achievement of complex tasks and automate the execution of related processes.

The complexity inherent in smart environments arises primarily by the distribution and amount of heterogeneous components (devices and systems), as well as the large volume of produced data and high communication among components. This problem arises from the different communication technologies and data models used by these components. In this sense, service-oriented architectures (SOA) are a promising technology to support the interoperability among different components [Ruta et al.

2013]. In the last years, service-oriented technologies have been adopted in order to handle these problems, creating standardized interfaces and mechanisms for publishing and discovering new devices connected to the network. Examples of such technologies are OPC-UA and DPWS (Devices Profile for Web Services) [Izaguirre, Lobov and Lastra 2011].

In this context, new approaches for devices management and control must arise in order to ensure the cooperation and coordination of the connected devices. The Multi-Agent System (MAS) approach provides a set of appropriate abstractions for dealing with the design and development of distributed, open, dynamic and complex heterogeneous systems [Jennings and Wooldridge 1998], such as IoT environments. In a MAS approach a set of agents is designed as independent and autonomous components responsible to create organizations and cooperate in order to achieve the required task.

In addition, Semantic Web technologies such as ontologies have been used to overcome the semantic heterogeneity inherent from the variety of data models used by the smart devices. Such technologies are employed in order to increase the data models interoperability and the systems tasks automation, by defining common vocabularies for representing the application knowledge. Thus providing better data integration and processing [Aloulou et al. 2013].

In this work we present a proposal of a reference model architecture that integrates MAS approach, SOA and Semantic Web technologies. The architecture organization and components were defined and specified in order to be used as a reference model to design and develop applications for automating the monitoring and controlling tasks in IoT based environments.

2. Related work

The IoT is a research field that has emerged with the increasing usage of smart devices in many different domains. So, this research field aims to investigate and propose solutions to address the new challenges posed by the IoT environments. Such challenges are mainly related to the connection, discovery, access, and integration of heterogeneous smart devices and also to retrieve, integrate and process the large amount of data produced [Alberti and Singh 2013].

In this context, many approaches can be found in the literature proposing a variety of solutions in order to support IoT features, such as network protocols and frameworks [Presser et. al. 2009].

In the IoT SOA approaches, based on Web Services and RESTful, have been widely adopted to ensure the connectivity and interoperability necessary for accessing and integrating several heterogeneous devices [Stirbu 2008]. Also others light-weight implementations are adopted, such as DPWS and OPC-UA [Izaguirre, Lobov and Lastra 2011], which comprises a collection of standards that empowers embedded devices allowing them to run web services natively [Sucic; Bony and Guise 2012].

Semantic Web technologies, such as ontologies, have been used for knowledge representation and sharing, improving system interoperability and data integration. Ontologies can also be used to semantically annotate services provided by devices in order to improve and automate their discovery, access and composition. In this sense, domain and application ontologies should be specified to define a common vocabulary

that describes the domain concepts such as devices, data models, events, monitoring and control tasks [Song, Cardenas and Masuoka 2010]. There are already several ontologies developed for different domains that can be used and adapted for this purpose. Through the use of ontologies, application data become machine readable and interpretable, enabling agents to use it to automate their tasks.

In [Aloulou et al. 2013], ontology and rules are used to represent the knowledge about entities in a collaborating nursing home environment. This collaborating environment involves patients, caregivers, assistive services, sensors and interaction devices. The system behavior is determined by reasoning through the semantic model and defined rules. The semantic model keeps the knowledge about patient conditions and needs.

Ontologies are also used to describe the application context in order to perform the execution of most adequate services in a given scenario. The context-awareness enables the automatic modification of the system behavior according to the current situation with minimal human intervention. In [Han, Lee, and Crespi 2014] is presented an architecture for treating data acquired by devices and coordinate them according to the user environment context and the context rules.

MAS approach is a paradigm suitable for designing and developing distributed and heterogeneous systems that involve complex interaction between entities, e.g. humans, industrial robots, and smart devices. Therefore MAS approaches have been applied to ensure organization and cooperation between devices in IoT applications [Angulo-Lopez and Jimenez-Perez 2012]. In [Karnouskos and Tariq 2009], a MAS approach is used for simulating a dynamic environment containing distributed devices in IoT. In this MAS a set of agents with different roles is in charge to handle various tasks along the system.

3. Proposed architecture

In the architecture proposed, presented in Figure 1, it is defined a set of components to support the development of IoT applications for environment monitor and control, which integrates MAS, SOA and Semantic Web features. The architecture is composed of different layers responsible for a set of different tasks:

Devices Layer: corresponds to the physical devices (sensors and actuators) connected to the network, responsible for sensing and modifying the physical environment.

Service Layer: comprises a service middleware, responsible for wrapping devices resources and creating an interoperability layer for publishing and discovering their services on the network. The Service Layer provides an infrastructure that allows the agents of MAS to easily and seamlessly search, discover and access the resources. This layer is proposed to be developed using DPWS technology.

Multi-agent Layer: presents set of agents responsible for the integration and management of the resources. This layer is composed by Reactive and Cognitive Components.

- Reactive Component: It is composed by the Resources agents responsible to directly handle the physical devices provided by Service Layer.
- Cognitive Component : comprises the agents responsible to perform reasoning and inferences along the application knowledge in order to delegate tasks and

coordinate other agents in the execution of the application tasks. These agents are responsible for decision-making and plan coordination to ensure system self -adaptation.

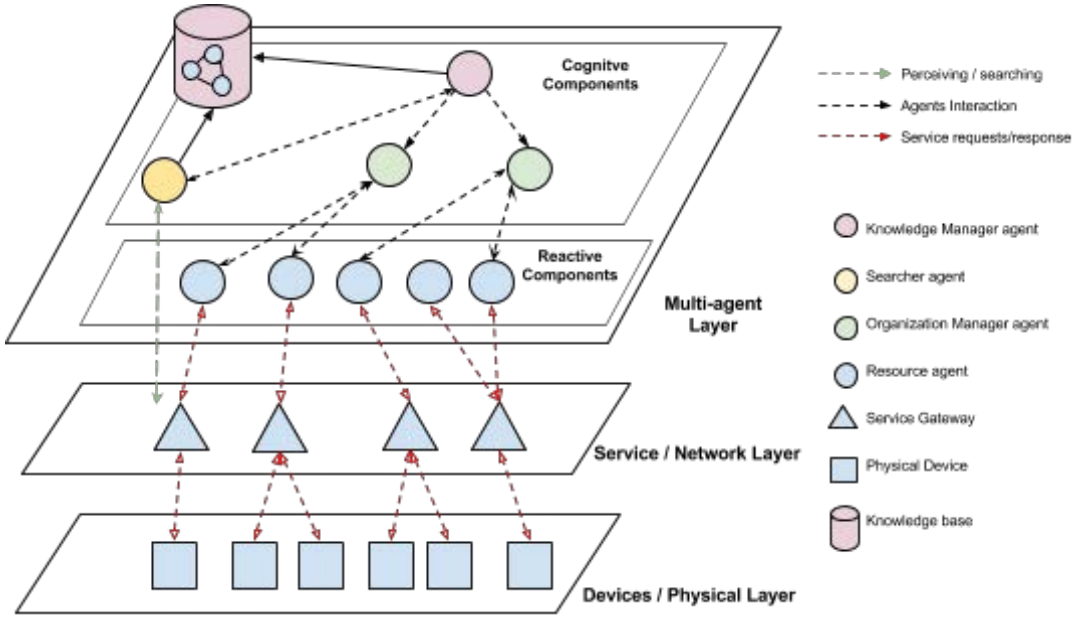


Figure 1: Proposed MAS architecture overview.

3.1. Multi-agent layer

The Multi-agent Layer is composed by one or more agents that can perform a set of different roles, described in the following:

- Searching Agent: responsible for monitoring and searching for devices in the Service Layer and initialize a Resource Agent for handling their resources. Thus, when a new device is found or connected, the agents performing this role is responsible to search for the semantic description of this device and their resources. The device semantic description need to be defined previously and stored in the application knowledge base. The knowledge retrieved through the Knowledge Manager Agent will be used to initialize a new specialized Resource agent for handling the device's resources.
- Resource Agent: are specialized for every different class of devices and have the capabilities for data processing according to the devices requirements and specifications. For example, if the device is a camera, the related agent must have all knowledge and data analysis functions necessary for image processing.
- Knowledge Manager Agent: are responsible to manage all the application knowledge that includes the semantic description of data models, devices and process. The knowledge is stored in the application knowledge base and structured according to application ontologies. This agent must monitor the Resource agents and their resources, in order to try to identify the application context. Then determining the tasks that can be executed with the resources connected and available to the system. If in a given context a complex task can be executed, it must instantiate an Organization Manager Agent and defines the tasks to be performed by each Resource agent.

- Organization Manager Agent: manages and creates the organization for a given task realization. This agent can create organizations in a dynamic way, assigning a set of goals to be achieved for each agent in the system. Another function performed by this agent is the task of management of each Resource Agent, according to their roles in the organization. This agent is also responsible for destroying or aborting organization's global goals. In the application, there can be several different organizations created according to the user needs, context and resources availability.

4. The architecture implementation

For the architecture implementation is proposed the use of a set of frameworks to achieve each of the architecture requirements. Thus, the Service Layer is implemented using a DPWS framework with a Java API called JMEDS (http://ws4d.e-technik.uni-rostock.de/jmeds/). JMEDS implements the DPWS stack protocol and will be responsible for connecting the device along the network and provide an interface between the physical devices and MAS.

The MAS will be implemented using the framework JaCaMo (Jason, CArtAgo e Moise) (http://jacamo.sourceforge.net/), which provides the infrastructure necessary for developing the agents and the communication between them. The Jason framework will be used to implement the agents. The framework CArtAgo will be used to develop the environment of the agents, where artifacts will provide an interface between the agents and Service Layer. The Moise framework will be used for the creation, management and maintaining of the agents organizations, where the structural, functional, and normative specifications of the organizations will be described.

The ontologies, coded in OWL, will be reused or created and managed using the Protégé and Jena frameworks.

Conclusion

An architecture has been proposed based on MAS, SOA and Semantic Web technologies, for application to automate the monitoring and controlling tasks in IoT based environments. Together these technologies can be used to create a robust, flexible and scalable software infrastructure for monitor and control IoT environments, managing and coordinating the smart devices in the execution of tasks according to the application context or user needs.

The work in progress, has already implemented the lower level layer of the architecture. The implementation is achieved by integrating the technologies provided by the frameworks JaCaMo and JMEDS. The MAS can already automatically detect and access any DPWS device connected on the network.

The next step is the creation of the knowledge base and the goals for the Knowledge Manager agent annotate automatically the services and identify the semantic context, through the devices and services metadata. Also with the knowledge base, the Organization Manager Agent will be able to assign roles and plans for each Resource Agent created, in order to achieve the tasks defined by the context.

References

Angulo-Lopez, P. and Jimenez-Perez, G. (2012) “Collaborative agents framework for the Internet of Things”. Workshop Proceedings of the 8th International Conference on Intelligent Environments, pages 191 - 199

Alberti A. M., Singh D. (2013) "Internet of Things: Perspectives, Challenges and Opportunities", International Workshop on Telecommunications (IWT 2013), INATEL, Santa Rita do Sapucai, May 6-9.

Aloulou, H. et al. (2013) “Deployment of assistive living technology in a nursing home environment: methods and lessons learned”. BMC medical informatics and decision making, v. 13, pages. 42.

Han, S. N.; Lee, G. M.; Crespi, N. (2014) “Semantic Context-Aware Service Composition for Building Automation System”. IEEE Transactions on Industrial Informatics, v. 10, n. 1, pages. 752–761.

Izaguirre, J. G.; Lobov, A.; Lastra, J. L. M. (2011) “OPC-UA and DPWS Interoperability for Factory Floor Monitoring using Complex Event Processing”. pages. 205–210.

Jennings, N. R. and Wooldridge M., “Applications of Agent Technology,” In Agent Technology - Foundations, Applications, and Markets, Springer-Verlag, 1998, pp. 3-27.

Karnouskos, S.; Tariq, M. M. J. (2009) “Using multi-agent systems to simulate dynamic infrastructures populated with large numbers of web service enabled devices”. 2009 International Symposium on Autonomous Decentralized Systems, pages. 1–7

Presser, M.; Barnaghi, P.M.; Eurich, M.; Villalonga, C (2009). The SENSEI project: integrating the physical world with the digital world of the network of the future. Communications Magazine, IEEE 47 (4).

Ruta, M. et al. (2013).“Semantic-Based Knowledge Dissemination and Extraction in Smart Environments”. 2013 27th International Conference on Advanced Information Networking and Applications Workshops, pages. 1289–1294.

Song Z., Cardenas A. A. and Masuoka R. (2010) “Semantic Middleware for the Internet of Things”. Internet of Things (IOT), IoT for a green Planet, Tokyo, Japan, November 29-December 1, 2010. Proceedings, 2010.

Stirbu, V. (2008) “Towards a RESTful Plug and Play Experience in the Web of Things”. 2008 IEEE International Conference on Semantic Computing, pages. 512–517.

Sucic, S.; Bony, B.; Guise, L. (2012) “Standards-compliant event-driven SOA for semantic-enabled smart grid automation: Evaluating IEC 61850 and DPWS integration.” 2012 IEEE International Conference on Industrial Technology, pages. 403–408.

Tiberghien, T., Mokhtari, M., Aloulou, H. and Biswas, J. (2012). “Semantic Reasoning in Context-Aware Assistive Environments to Support Ageing with Dementia”.11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012 Proceedings, Part II, pages 212-227.

Integrating Robot Control into the AgentSpeak(L) Programming Language

Rodrigo Wesz[1]

[1]Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS)
Porto Alegre – RS – Brazil

rodrigo.wesz@acad.pucrs.br

Abstract. *Multiple software frameworks have been created to help developers model robot applications. These frameworks use low-level programming constructs suitable to control hardware components, such as sensors and actuators, but are limited in abstracting complexity. Conversely, agent programming languages support the implementation of agents using a higher level of abstraction, but these languages have been mostly restricted to the development of software agents. In this paper, we outline an architecture and programming constructs that integrate an agent programming language with a robot development framework in order to program autonomous robots using a higher-level abstraction. The resulting programming environment aims to facilitate the modeling of complex behaviors on robots using the abstraction of autonomous cognitive agents.*

1. Introduction

Programming robots is a complex activity since a number of challenges must be overcome: the limitations of an embedded system (e.g. limited battery, processing power and memory); the concurrency for the use of shared resources such as sensors and actuators and, at a high level, the challenge of making the robot perceive its current state and then choose actions to be performed in order to reach a goal. To help developers program mobile robots several *robot development frameworks* are available such as Carmen [Montemerlo et al. 2003], MOOS [Newman 2006] and ROS [Quigley et al. 2009]. These frameworks are able to handle sensors and actuators, but are limited when the use of *autonomous behaviour* is needed. One of the most common models for developing autonomous agents is based on the theory of human practical reasoning developed by Michael Bratman [Bratman 1987]. This computational model is called Beliefs, Desires and Intentions (BDI) Architecture and provides a computational analogy for the human practical reasoning characteristics.Several *agent programming languages (APLs)* exist that allow BDI agents to be programmed, such as AgentSpeak(L) [Rao 1996], Jason [Bordini et al. 2007] and 3APL [Hindrijs et al. 1999]. Through the use of these languages, we are able to simulate beliefs, desires and intentions at the software level, creating agents that have some characteristics, such as perception of the environment, knowledge acquisition and autonomous decision.

We aim to use agent programming languages architecture and their programming constructs to develop mobile robots using a higher-level abstraction. There are two advantages for choosing APLs to program mobile robots: first, the development process of any

software using a higher-level abstraction is simpler and has better maintainability compared to the use of a low-level programming language; second, these languages follow the BDI model, a model that tries to reproduce the human reasoning process as proposed by Bratman [Bratman 1987]. Other researchers have also studied the integration of robot programming constructs into agent programming languages, such as Ziafati [Ziafati 2013] [Ziafati et al. 2013], Verbeek [Verbeek 2003] and Wei [Wei and Hindriks 2013] and there is still work to be done in this area. This paper is a proposal for the integration of ROS and Jason using Cartago and the resulting programming environment aims to facilitate the modeling of complex behaviors on robots using the abstraction of autonomous cognitive agents.

This paper is organized as follows: In Section 2, we present a brief description of ROS, Jason, Cartago and JaCa Android. Section 3 presents the reason why we choose ROS and Jason, and a general description of the integration between them. In Section 4, we present some related work and finally, we conclude the paper in Section 5.

2. Background

2.1. ROS

The Robot Operating System (ROS) [Quigley et al. 2009] is a framework that aims to help software developers create applications for robots. It is composed of nodes, messages, topics, and services: *nodes* are software modules, namely processes that perform computation; *messages* are the data changed by nodes; *topics* are responsible to handle the communication between nodes: a node that is interested in a certain kind of data subscribes to the appropriate topic and a node expose its own data outside itself by publishing it to a given topic; and *services* are an alternative communication method in ROS, used for synchronous transactions.

2.2. Jason, Cartago and JaCa Android

Jason [Bordini et al. 2007] is an implementation of AgentSpeak(L) in Java. AgentSpeak(L) [Rao 1996] is a logical programming language for the implementation of BDI agents based on a restricted first-order language with events and actions. It can be viewed as a simplified textual language of Procedural Reasoning System (PRS) [Rao et al. 1992], one of the first implemented systems to be based on a BDI architecture and created to support real-time malfunction-handling and diagnostic systems [Weiss 1999]. PRS is implemented in LISP and provides goal-oriented behaviour and reactive behaviour.

CArtAgO (Common ARTifact infrastructure for AGents Open environments) [Ricci et al. 2009] is a framework used for engineering multi-agent applications, providing designer with the artifact notion and all the related concepts, such as object-oriented and service-oriented abstractions, autonomous activities (typically goal / task oriented) and structures (typically passive and reactive entities which are constructed, shared and used by the agents) [Ricci et al. 2009]. In CArtAgO, the sensors and actuators are structures that agents can create and use to partition and to control the information flow perceived from artifacts.

Artifacts are runtime objects that provide some kind of function or service which agents can use in order to achieve their objectives. Artifacts have a function description that can be read by the agents in order to discover the characteristics of a given artifact

and a set of operating instructions. Therefore, agents are able to interact with the artifacts by invoking operations and then observing the events generated from them. Namely, CArtAgO provide a natural way to model object-oriented and service-oriented abstractions (objects, components, services) at the agent level of abstraction [Ricci et al. 2009] through the use of artifacts.

JaCa is an agent-oriented programming platform comprising the integration of Jason and CArtAgO. JaCa uses Jason as programming language to develop and execute the agents and CArtAgO as the framework to develop and execute the environments [Santi et al. 2010]. In order to make JaCa language support mobile computing, a porting platform was developed on top of Android, where a mobile application can be realised as a workspace in which Jason agents are used to encapsulate the logic and the control of mobile application tasks, and artifacts are used as tools for agents to exploit available Android components and services [Santi et al. 2010]. Namely, JaCa Android is the porting of JaCa on Android, extended with a predefined set of artifacts specifically designed for exploiting Android functionalities [Santi et al. 2010]. According to Santi [Santi et al. 2010], the motivation to use JaCa on Android is address issues such as concurrency, asynchronous interactions with different kinds of services, and mostly, handle behaviours governed by specific context information (a typical characteristic of mobile applications).

3. A Proposal for the Integration of Jason and ROS

We decided for the use of ROS in this paper because of some important characteristics, such as: it is free and open source, thus we are able to change from the core code of ROS to implementations made by other researchers; it is very used by the academic community to program mobile robots, allowing us to have a large variety of tools already implemented and a good technical support; it supports cross-language development, allowing us to use Java (needed by Jason) with C (the ROS native language) and it has support for a great number of sensors and actuators. We use Jason to take advantage of some characteristics, such as: easily extensibility using Java; support for all the most important elements of the BDI architecture; support of strong negation, making available both closed-world and open-world assumptions and support for inclusion of additional information in beliefs and plans, allowing the creation of elaborated selection functions. Jason supports the creation of new internal actions, allowing developers to extend current Jason functionalities. We aim to use this feature in order to extend the agent internal capabilities and allow the agent to use legacy code.

In this paper, we propose the integration between Jason and ROS by creating a model that connects Jason primitives and ROS topics, extending the design of CArtAgO artifacts to support ROS features and behaviours, in order to program autonomous robots using a higher-level abstraction. This connection between the ROS low-level with the Jason high-level must be transparent to the mobile robot developer and the use of artifacts will help the achievement of this goal. The integration should support signatures and updates on the ROS topics and the conversion of the data from the ROS topics to a data understood by Jason (reciprocally, Jason's data must be converted to a data understood by ROS). Namely, we aim at expanding Jason agent programming language beyond the development of multi-agent systems, improving it to a high level mobile robot programming language. Inspired on JaCa Android proposal [Santi et al. 2010], which integrated

JaCa with Android, extending the agent language with a predefined set of artifacts specifically designed for exploiting Android functionalities, we aim to use Jason and CArtAgO to help the process of modeling the intermediate abstraction layer responsible to translate the Jason abstraction into ROS low-level robot control directives.

3.1. Proof of Concept

In order to verify the technical feasibility of implementing our proposed integration, an initial model was designed, with a basic set of primitives in Jason and two ROS topics. This proof of concept is a simple API, divided in two layers: the Java layer and the Jason layer, as shown in Figure 1. *Java layer* is basically composed by two modules: perceive and act. The perceive module is responsible for subscribing on the correct ROS sensor topics and to collect information from the sensors in order to forward this information as beliefs to Jason. The act module is responsible for publishing information from Jason into the ROS actuator topics. *Jason layer* is composed of a set of sensor primitives and a set of actuator primitives. The sensor primitives are connected to odom (nav_msgs/Odometry) topic and base_scan (sensor_msgs/LaserScan) topic, and the actuators primitives are connected just with odom ROS topic. Finally, a simple environment was created using Stage [Vaughan et al. 2003], composed by a robot free to "walk" on a hall and able to avoid obstacles (walls).

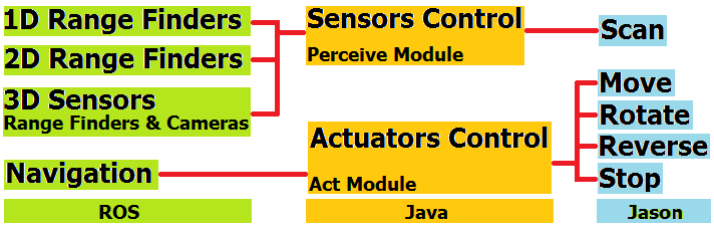


Figure 1. The ROS layer, the Java layer and the Jason layer (our initial API)

3.2. A proposal for the Integration Architecture

For a better understanding of how the integration of Jason and ROS works through the use of CArtAgO, we first briefly introduce how a common artifact is created. Listing 1 shows the Java pseudo-code of a CArtAgO artifact. We start importing the CArtAgO libraries (Line 1) and our artifact needs to extend Artifact class (Line 2). Inside the method init (Line 3), we need to define a property to be observable with the use of the artifact. The code continues with the definition of the artifacts operations (Line 6 and above). Details about how the operations works is beyond the scope of this paper.

For the purpose of modeling the intermediate abstraction layer between Jason and ROS, we propose to change the way in which the artifacts are created in order to make it supports ROS communication mechanisms. We propose the development of a new class called RosArtifact, that extends Artifact class from CArtAgO, adding features specifically designed for exploiting ROS robot sensors and control directives. Through the use of this class, the artifacts will be able to publish and to subscribe on ROS topics, thus the Jason agents are able to interact with ROS through the use of them. JaCa Android proposal [Santi et al. 2010] works in the same way: extending the Artifact class from CArtAgO

```
1  import cartago.*;
2  public class ClassName extends Artifact {
3   void init(){
4     defineObsProperty(...)
5     ...
6    }
7    @OPERATION void operationName(){
8    ...
9    }
10   ...
11 }
```

Listing 1: CArtAgO artifact pseudo-code sample.

and using it to define the artifacts that support control and observation of Android view interface.

We are developing a number of ROS artifacts, including: Movement, BaseIRScan and LaserScan, and we use artifact Movement as an example throughout this paper[1] Listing 2 shows the Java pseudo-code of a ROS artifact. We import ROS libraries (Line 2) and we create the ROS artifact extending RosArtifact class (Line 4). We use the method `subscribeRos()` from the mother class to subscribe the artifact on the ROS topic. A pseudo-code of `subscribeRos()` is shown on Listing 3. The sample artifact controls the movement of a robot, so we need to create the variables that will save linear and angular data to subscribe on the ROS topic: `linearData` (Line 8) and `angularData` (Line 9). Linear (Line 14), angular and odometer operations need to be created to control the data flow and to publish on ROS topic.

```
1  import cartago.*;
2  import ros.*;
3
4  RosClass.OdomData odom = new RosClass.OdomData(); //this will hold odometry data from ROS
5
6  public class Movement extends RosArtifact {
7    void init(string agentName, string rosTopic, string rosType){
8      subscribeRos(agentName, string rosTopic, string rosType); //this will subscribe on ROS topic
9      defineObsProperty("odom", odom);
10   }
11 @OPERATION
12 void linear(string direction, string velocity) { //handle linear odometry info from ROS topic
13   if ( direction != null && velocity != null ) {
14     RosClass.geometry_msgs.Twist twist = RosClass.publisherCmdVel.newMessage();
15     if (direction == "X"){ twist.getLinear().setX(velocity); }
16     ... // do the same for Y and Z
17     }
18   }
19 ... //TODO: angular and odometer operations
20 }
```

Listing 2: ROS artifact pseudo-code sample.

Listing 3 is the proposal of a pseudo-code of how the RosArtifact class subscribe on a ROS topic (Line 3). On this example, we are connecting on a IR sensor. We add a listener on the ROS topic thus any information from the sensor can be handled by our class (Line 4). All the details about how this class connects on a ROS topic is beyond the scope of this paper. In our proposal, the RosArtifact class should controls all the data flow between Jason and ROS. This is needed to avoid the classic bounded-buffer problem if ROS produce much more data than Jason could consume.

[1] Space limitations prevented us from detailing the other artifacts.

```
1  public void SubscribeRos(String agentName, String rosTopic, String rosType) {
2    ...
3    subscriberScan = m_rosnode.getConnectedNode().newSubscriber("/"+agentName+rosTopic, rosType+"._TYPE"
         );
4    subscriberScan.addMessageListener(new MessageListener<sensor_msgs.LaserScan>() {
5      @Override
6      public void onNewMessage(sensor_msgs.LaserScan message) {
7        m_scan = message; //holds odometry data
8      }
9    });
10   ...
11 }
```

Listing 3: ROS subscription on a sensor pseudo-code sample.

4. Related Work

Other researchers have investigated the integration of robot development frameworks and APLs. Ziafati et al [Ziafati et al. 2013] presents four requirements for BDI-based agent programming languages to facilitate the implementation of autonomous robot control systems; Ziafati [Ziafati 2013] identified and addressed BDI-based agent programming languages requirements for autonomous robot programming. The author developed an environment interface for 2APL to facilitate its integration with ROS; Verbeek [Verbeek 2003] research extends the 3APL language for high level robot control by creating a communication interface between the 3APL and ROS. He conducted experiments in a simulated and in a physical environment; and Wei [Wei and Hindriks 2013] presents a cognitive robot control architecture using GOAL [Hindriks 2009] and URBI [Baillie 2005].

5. Conclusion

In this paper, we propose the integration of an agent programming language and a robot development framework namely, the Jason implementation of AgentSpeak(L) and ROS. This integration of ROS and Jason using CArtAgO is a work in progress and is currently being implemented. We investigated the technical feasibility of implementing our proposed intermediate layer between Jason and ROS through a proof of concept implementation (presented in Section 3.1). In our proof of concept, communication between Jason and ROS is achieved using a ROS node extending Jason AgArch class and using ROS Java libraries. Our current work is on the codification of the complete design, which composes not only Jason and ROS, but the use of CArtAgO to reach our proposal objectives: to facilitate the modeling of complex behaviors on robots using the abstraction of autonomous cognitive agents.

Acknowledgements .
The author would like to acknowledge the guidance provided in this paper to Felipe Meneguzzi who supervises his MSc, and provided invaluable advice in the writing of this paper. I would like to thank the anonymous WESAAC reviewers for their valuable and constructive suggestions.

References

Baillie, J.-C. (2005). Urbi: Towards a universal robotic low-level programming language. In *On 2005 IEEE RSJ International Conference on Intelligent Robots and Systems*, pages 820–825. IEEE.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. Wiley. com.

Bratman, M. E. (1987). Intention, plans, and practical reason.

Hindrijs, K. V., de Boer, F. S., van der Hoek, W., and Meyer, J.-J. C. (1999). Agent programming in 3apl. In *Autonomous Agents and Multi-Agent Systems*.

Hindriks, K. V. (2009). Programming rational agents in goal. In *Multi-Agent Programming: Languages, Tools and Applications*, pages 119–157. Springer.

Montemerlo, D., Roy, N., and Thrun, S. (2003). Perspectives on standardization in mobile robot programming: the carnegie mellon navigation (carmen) toolkit. *Intelligent Robots and Systems, 2003. (IROS 2003)*, 3:2436 – 2441.

Newman, P. M. (2006). Moos - mission orientated operating suite. Department of Engineering Science Oxford University.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3.

Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world*, MAAMAW '96, Secaucus, NJ, USA. Springer-Verlag New York.

Rao, A. S., Georgeff, M., and Ingrand, F. (1992). An architecture for real-time reasoning and system control. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world: agents breaking away*, pages 34–44.

Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009). Environment programming in cartago. In *Multi-Agent Programming: Languages, Tools and Applications*, pages 259–288. Springer US.

Santi, A., Guidi, M., and Ricci, A. (2010). Jaca-android: An agent-based platform for building smart mobile applications. *Third International Workshop, LADS 2010*.

Vaughan, R. T., Gerkey, B. P., and Howard, A. (2003). On device abstractions for portable, reusable robot code. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2121 – 2427.

Verbeek, M. (2003). 3apl as programming language for cognitive robots. Master's thesis, Utrecht University.

Wei, C. and Hindriks, K. V. (2013). An agent-based cognitive robot architecture. In *Programming Multi-Agent Systems*, pages 54–71. Springer.

Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Modern Approach to Artificial Intelligence*. Massachusetts Institute of Technology.

Ziafati, P. (AAMAS 2013). Programming autonomous robots using agent programming languages. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*.

Ziafati, P., Dastani, M., Meyer, J.-J., and van der Torre, L. (2013). Agent programming languages requirements for programming autonomous robots. In *Programming Multi-Agent Systems*, pages 35–53. Springer.

Métodos para Modelagem de Sistema Abertos: Uma Análise Inicial

Daniela Maria Uez, Jomi F. Hübner

[1]Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

{dmuez,jomi}@das.ufsc.br

Resumo. *Sistemas multiagentes abertos possuem características específicas que precisam ser levadas em conta durante as fases de projeto, análise e implementação. Diante disso, é importante que os métodos estejam preparados para lidar com essas características. Esse artigo apresenta uma investigação inicial dos métodos da área de AOSE com o objetivo de verificar como estes lidam com essas características inerentes aos sistemas multiagentes abertos.*

Abstract. *To design open multi-agent systems it is necessary to consider some special issues that are not considered in other multi-agent systems. Therefore, AOSE method should be prepared to deal with these issues. In this paper we present an initial research to investigate how AOSE methods deal with open systems specific issues.*

1. Introdução

Diversos métodos para projeto e análise de sistemas multiagentes (SMA) foram propostos nos últimos anos. Muitos foram desenvolvidos para um único projeto e deixaram de ser usados ao longo do tempo, enquanto outros têm sido refinados e testados. Apesar disso, nenhum desses métodos pode ser considerado um padrão, já que SMA são utilizados em diversos domínios e, para cada um, diferentes características devem ser observadas [Wooldridge 2002].

Um tipo de SMA que tem sido muito estudado são os sistemas ditos abertos, aqueles nos quais os agentes podem entrar e sair livremente. Ao contrário dos SMA tradicionais, durante o desenvolvimento dos sistemas abertos não se sabe quantos nem quais agentes terão parte no sistema, o que faz com que os agentes possam apresentar diferentes arquiteturas, diferentes graus de deliberação, diferentes tipos de conhecimento e também comportamentos maliciosos, podendo prejudicar os outros agentes e o próprio sistema [Dignum et al. 2008, Eijk et al. 2000].

Portanto, ao contrário de outros sistemas, SMA abertos não podem ser pensados como sendo a soma do comportamento de diversos agentes, simplesmente porque não se tem conhecimento prévio do comportamento que esses agentes apresentarão. Para que se possa projetar e analisar esses sistemas, é importante que os métodos orientados a agentes permitam o projeto e a análise das outras dimensões do SMA, como a organização, o ambiente e a interação, sem que essas estejam ligadas a um tipo de agente predefinido.

Nesse artigo é apresentada uma breve análise da área de engenharia de software orientada a agentes (AOSE) com o objetivo de verificar se os métodos existentes permitem

o projeto e a análise de SMA abertos. Essa análise é feita no Seção 2 e após, na Seção 3, são apresentadas algumas considerações finais sobre a análise e algumas diretrizes para futuros trabalhos.

2. Engenharia de software para sistemas abertos

Métodos tradicionais da área de AOSE são fortemente focados no agente, ou seja, o sistema é definido com base nos processos mentais dos agentes. Por exemplo, no método Prometheus [Padgham and Winikoff 2004], protocolos de interação são definidos como uma sequência de mensagens trocadas entre dois agentes. As interações são especificadas para ocorrer com agentes pré-determinados, já que as mensagens são explicitamente definidas nos planos desses agentes. Essa abordagem não leva em conta a possibilidade de novos agentes serem inseridos no sistema, nem mesmo se preocupa com questões relacionadas à confiabilidade dos agentes nas interações, já que todos os agentes são definidos durante o processo de análise do sistema. O método permite a definição de papéis que são concebidos a partir de um conjunto de objetivos e não têm exatamente uma função organizacional. Durante a fase de Projeto Arquitetural, esses papéis são agrupados e dão origem aos agentes que farão parte do sistema e, na fase de implementação, os agentes foram definidos e seu comportamento é fixo. Esses métodos também não permitem a definição dos recursos que estão disponíveis no ambiente onde os agentes se encontram, mas permitem que se especifique as ações que os agentes podem executar no ambiente. Além do Prometheus, os métodos Tropos [Bresciani et al. 2004] e Adelfe [Bernon et al. 2002] também utilizam essa abordagem.

Outros métodos têm incorporado a questão da organização como entidade de primeira classe na modelagem de SMA. Por exemplo, o método O-MaSe [DeLoach and García-Ojeda 2010] permite uma definição mais clara dos aspectos organizacionais: são definidos papéis organizacionais e com base nesses papéis são definidas as interações possíveis entre os agentes. O mesmo ocorre no método Ingenias [Gómez-Sanz et al. 2008]. Porém, o Ingenias não permite que sejam definidas normas para o SMA, enquanto o O-MaSe permite a definição de políticas que são associadas aos papéis ou a agentes específicos. As normas são importantes pois regulam o comportamento dos agentes, auxiliando o sistema a atingir seus objetivos globais. Ambos os métodos permitem definir os tipos de agentes que farão parte do sistema. Esses agentes são definidos para desempenhar um papel específico. Além disso, nenhum desses métodos leva em conta os aspectos relacionados a modelagem do ambiente.

[Dastani et al. 2004] apresenta um método baseado no modelo organizacional OperA+Environment [Dastani et al. 2003]. Esse trabalho, apesar de usar abstrações do modelo OperA para modelagem dos conceitos organizacionais, ao final os papéis modelados são agrupados para dar origem aos tipos de agentes que efetivamente farão parte do sistema. Ao final, portanto, tem-se um conjunto de agentes que podem desempenhar os papéis definidos na fase de análise. Além disso, as interações também são especificadas com base nos tipos de agentes definidos e os recursos do ambiente são definidos tendo em vista sua utilização pelos agentes.

O método Prometheus AEOlus [Uez et al. 2013] foi desenvolvido com base no método Prometheus e inclui a modelagem de aspectos organizacionais e do ambiente. Os aspectos organizacionais foram definidos levando-se em conta os conceitos utiliza-

Método	Análise	Projeto	Implementação
Prometheus Tropos Adelfe	AEO	AEI	A
Ingenias O-MaSe	AIO	AIO	A
Opera+Environment	AEIO	AEIO	A
Prometheus AEOlus	AEO	AEO	AEO

Table 1. Fases de desenvolvimento dos métodos e quais dimensões do SMA cada uma trata: agente (A), ambiente (E), interação (I) ou organização (O)

dos pelo modelo organizacional Moise [Hübner et al. 2010]. Esse método permite a definição dos aspectos organizacionais, como papéis e grupos, e também permite que se defina como os papéis se relacionam entre si. O Prometheus AEOlus também permite a especificação do ambiente: os artefatos que fazem parte desse ambiente, as operações que são disponibilizadas para os agentes por cada artefato e as propriedades observáveis de cada artefato que são percebidas pelos agentes. O Prometheus AEOlus também permite a definição dos tipos de agentes que farão parte do sistema e quais papéis cada tipo de agente está preparado para assumir. Porém, a especificação da organização não é substituída pela definição dos agentes. Dessa forma, na fase de geração de código, as três dimensões (organização, ambiente e agentes) foram projetadas e analisadas usando abstrações próprias, podendo ser implementadas através de linguagens específicas. Por outro lado, o Prometheus AEOlus não permite a definição da interação, que é feita da mesma forma que no método Prometheus, através das mensagens definidas nos planos dos agentes.

Essa análise pode ser resumida com base em conta em quais dimensões cada fase do método atua, conforme apresentado na Tabela 1. Nos métodos Prometheus, Tropos e Adelfe, durante a fase de análise são levados em conta conceitos das dimensões de agente (A), ambiente (E) e organização (O). Na fase de Projeto, os conceitos organizacionais são substituídos por conceitos de agente (A) e também são definidas as interações entre os agentes (I). Porém, na fase de implementação somente são utilizados conceitos de agente. O mesmo ocorre com os métodos Ingenias e O-MaSe, nos quais as fases de análise e projeto são usados conceitos de agente, interação e organização. E também ocorre no método OperA+Environment, no qual conceitos de agente, ambiente, interação e organização são usados nas fases de análise e projeto. Nesses casos tem-se a análise e o projeto de conceitos que não são utilizados durante a implementação. No caso do método Prometheus AEOlus, este utiliza conceitos das dimensões de agente, ambiente e organização tanto durante as fases de análise e projeto, quanto na fase de implementação. Porém esse método não inclui a dimensão de interação como entidade de primeira classe em nenhuma das fases (as interações entre agentes são definidas diretamente no agente).

3. Considerações Finais

O objetivo principal desse artigo é investigar se os métodos da área de AOSE permitem o projeto e análise de sistemas abertos. Devido a sua natureza heterogênea e dinâmica, as dimensões de organização, ambiente e interação de SMA abertos precisam ser pensadas sem que seja necessário saber como serão os agentes que farão parte desse sistema. Apesar dessa investigação preliminar ter levado em conta alguns poucos métodos, dois problemas para modelagem de SMA abertos podem ser destacados:

- Alguns métodos que afirmam ser adaptados para trabalhar com sistemas abertos se preocupam com a modelagem da organização como entidade de primeira classe e não permitem que outras dimensões do SMA, principalmente o ambiente e a interação, também sejam modeladas como entidades de primeira classe.
- Alguns métodos, mesmo que as dimensões sejam analisadas como entidades de primeira classe, acabam por reduzir os conceitos específicos em conceitos de agentes nas fases seguintes, seja durante projeto ou na implementação do SMA. Por exemplo, o sistema é pensado do ponto de vista organizacional, usando abstrações organizacionais, mas acaba sendo implementado como um conjunto de agentes, usando abstrações relativas ao agente.

Em ambos os casos, não os métodos não permitem efetivamente a análise, o projeto e a implementação de SMA abertos. Obviamente, para que essa investigação seja mais precisa, ela deve ser expandida para cobrir outros métodos da área. Com os resultados futuros dessa análise, pretende-se apresentar um método que permita a análise e o projeto de SMA abertos através de melhorias no método Prometheus AEOlus.

References

Bernon, C., Gleizes, M. P., Peyruqueou, S., and Picard, G. (2002). Adelfe: A methodology for adaptive multi-agent systems engineering. In Petta, P., Tolksdorf, R., and Zambonelli, F., editors, *ESAW*, volume 2577 of *Lecture Notes in Computer Science*, pages 156–169. Springer.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). TROPOS: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

Dastani, M., Dignum, V., and Dignum, F. (2003). Role-assignment in open agent societies. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 489–496, New York, NY, USA. ACM.

Dastani, M., Hulstijn, J., Dignum, F., and Meyer, J.-J. C. (2004). Issues in multiagent system development. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 922–929, Washington, DC, USA. IEEE Computer Society.

DeLoach, S. A. and García-Ojeda, J. C. (2010). O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems. *IJAOSE*, 4(3):244–280.

Dignum, F., Dignum, V., Thangarajah, J., Padgham, L., and Winikoff, M. (2008). Open agent systems ??? In Luck, M. and Padgham, L., editors, *Agent-Oriented Software Engineering VIII*, volume 4951 of *Lecture Notes in Computer Science*, pages 73–87. Springer Berlin Heidelberg.

Eijk, R., Boer, F., Hoek, W., and Meyer, J.-J. (2000). Open multi-agent systems: Agent communication and integration. In Jennings, N. and Lespérance, Y., editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *Lecture Notes in Computer Science*, pages 218–232. Springer Berlin Heidelberg.

Gómez-Sanz, J. J., Fuentes, R., Pavón, J., and García-Magariño, I. (2008). Ingenias development kit: a visual multi-agent system development environment. In *AAMAS (Demos)*, pages 1675–1676. IFAAMAS.

Hübner, J. F., Boissier, O., Kitio, R., and Ricci, A. (2010). Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, 20(3):369–400.

Padgham, L. and Winikoff, M. (2004). *Developing Intelligent Agent Systems: A Practical Guide*. Halsted Press, New York, NY, USA.

Uez, D. M., Hübner, J. F., and Webber, C. (2013). Método para modelagem de agentes, ambiente e organização de sistemas multiagentes. In *Autosoft 2013 - IV workshop sobre sistemas de software autônomos*, pages 41–50.

Wooldridge, M. (2002). Intelligent agents: The key concepts. In *Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications II-Selected Revised Papers*, pages 3–43, London, UK, UK. Springer-Verlag.

SAAPIENS: UMA FERRAMENTA DE AUTORIA DE OBJETOS DE APRENDIZAGEM E APOIO PEDAGÓGICO NA DEDUÇÃO NATURAL NA LÓGICA PROPOSICIONAL

Agnaldo M. Rodrigues[1], João Carlos Gluz[1]

[1]Programa Interdisciplinar de Pós Graduação em Computação Aplicada (PIPCA) – Universidade do Vale do Rio dos Sinos (UNISINOS) – Caixa Postal 275 – 93.022-000 – São Leopoldo –RS

{agnaldo@sapucaia.ifsul.edu.br, jcgluz@unisinos.br}

ABSTRACT: *The need of tools to help teachers, which are able to assist them in planning, implementation and monitoring of the teaching-learning process, is an important necessity in the educational area. In some cases, difficulties to access to these technologies, coupled with the lack of knowledge and understanding on the part of teachers, undermines the way in which educational contents could be developed and used. This is the main motivation behind the creation of SAAPIEnS authoring and monitoring. The tool has a multiagent system architecture built over the Heraclito system. It makes possible for teachers to create customizable learning objects for the learning domain of natural deduction in propositional logic. It also helps teachers to check and monitor how much the authored learning objects are contributing to the improvement and progress of learners.*

1. Introdução

A falta de ferramentas de autoria e acompanhamento de uso de conteúdos digitais, em particular para materiais em formato de Objetos de Aprendizagem (OA), que sejam adequadamente projetadas ao uso dos professores em ambientes digitais de ensino (sejam ambientes Web ou não), potencialmente exige um maior conhecimento técnico para a criação destes conteúdos, o que não é razoável.

Esta exigência, evidentemente, pode gerar barreiras no uso desse tipo de material digital por parte dos professores. Uma pressuposição primordial do presente trabalho é que professores podem e devem se tornar autores de conteúdos digitais instrucionais, mas sem a necessidade de habilidades de programação ou treinamento técnico extensivo na área de Tecnologias de Informação e Comunicação (TIC). Também se espera que os professores possam ter acesso às informações sobre o uso destes objetos, tais como registros nos bancos de dados dos ambientes de ensino, sem necessitar de conhecimentos técnicos aprofundados. Neste contexto, o presente trabalho buscou centrar sua pesquisa sobre ferramentas de apoio a autoria e acompanhamento de conteúdos digitais baseadas em infraestruturas previamente existentes. A infraestrutura escolhida foi o sistema de tutoria *Heráclito* [PENTEADO e GLUZ, 2011], que é voltada ao ensino de Lógica Proposicional através da combinação das tecnologias de agentes inteligentes e OA. Este sistema vem sendo aplicado em processos de ensino real em sala de aula, obtendo resultados significativos [PENTEADO, 2012].

O objetivo final do presente trabalho é desenvolver um conjunto de ferramentas de apoio à autoria e ao acompanhamento do uso dos OA *Heráclito* (OAH) complementando o cenário de suporte ao professor. Assim sendo, esse artigo apresenta a arquitetura e a proposta da criação do sistema *SAAPIEnS* (Sistema de Autoria e

AcomPanhamento Inteligente de Ensino Superior - módulo compatível com OA Heráclito) que tem como objetivo fornecer suporte ao professor através de uma ferramenta de autoria que facilite a criação, manutenção e reutilização dos OAH e que possibilite também, as análises e o acompanhamento contínuo do uso destes objetos pelos alunos. A junção futura destas e de outras ferramentas servirão de base para a plataforma MILOS [GLUZ e VICARI, 2010] de apoio ao uso de OA compatíveis com a proposta de metadados OBAA [Bez et al., 2010].

2. Arquitetura

O Sistema *SAAPIEnS* é dividido em três subsistemas principais (Fig. 1): *Autoria*, responsável pela criação e edição dos OAH; *Acompanhamento*, responsável pela coleta , tratamento dos dados, controle das atividades e a geração de análises; *Gerência*, responsável pelo controle, parametrização e manutenção da ferramenta.

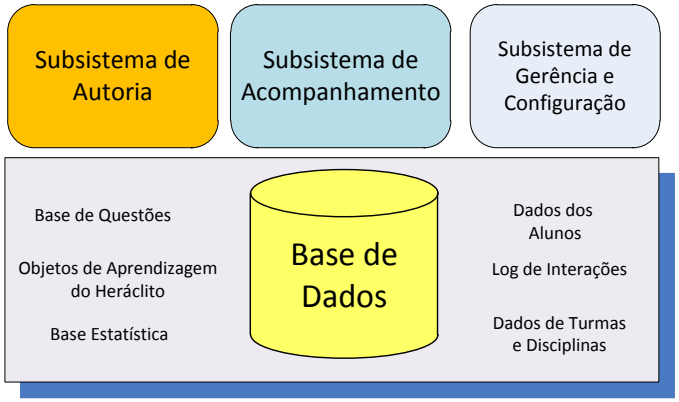


Figura 1. Arquitetura do Sistema SAAPIEnS.

3. Subsistemas

O subsistema de *Autoria* implementa a ferramenta principal. É através dele que se dá o processo da concepção dos OAH e a disponibilização dos seus recursos. Entre algumas das facilidades que o sistema apresenta é a possibilidade de ser acessado via Web, com suporte a conteúdos educacionais multimídia, ativos e com tutoria inteligente.

O processo de criação de um OAH se inicia pela definição dos atributos (parâmetros) básicos do OAH, seguida da definição do conjunto de atividades didático-pedagógicas de acordo com os objetivos de ensino a serem alcançados pelo uso do OA, ou seja, qual o nível de conhecimento que se espera que os alunos atinjam. Essas atividades poderão ser posteriormente alteradas e customizadas.

Além dos atributos básicos de identificação do OAH, deverão ser definidos os tipos de atividades que farão parte do objeto. Estas atividades são divididas em duas categorias distintas de materiais, o primeiro relacionado aos exercícios de lógica, destinado à prática dos alunos e o segundo, para materiais expositivos como apoio às atividades:

- Exercício de Lógica: Argumento de Dedução Natural da Lógica Proposicional (DNLP), Tabela-Verdade e Avaliação de Fórmulas;
- Material Expositivo: Textos, Imagens, Arquivos (PDF, DOC...), Material de Apresentação (arquivo ppt), Animações e outros;

Todo conjunto de atividades do OAH deve ser categorizado e classificado, podendo ser dividido em quatro níveis: 0 - Fundamental, 1-Básico, 2-Intermediário e 3-Avançado de acordo com critério estabelecido pelo professor;

Uma vez definidos os parâmetros básicos e as atividades do OAH, é gerada uma identificação do objeto (ID) no *SAAPIEnS*. Após essa geração o professor pode customizar o perfil de ajuda para o OAH, configurando funcionalidades de tutoria automática do *Heráclito,* incluindo não só a habilitação da tutoria, mas também como dará esta tutoria de acordo com o nível de classificação das atividades do OAH. Os recursos de tutoria fornecidos ao aluno são divididos em dois níveis de suporte: o primeiro nível de suporte fornece ajuda mais genérica que pode ser a indicação um material extra para o aluno, uma explicação de provas através de exemplos ou um tutorial em vídeo. No segundo nível de suporte o sistema fornece auxílios pontuais, como sugestão de regras de inferência a serem usadas na continuação do exercício e fornecimento de avisos quando uma regra está sendo usada de forma incorreta ou redundante. Neste caso o parâmetro *Dicas* indica se o *Heráclito* fornecerá o próximo passo da solução, enquanto que *PróximoPasso* que habilita *Feedback* passo-a-passo no processo de resolução.

Através do subsistema de *Acompanhamento* o professor tem acesso a um conjunto de análises estatísticas, podendo verificar se a configuração proposta do OAH está realmente de acordo como o nível de aprendizagem desejado. Dessa forma, o professor pode fazer os ajustes necessários ao detectar falhas, dificuldades ou facilidades encontradas por parte dos alunos.

O subsistema de *Acompanhamento* é responsável pela coleta dos dados e das atividades feitas pelos alunos e pela geração das informações estatísticas baseadas nas análises destes dados. Para tanto, este subsistema é capaz de monitorar as interações entre os alunos e as atividades de resolução de exercícios contidos nos OAH. São os registros dessas interações que fornecerão a base para as análises e controles do subsistema de acompanhamento.

O subsistema de *Gerência* é o responsável pelas ações de controle e parametrização da ferramenta. Suas principais funcionalidades são: segurança dos dados, manutenção das contas e cadastros gerais, parametrização e habilitação de recursos da ferramenta, importação e exportação de arquivos e manter a consistência da base de dados.

4. Agentes de Software

Em termos de arquitetura o *SAAPIEnS* é formado por dois agentes de *software* [WOOLDRIDGE, 2009] [BORDINI et al, 2007]: o agente *Monitor* e o agente *Analyzer*.

O agente *Monitor* tem o papel de monitorar as interações dos alunos com os OAH e atualizar as bases de dados com os registros dessas monitorações. As ações realizadas pelo aluno durante a execução das atividades são gravadas no registro (*log*) de atividades que servirão para as análises posteriores. Entre as ações monitoradas pelo agente *Monitor* estão: (a) o tempo total que o aluno está conectado realizando atividade, (b) o tempo dispendido para realizar uma determinada prova seja ela parcial ou total (define-se prova parcial, àquela em que o aluno não chegou ao resultado final, porém deu início a sua resolução), (c) o tempo de inatividade do aluno, (d) o número de passos que o aluno usou para realizar determinada prova, (e) uso de ajuda do tutor solicitado pelo aluno e (f) o uso de dicas que o aluno teve para determinada solução. O agente

Monitor também controla quais as regras utilizadas e a quantidade de vezes que foram utilizadas tanto corretas, quanto incorretamente na sua solução.

Um OAH pode operar em modo *stand-alone*, sem estar conectado aos servidores da plataforma. Neste caso o OAH mantém os históricos das atividades em um arquivo temporário, que é transferido para o agente *Monitor* no momento da próxima conexão à plataforma, de forma a manter atualizado o log de interações do SAAPIEnS.

O agente *Analyzer* é o responsável pelo pré-processamento dos dados e geração das análises estatísticas. A principal função do Agente *Analyzer* é gerar os resultados que servirão de subsídios ao professor, para que o mesmo possa fazer o acompanhamento do desempenho dos alunos, assim como desenvolver ações pedagógicas visando contribuir na melhoria do processo de ensino.

Os agentes do *SAAPIEnS* interagem com agentes do *Heráclito* na plataforma da MILOS, através de troca de mensagens e de recursos entre os mesmos. Na Figura 2, é possível visualizar o ambiente da plataforma e de interação entre os agentes do sistema. O agente *Mediador* é responsável pelo acompanhamento de todo o processo de tutoria, de acordo com a parametrização definida na autoria de OAH, o professor poderá habilitar o grau de granularidade de apoio do serviço de tutoria [VANLENHN,2006]. O agente *Especialista* em dedução natural é o responsável pela análise das provas realizadas pelos alunos podendo gerar automaticamente provas de dedução natural completa, mas também é capaz de finalizar provas parciais, ou seja, completar as provas que estão sendo desenvolvidas pelo aluno. O agente *Modelo do aluno* é responsável por receber as ações do aluno manter informações relacionadas ao aluno e fazer a troca de informações com o agente *Monitor*.

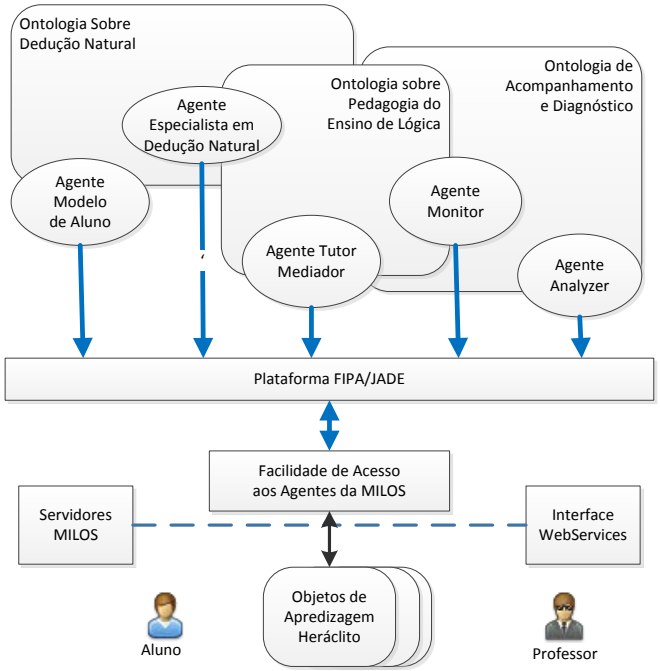


Figura 2: Integração da Plataforma SAAPIEnS-Heráclito

No caso dos OAH, o acesso aos agentes do sistema de apoio pedagógico é disponibilizado por meio de uma interface *WebService*. Os OAH implementam um cliente de serviços, que acessa o provedor de serviços implementado pelo componente Facilidade de Acesso aos Agentes da MILOS. Este provedor, por sua vez, implementa mecanismos de comunicação que convertem as requisições de serviços dos OAH que

podem então ser enviadas aos agentes MILOS por meio da plataforma de comunicação JADE [BELLIFEMINE, 2010].

5. Considerações Finais

O uso de tecnologias nos contextos educacionais tem contribuído de maneira substancial no processo de ensino-aprendizagem nos últimos anos. O crescimento da *web* como fonte de informações e a quantidade de recursos disponíveis tanto a alunos quanto para professores, fazem desse cenário um ambiente promissor para pesquisa e o desenvolvimento de novas ideias no âmbito da educação. Nesse contexto é preciso levar em consideração a importância que o professor representa dentro do cenário educacional, tanto como especialista em ensino, quanto como um usuário de ambientes digitais, sendo, portanto, imprescindível provê-lo de estruturas que contribuam para a melhoria do seu trabalho.

Espera-se que o conjunto de recursos de criação, acompanhamento e análise do uso de OA fornecido pelo *SAAPIEnS* permita aos professores avaliar o desempenho alcançado pelos alunos de uma maneira significativa, permitindo que se possa não só realizar ações corretivas, como também prever situações futuras e agir proativamente. O *SAAPIEnS* não foi concebido com objetivo de um ciclo único de aprendizagem. Na medida em que informações de acompanhamento são atualizadas e adicionadas à base, novas análises se tornam possíveis e novas conclusões podem ser percebidas. Espera-se que isso ajude o professor a estabelecer critérios no desenvolvimento de planos de ensinos, adequados à realidade do contexto dos seus alunos. As facilidades oferecidas pelo uso desta ferramenta facilitam também o desenvolvimento dos objetos de aprendizagem inteligentes customizados especificamente para um determinado fim, visando atingir determinado grupo de alunos, podendo assim contribuir de maneira qualitativa no progresso realizado pelos mesmos.

6. Agradecimentos

Os autores agradecem ao MCT/FINEP/MC/FUNTTEL, a CAPES e ao CNPq pelo financiamento dessa pesquisa.

Referências

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. Developing Multi-Agent Systems with JADE. John Wiley & Sons, 2007.

BEZ, M.; VICARI, R. M.; SILVA, J. M.; RIBEIRO, A.; GLUZ, J. C.; PASSERINO, L. M.; SANTOS, E.; PRIMO, T.; ROSSI, L.; BEHAR, P.; Filho, R.; ROESLER, V. Proposta Brasileira de Metadados para Objetos de Aprendizagem Baseados em Agentes (OBAA). RENOTE. 2010, v.8, p.1 - 10.

BORDINI, R., HÜBNER, J., WOOLDRIDGE, Michael. Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology, 2007.

MOSSMAN, M.; GOMES, L.; GLUZ, J. C. Objetos de Aprendizagem Móveis para Ensino de Dedução Natural na Lógica Proposicional. In: Anais do Simpósio Brasileiro de Informática na Educação. 2012.

PENTEADO, F.; GLUZ, J. C. Sistema Heráclito: Suporte a Objetos de Aprendizagem Interativos e Dialéticos Voltados ao Ensino de Dedução Natural na Lógica Proposicional. In: Anais do Simpósio Brasileiro de Informática na Educação. 2011.

PENTEADO, F. Agente pedagógico para mediação do processo de ensino aprendizagem da dedução natural na lógica. 2013. 127 f Dissertação (mestrado). Universidade do Vale do Rio dos Sinos, Pós-Graduação em Computação Aplicada, RS, 2013.

VANLEHN, K. The behavior of tutoring systems. International journal of artificial intelligence in education, v. 16, n. 3, p. 227-265, 2006.

VICCARI, R.; GLUZ, J; PASSERINO, L.; et al. The OBAA Proposal for Learning Objects Supported by Agents. Procs. Of MASEIE Workshop – AAMAS 2010, Toronto, Canada, 2010.

WOOLDRIDGE. M. An Introduction to MultiAgent Systems. 2nd ed. John Wiley & Sons, 2009.

Fragmenting ADELFE using the Medee Framework

Thomas Liu de Almeida, Sara Casare, Anarosa A. F. Brandão

Laboratório de Técnicas Inteligentes
Escola Politécnica – Universidade de São Paulo (USP)

{thomas.almeida, anarosa.brandao}@usp.br, sjcasare@uol.com

Abstract: *This work describes the fragmentation of the agent-oriented method ADELFE following the approach defined by the Medee Framework. The fragmentation goal is to increase the population of the Medee Method Framework repository. The Medee method framework supports the development of multiagent systems based on the Situational Method Engineering approach. To achieve our goal, the paper briefly explains the Medee Method Repository structure and its associate process for fragmenting methods. Then, it's described how this process was applied to define ADELFE fragments.*

1. Introduction

The multiagent paradigm is based on the interaction of autonomous elements in order of achieving goals in a given system. It is a fitting solution for various situations of informatics, specially nowadays when huge amount of information and fast changes on the virtual environment request some kind adaptation and application of artificial intelligence techniques. However, the lack of tools and frameworks, combined with complexity of systems based on this paradigm makes its adoption scarce in the software industry.

The Medee Method Framework [Casare et al, 2013] aims to provide this kind of support for developing multiagent systems. It allows the building of customized methods on demand according to the project specifications. This is based on Situational Method Engineering concepts, which is a section of Method Engineering [Brinkkemper, 1996] that focuses on these tailored solutions to each given situation. They are called situational methods.

In order to achieve that, the framework stores pieces of agent-oriented methodologies, such as Tropos [Giorgini et al, 2004], Gaia [Zambonelli et al, 2003] and PASSI [Cossentino, 2005], as well as organizational models, like MOISE+ [Hubner et al, 2002]. Each one of these pieces is called a fragment. Fragments are standardized and represent a coherent part of a method that can be used and re-used to solve entire problems or parts of them. This paper consists in the fragmentation of the agent-oriented method ADELFE [ADELFE, 2003] utilizing the process proposed in the Medee Framework.

2. Medee Method Framework

The Medee Method Framework aims to provide methods for developing organization centered MAS. It allows you the combination of advantages from AOSE methods and agent organizational models by reusing portions of these two types of MAS development approaches.

The Medee Method Framework is basically composed by the Medee Conceptual Model, the Medee Method Repository and the Medee Delivery Process [Casare et al, 2013].

2.1 Medee Method Repository

The architecture of the Medee Method Repository consists in three pillars: the Medee Elements, the Medee Fragments and the Medee Methods. The first pillar provides SPEM method elements [SPEM, 2008], that are: tasks, work products, roles and guidance. They will serve for elaborating Medee MAS method fragments stored in the second pillar, which in their turn provide fragments to compose methods stored in the third pillar. SPEM is a standard for describing methods and processes.

Method fragments in the second pillar pertain to one of the following layers of granularity: activity, phase, iteration or process. They are also standardized according to Medee MAS Method fragment definition, in order to make the reuse and composition of methods with fragments from different sources possible.

The Medee Methods pillar stores Medee Situational Methods and Medee AOSE Methods. The former is composed according to a given project situation, and the latter is formed by a set of fragments usually captured from on single source.

2.2. Medee Delivery Process

The Medee Delivery Process specifies how to populate the three pillars of the Medee Repository. It is composed of three phases: Method Elements Capture, Method Fragment Elaboration and Medee Method Composition, each one describing one pillar of the Medee Repository, from the first to the third, respectively. During the description of ADELFE fragmentation the phases will be outlined.

3. Fragmenting ADELFE using the Medee Delivery Process

3.1 ADELFE in a nutshell

The name ADELFE is the French acronym for "toolkit to develop software with emergent functionality" (Atelier pour le DEveloppement de Logiciels à Fonctionnalité Emergente) [Bernon et al, 2005]. It is a method based on object-oriented methods, which follows the Rational Unified Process (RUP) and uses UML and AUML notations [Bauer et al, 2001] . ADELFE aims to guide the development of Adaptative Multiagent Systems (AMAS). These systems are embedded in a dynamic environment, where agents are continuously searching and adapting themselves to keep cooperating in collective tasks. It is composed by four phases: preliminary requirements, final requirements, analysis and design. Figure 1 shows the ADELFE workflow detailing its phases.

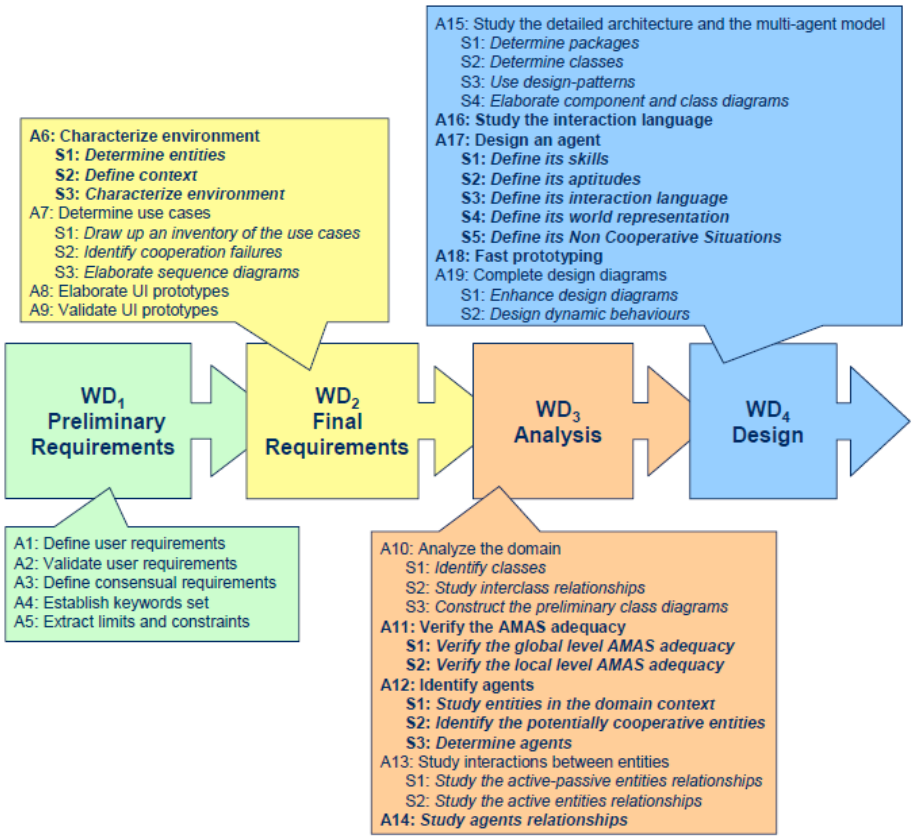


Figure 1: ADELFE workflow, sourced from [Bernon et al, 2005].

3.2 Fragmenting ADELFE

Although the Medee Delivery Process is composed of three phases, fragmentation itself is conducted by phases one (*Method Element Capture*) and two (*Method Fragment Elaboration*).

The *Method Element Capture* phase comprises the following tasks: Outline Method Content; Detail Method Content; Build AOSE Method As Is; Publish AOSE Method As Is, and its outcomes are SPEM elements such as work products, roles, tasks, guidelines captured from the target method, and the method as it is using those elements. It is all made using the tool EPF Composer [Haumer, 2007], a tool that implements SPEM.

Firstly, an understanding about the method was needed. It was made through [Bernon et al, 2005] and [ADELFE, 2003] . In addition, [Cossentino and Seidita, 2005] presented ADELFE fragmentation following another approach that uses an extended version of SPEM to describe fragments. Almost the same was presented in [ADELFE, 2003] . the difference between these them is that the first contain just one Phase for Requirements, while the second divides it in Preliminary Requirements and Final Requirements. Then, populating the first pillar of the repository was possible by mapping the SPEM elements already defined and describing them using the EPF Composer. To store the SPEM elements as tasks, roles, work products and guidance, a plugin named A: ADELFE Content Elements was created in EPF, containing the Main ADELFE Content Package, with divisions for each one of the elements, as can be seen at Figure 2.

Then, the method elements were organized in the form of process, to represent the method as the combination of SPEM elements, following [Bernon et al, 2005]. Therefore, the "method as it is" was stored in another EPF plugin, named A: ADELFE As Is. Figure 3 shows the work breakdown structure of the process related to the method.

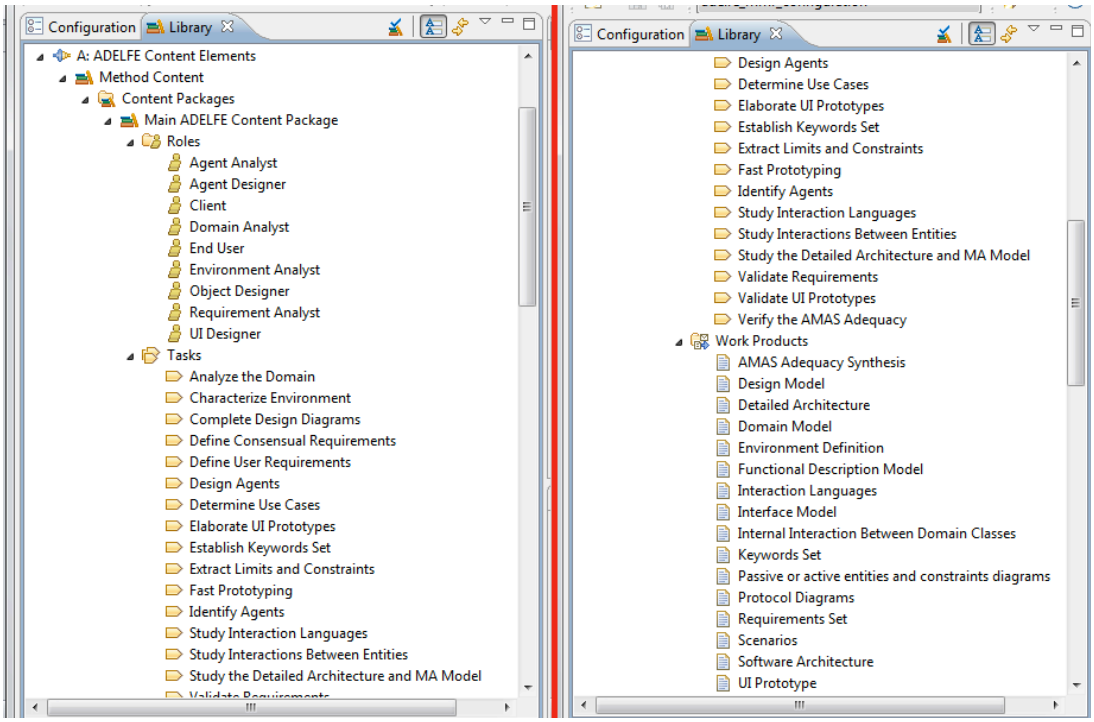


Figure 2: SPEM elements captured stored in the EPF Composer. The red line indicates a junction. Actually, it's a vertically continuous list.

Once that all method elements are defined and documented, the *Method Fragment Elaboration* phase begins, in order to populate the second pillar of the Medee Repository with fragments sourced from ADELFE. It consists in the following tasks: Build MAS Variability; Build Activity Method Fragment; Create Iteration Method Fragment; Create Phase Method Fragment; Create Process Method Fragment. Fragments must be defined according to granularity layers such as activity, phase or iteration and process. Also, they must be standardized as proposed at [Casare et al, 2013], in order to provide reuse and combination with other fragments sourced from different methods.

An activity fragment is a set of one or more tasks, considered as the smaller part of the method that can be fragmented and reused. For example, the activity showed on Figure 4 has four tasks. In our case, no iteration was created. A phase fragment is a set of activity fragments and here phase fragments were mapped to the phases proposed in "ADELFE as it is" process. A process fragment is a combination of phases. The phases are grouped in the ADELFE Base Method, which is basically the entire process that describes the method and can be used as a fragment. The fragments are stored in an EPF plugin named B: ADELFE Method Fragment. Figure 4 shows the Medee method fragments sourced from ADELFE, detailing at the right side the MMF Requirements Description with ADELFE one.

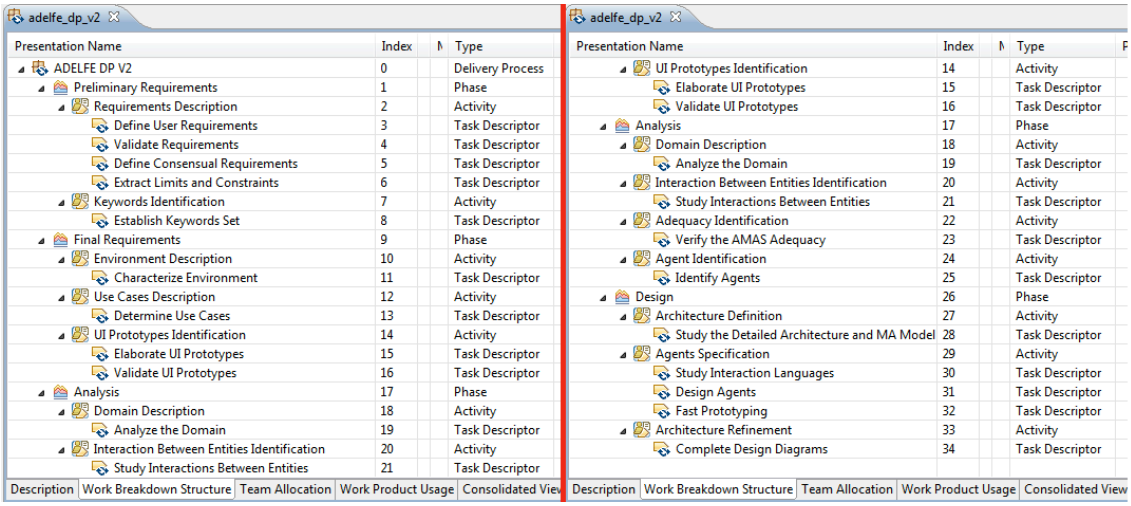


Figure 3: ADELFE method as it is. The red line indicates a junction; actually, it's a vertically continuous list.

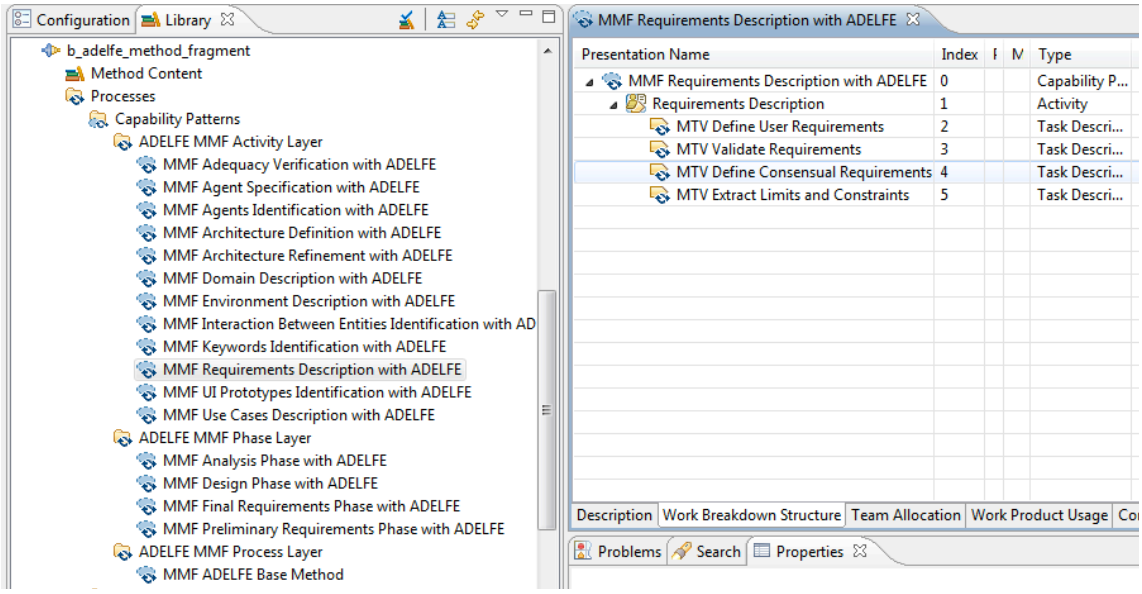


Figure 4: Overview of the fragments, organized in folders according to its granularity. On the right, details of an activity fragment, containing four task descriptors.

4. Conclusion

In this paper we describe the fragmentation of ADELFE following the Medee Delivery Process. The adoption of SPEM as the underlying technology to describe fragments facilitated the work while mapping existing description of ADELFE according to SPEM elements. Nevertheless, existing description didn't provide tool support for creating and storing fragments in CASE tools that implements SPEM, such as EPF Composer, in a sense that they adopts an extended version of SPEM that are not supported yet.

Seventeen new fragments were stored at the Medee Method Repository, increasing its population and diversity of sources in order to provide more flexible combination of fragments to create new situational methods to support MAS development.

Acknowledges

Thomas Liu de Almeida is partially supported by grant #2013-3326, Institucional CNPq. Anarosa A. F. Brandão is partially supported by grant #010/2640-5, São Paulo Research Foundation (FAPESP).

References

[ADELFE, 2003] ADELFE Process Description available at http://www.irit.fr/ADELFE/ENGLISH/AdelfeDescription.html. Accessed on 03/11/2014

[Bauer et al, 2001] Bauer, B.; Müller, J.; Odell, J.. Agent UML: A Formalism for Specifying Multiagent Software Systems. International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3, pp.1-24, 2001.

[Bernon et al, 2005] Bernon, C; Camps, V.; Gleizes, M.Picard, G.. Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology. Published in B. Henderson-Sellers and P. Giorgini, Es, Agent Oriented Methodologies, pages 172-202, Idea Group Publishing, 2005.

[Brinkkemper, 1996] Brinkkemper, S. 1996. Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology, vol. 38 (4), 275-280

[Casare et al, 2013] Casare, S. J. ; Brandão, Anarosa A.F. ; Guessoum, Z. ; Sichman, J.S. 2013. Medee Method Framework: a Situational Approach for Organization-Centered MAS. Autonomous Agents and Multi-Agent Systems (Dordrecht. Online), v. tbd, p. 1-44, 2013. (http://dx.doi.org/10.1007/s10458-013-9228-y).

[Cossentino, 2005] Cossentino M. 2005. From Requirements to Code with the PASSI Methodology. In: Henderson-Sellers, B. and Giorgini, P. (Eds) *Agent Oriented Methodologies*, pp.79-106, Idea Group Publishing, USA.

[Cossentino and Seidita, 2005] Cossentino, M. ; Seidita, V. . SPEM Description of ADELFE Process. 2005 Rapporto Tecnico N: RT-ICAR-PA-05-07 available at: http://www.pa.icar.cnr.it/cossentino/paper/adelfe_spem_05-07.pdf

[Giorgini et al, 2004] Giorgini, P. et al. 2004. The Tropos Methodology. In: Bergenti, V; Gleizes, M. P.; Zambonelli, F.(Ed.), Methodologies and software engineering for agent systems, Kluwer Academic Publishers, pp. 89-106.

[Haumer, 2007] Haumer, P. 2007. Eclipse Process Framework Composer – Part 1 – Key Concepts. Available at: <http://www.eclipse.org/epf>.

[Hubner et al, 2002] Hubner J., Sichman J., Boissier O. 2002. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Bittencourt, G. & Ramalho, G. L. (Eds.), 16th Brazilian Symposium on AI, SBIA'02, LNAI 2507, Berlin: Springer, p. 118-128

[SPEM 2008] OMG. (2008). Object Management Group. Software & Systems Process Engineering Meta-Model Specification, version 2.0. OMG document number: formal/2008-04-01. http://www.omg.org/spec/SPEM/2.0/PDF.

[Zambonelli et al, 2003] Zambonelli F., Jennings N. R., Wooldridge M. 2003. Developing multiagent systems: The Gaia methodology. ACM Transaction on Software Engineering and Methodology, vol 12(3), 417-470