ARTUR VIDAL MAIA

**Autonomia de planejamento no modelo
organizacional MOISE**

São Paulo
2018

ARTUR VIDAL MAIA

**Autonomia de planejamento no modelo
organizacional MOISE**

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2018

ARTUR VIDAL MAIA

**Autonomia de planejamento no modelo
organizacional MOISE**

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:

Engenharia de Computação

Orientador:

Prof. Dr. Jaime Simão Sichman

São Paulo
2018

Trabalho Aprovado
Banca Revisora

Prof. Dr. Jaime Simão Sichman - USP

Prof. Dr. Jomi Fred Hübner - UFSC

Prof. Dr. Felipe Meneguzzi - PUCRS

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Maia, Artur Vidal

Autonomia de planejamento no modelo organizacional MOISE / A. V. Maia -- versão corr. -- São Paulo, 2018.

77 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sistemas multiagentes I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Este trabalho é dedicado à todos
que acreditam que nossos erros são
a maior fonte de conhecimento.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à minha família que tanto me apoiou durante as noites escrevendo, em especial a minha mãe que sempre me incentivou a ficar calmo e ao meu pai que sempre foi rígido e deu sugestões sobre o que eu deveria fazer.

Também gostaria de agradecer e a todos os professores que contribuíram para que eu me interessasse sobre os mais diversos assuntos e para que este trabalho fosse realizado, em especial ao meu orientador que teve muita paciência comigo e minha ansiedade.

RESUMO

Essa dissertação apresenta um mecanismo para incorporar autonomia de planejamento em modelos organizacionais multiagentes. Para tanto, propõe-se um modelo formal para representar a presença e ausência de autonomia de planejamento, utilizando dois tipos de objetivos: procedurais e declarativos. O modelo é implementado na plataforma **JaCaMo** na qual se realiza um estudo de caso de uma organização, onde coexistem agentes que possuem ou não possuem autonomia de planejamento.

Palavras-Chave – Sistemas multiagentes, organizações de agentes, autonomia.

ABSTRACT

This dissertation presents a mechanism to incorporate planning autonomy in multi-agent organizational models. Therefore, we propose a formal model to represent presence or absence of planning autonomy, using two types of objectives: procedural and declarative ones. The model is implemented using the **JaCaMo** platform, in which an organization case study is proposed, where agents who have or do not have planning autonomy co-exist.

Keywords – multi-agent systems, agent organizations, autonomy.

LISTA DE FIGURAS

1	Influencia de diferentes tipos de autonomia em um agente.	23
2	Exemplo de uma especificação estrutural para um restaurante.	30
3	Exemplo de uma especificação funcional para um restaurante.	32
4	Como funciona a busca em espaço de estados para planejamento.	34
5	Taxonomia dos tipos de objetivo.	40
6	Redução do espaço de estados utilizando planejamento hierárquico com múltiplos agentes.	41
7	Funcionamento do arcabouço proposto para planejamento descentralizado.	42
8	Relação entre os elementos de um SMA e os elementos da Especificação de Domínio.	47
9	Exemplos de artefato.	51
10	Visão geral da plataforma JaCaMo.	53
11	Artefato DomainBoard usado para a EDom.	54
12	Artefato BasePlanner.	56
13	Diagrama de sequência para cumprimento de um objetivo procedural. . . .	59
14	Diagrama de sequência para cumprimento de um objetivo declarativo. . . .	60
15	Cenário de transporte de passageiros.	65
16	Especificação Estrutural para transporte de passageiros.	65
17	Especificação Funcional para transporte de passageiros.	66

SUMÁRIO

1	Introdução	12
1.1	Objetivos	13
1.2	Contribuições	14
1.3	Metodologia	14
1.4	Organização do Trabalho	15
2	Conceitos Básicos	17
2.1	Sistemas Multiagentes	17
2.1.1	Organizações	18
2.1.2	Autonomia	19
2.2	Autonomia	20
2.3	Organizações em SMA	24
2.3.1	Modelos Organizacionais Existentes	25
2.3.2	Modelo MOISE	26
2.3.3	Exemplo: Modelo Organizacional de um Restaurante	29
2.4	Planejamento	33
2.4.1	Tipos de Planejamento	35
2.4.2	Planejamento Hierárquico	35
2.4.3	Exemplo: Domínio de Planejamento de um Restaurante	39
2.5	Tipos de Objetivo	40
2.6	Trabalhos Relacionados	41
2.6.1	Organizações e Planejamento	41
2.6.2	Organizações e Autonomia	42

3	Autonomia de Planejamento em Organizações	43
3.1	Especificações de Domínio	43
3.2	Modelo Formal	44
3.3	Uso de Especificação de Domínio	48
3.3.1	Objetivos Declarativos	48
3.3.2	Objetivos Procedurais	48
4	Implementação	50
4.1	JaCaMo	50
4.1.1	Jason	50
4.1.2	CARtAgO	51
4.1.3	MOISE	52
4.1.4	Integração das Ferramentas	52
4.2	Artefatos	53
4.2.1	Especificação de Domínio (DomainBoard)	53
4.2.2	Planejador (BasePlanner)	55
4.3	Agentes	57
4.3.1	Ações Internas	57
4.3.2	Leitura da Especificação de Domínio	58
5	Estudo de Caso	64
5.1	Descrição do Cenário	64
5.2	Descrição da Organização	64
5.3	Agente sem planejamento on-line	67
5.4	Agente com planejamento hierárquico	68
5.5	Análise Comparativa	68
6	Conclusões e trabalhos futuros	72

6.1	Conclusões	72
6.2	Trabalhos Futuros	73
	Referências	74

1 INTRODUÇÃO

Inteligência Artificial (IA) é um campo de estudo da área de computação, surgido no meio do século 20 [RUSSELL; NORVIG, 2009]. Em seu início, a IA tinha forte influência de disciplinas mais tradicionais, como Filosofia e Matemática, e focava principalmente em métodos para aplicar o modo de pensar humano em computadores. De certa forma, pode-se dizer que a IA foi concebida como uma mistura de Ciência Cognitiva com Computação; com o passar dos anos, foram propostas diversas formas de integrar esse campo com conhecimentos vindos de outras áreas, tais como Biologia, Psicologia, Direito, etc.

Um conceito interessante que é estudado tanto em Biologia quanto em Psicologia é a forma como as pessoas e outros animais se agrupam em sociedades, criando relações harmônicas ou desarmônicas. Do ponto de vista biológico, essas relações dizem respeito às condições de vida de uma espécie, enquanto do ponto de vista psicológico pode-se compreender como as sociedades são organizadas. O estudo de características de tais sociedades, na área computacional, deu origem aos chamados Sistemas Multiagentes (SMA).

Basicamente, um SMA consiste em diversas entidades, denominadas agentes, que encontram-se situadas em um mesmo ambiente e que interagem entre si. Tais interações podem ser reguladas através de normas sociais. Na sociedade humana, tais normas formam a base da coordenação e cooperação entre indivíduos: um exemplo claro são as regras de trânsito, que se devidamente obedecidas, minimizam a ocorrência de acidentes e maximizam a utilidade da sociedade, por exemplo fazendo com que todos cheguem mais rapidamente a seus destinos.

Normas encontram-se também presentes nas organizações humanas, como empresas e indústrias. Por exemplo, as linhas de produção, surgidas no início do século XX, serviram para otimizar a produção de veículos, maximizando a utilidade da organização. Em SMA, existe também uma linha de pesquisa dedicada a modelar e implementar organizações de agentes com a finalidade de regular o comportamento dos agentes. Em particular, tais modelos podem impor planos pré-definidos, que se corretamente seguidos pelos agentes, atingirão os objetivos do SMA. Um modelo organizacional bastante utilizado pela comu-

nidade acadêmica é o modelo organizacional **MOISE** [HÜBNER, 2003].

Por outro lado, a regulação dos comportamentos dos agentes, tanto na sociedade humana como em SMA, se faz necessária: tratam-se de sistemas compostos de entidades *autônomas*, i.e., que têm o livre-arbítrio para seguir (ou não) as normas preestabelecidas. Cabe então às sociedades, quer humanas, quer de agentes, criar mecanismos para detectar possíveis violações destas normas e eventualmente aplicar um mecanismo de sanção a elas. Na literatura de SMA, uma proposta para tal mecanismo é apresentada em [NARDIN et al., 2016] e [LIMA; NARDIN; SICHMAN, 2018].

Por outro lado, em certas ocasiões, a existência de entidades autônomas pode ser benéfica, já que podem propor cursos de ação distintos daqueles previamente previstos em casos de contingências.

Dentro do contexto de autonomia, podemos falar sobre um caso específico de autonomia que as entidades de uma organização podem ter. Conhecida como autonomia de planejamento e define se essa entidade tem ou não autonomia para criar planos (ou definir processos) dentro do contexto organizacional.

Em organizações humanas, esse tipo de autonomia pode ser percebido em pessoas que têm liberdade para tomar decisões sobre como ela (ou outros) podem atuar dentro da organização. Como exemplo, pode-se citar um diretor que cria novas estratégias para a venda de um produto. A ausência desse tipo de autonomia também pode ser percebida em pessoas que devem seguir planos ou ações pré-definidos, como um vendedor que deve simplesmente seguir a estratégia de venda definida.

A motivação deste trabalho é representar explicitamente tais graus de autonomia em modelos organizacionais de agentes. Os modelos organizacionais estudados nesse trabalho não permitem tal distinção. A possibilidade de representar os diferentes tipos de objetivos presentes em organizações reais permite a criação de organizações mais funcionais, uma vez que cada tipo de objetivo tem suas vantagens e desvantagens.

1.1 Objetivos

O objetivo deste trabalho é propor um modelo, implementá-lo e avaliar sua eficácia para permitir autonomia de planejamento em modelos organizacionais de agentes.

Mais especificamente, pretende-se:

1. Propor um modelo formal que represente a presença ou ausência de autonomia de

planejamento em modelos organizacionais de agentes;

2. Demonstrar como o modelo proposto pode ser aplicado junto ao modelo MOISE [HÜBNER, 2003];
3. Implementar o modelo e analisar os resultados obtidos num estudo de caso de uma organização, na qual coexistem agentes que possuem e que não possuem autonomia de planejamento.

1.2 Contribuições

As contribuições esperadas deste trabalho são as seguintes:

- Um modelo formal para representar a presença ou a ausência de autonomia de planejamento em modelos organizacionais multiagentes;
- Uma implementação do modelo no arcabouço JaCaMo [BOISSIER et al., 2013], permitindo a construção de agentes com ou sem autonomia de planejamento em uma organização MOISE [HÜBNER, 2003];
- Uma implementação de um agente com autonomia de planejamento em Jason, que utiliza um planejador SHOP [NAU et al., 1999] programado em um artefato;
- Um estudo de caso de uma organização onde coexistam agentes com e sem autonomia de planejamento.

1.3 Metodologia

Para alcançar o objetivo geral descrito, o trabalho foi desenvolvido nas seguintes etapas:

Identificação do problema: Modelos organizacionais são especificações de alto nível de abstração a serem utilizadas como guias para o projeto de organizações concretas. Na prática, a aplicação destes modelos deixa um grande número de decisões de projeto para serem tomadas ad-hoc pelo projetista da organização. Uma destas lacunas é o suporte do modelo organizacional para se especificar a presença ou ausência de autonomia dos agentes;

Proposição conceitual e modelo formal: Para incluir autonomia de planejamento em um modelo organizacional, foi utilizada uma distinção entre diferentes tipos de objetivos. Essa distinção é baseada no formalismo de planejamento e em trabalhos anteriores sobre tipos de objetivos. Nessa etapa foi proposto um modelo formal, chamado de especificação de domínio (EDom). Tal modelo foi incorporado ao modelo MOISE [HÜBNER, 2003], sem alterar as definições já existentes deste último. Foi baseado em conceitos existentes na área de planejamento e programação de agentes, especificamente no formalismo de planejamento HTN [EROL; HENDLER; NAU, 1995] e no paradigma de agentes BDI [RAO, 1996]. O modelo propõe uma representação para os diferentes tipos de objetivos que podem existir numa organização: *procedurais*, cujos agentes comprometidos não têm autonomia para decidir como agir, e *declarativos*, que permitem ao agente criar/selecionar seu plano baseado nos estados do mundo requeridos de se atingir.

Implementação do modelo (Sintaxe): Nessa etapa foi implementada a proposta de Especificação de Domínio em uma linguagem a fim de permitir a sua interpretação pelos agentes. A proposta foi implementada no arcabouço JaCaMo e foi utilizado HTN como o formalismo de planejamento. HTN foi escolhido como formalismo por dois motivos principais. Em primeiro lugar, porque já que os agentes presentes em JaCaMo usam o paradigma BDI na linguagem Jason, e na literatura existem vários trabalhos mostrando as similaridades entre HTN e BDI, como [SARDINA; SILVA; PADGHAM, 2006; MENEGUZZI; SILVA, 2015; MENEGUZZI; LUCK, 2007; MENEGUZZI; ZORZO; MÓRA, 2004]; em segundo lugar, porque o modelo MOISE define a relação entre seus objetivos através de uma árvore de decomposição onde é comum os agentes tentarem alcançar objetivos-folhas. Logo, continuar a sua decomposição no agente é um processo similar ao que já é feito na organização;

Estudo de caso: Para demonstrar e avaliar as novas possibilidades que serão permitidas por autonomia de planejamento, foi realizado um estudo de caso em um ambiente que exige cooperação entre os agentes, exemplificando situações onde o sistema possa se beneficiar de possuir os diferentes tipos de objetivos definidos. Foi utilizado para tal o framework JaCaMo, abordando um cenário de transporte de passageiros.

1.4 Organização do Trabalho

Nesse primeiro capítulo, foram apresentados a motivação, a descrição do problema, os objetivos, as contribuições e a metodologia utilizados no trabalho. No capítulo 2 são

apresentados os conceitos necessários para a compreensão do trabalho, concluindo com uma seção onde os principais trabalhos relacionados são discutidos. No capítulo 3 é detalhado o modelo formal e no capítulo 4 a sua implementação no framework **JaCaMo**. O capítulo 5 apresenta o estudo de caso realizado, assim como uma análise dos resultados obtidos. Finalmente, no capítulo 6 as conclusões do trabalho são apresentadas e possíveis futuras extensões.

2 CONCEITOS BÁSICOS

2.1 Sistemas Multiagentes

Um conceito interessante que é estudado tanto em Biologia quanto em Psicologia é a forma como as pessoas e outros animais se agrupam em sociedades, criando relações harmônicas ou desarmônicas. Do ponto de vista biológico, essas relações dizem respeito às condições de vida de uma espécie, enquanto do ponto de vista psicológico pode-se compreender como as sociedades são organizadas. O estudo de características de tais sociedades, na área computacional, deu origem aos chamados Sistemas Multiagentes (SMA).

Um SMA é fortemente influenciado pelo conceito de sociedades. Ele consiste em diversas entidades situadas em um mesmo ambiente que interagem entre si através de normas sociais. Entretanto, ao invés de estudar espécies, como na Biologia, ou humanos, como na Psicologia, SMA estuda entidades virtuais, conhecidas como agentes inteligentes, e esse é o nosso tópico de interesse [RUSSELL; NORVIG, 2009].

Tendo em vista a similaridade entre SMA e seus tópicos equivalentes em outras ciências, muitos conceitos foram trazidos destas para serem aplicados aqui. Entre eles, pode-se citar algoritmos que utilizam conceitos similares aos feromônios usados por formigas em suas colônias; equações de Lotka–Volterra (mais conhecidas como modelo presa-predadora) que descrevem a interação entre uma espécie presa e outra espécie predadora em sistemas biológicos; e o conceito de organizações sociais que estudam como indivíduos interagem a fim de alcançarem um objetivo comum.

Mais especificamente, será aprofundado o conceito de organizações aplicado a SMAs. Um dos primeiros trabalhos sobre esse conceito foi proposto por [FERBER; GUTKNECHT, 1998], onde os autores propõem um modelo formal para representar organizações computacionais, baseado em organizações reais. Seu trabalho argumenta que considerar conceitos organizacionais como elementos primários quando se projeta um sistema é um ponto chave para construir sistemas complexos e de larga escala, e que a presença desses conceitos ajuda a resolver muitos problemas encontrados caso se adote um ponto de vista

orientado exclusivamente ao agente. Com o passar dos anos, o estudo de organizações de agentes foi evoluindo, criando e trazendo cada vez mais técnicas e conceitos vindos da psicologia e biologia para SMAs.

2.1.1 Organizações

Uma organização pode ser definida como uma série de normas sociais que os agentes devem seguir [FERBER; GUTKNECHT, 1998]. O estudo de organizações em SMAs possui diversas aplicações, e entre elas pode-se mencionar a simulação de comportamentos do mundo real em sistemas virtuais e a integração e interação entre sistemas heterogêneos e/ou distribuídos.

Organizações têm como função principal promover a cooperação, e para estudá-las é importante olhar tanto para os indivíduos presentes quanto para as suas interações; é importante também considerar o seu impacto tanto em escala micro (ou como ela irá influenciar o comportamento de cada agente) quanto em escala macro (sua influência no sistema como um todo).

De uma maneira simples, as organizações restringem o comportamento dos agentes que dela participam, tentando assim fazê-los cooperar para que atinjam um objetivo global. Essas restrições podem ser consideradas como maleáveis, uma vez que em alguns casos um agente pode escolher entre seguir ou não as normas da organização. Essa distinção é muito importante: infringir as normas de uma organização pode levar a comportamentos indesejados, mas também permite aos agentes se adaptarem a circunstâncias inesperadas que não foram consideradas durante o projeto da organização. Afinal, nem sempre é possível prever antecipadamente todas as situações que podem ocorrer num determinado contexto. Essa propriedade de adaptar-se para resistir às especificações imprecisas ou incompletas pode ser vista como um tipo de robustez do projeto organizacional.

As diferentes restrições das organizações reais impõem diferentes tipos de comportamentos aos indivíduos participantes. A fim de se representar essas restrições e reproduzir os comportamentos em um ambiente computacional, os modelos de especificação organizacional evoluíram nesse sentido, e um mesmo modelo pode ser usado para impor diferentes restrições organizacionais. Diferentes organizações podem levar a diferentes resultados com respeito a desempenho, facilidade, clareza ou robustez da implementação. Por exemplo, no modelo AGR [FERBER; GUTKNECHT; MICHEL, 2003], que é composto somente de um aspecto estrutural para definir relações entre os agentes, as restrições

são representadas por axiomas tais como: “Dois agentes podem se comunicar apenas se pertencerem ao mesmo grupo”. Já o modelo **MOISE** [HÜBNER, 2003], que é composto de diferentes especificações (estrutural, funcional e deôntica), as restrições podem abordar outros aspectos tais como “Um agente pode se comunicar com outro apenas se existir um link de comunicação ou autoridade entre os papéis dos dois agentes na especificação estrutural” ou “um agente somente pode tentar alcançar um objetivo se o seu papel permitir”.

Uma descrição mais detalhada sobre modelos organizacionais de agentes se encontra na Seção 2.3. Entretanto, para melhor compreender o efeito das restrições impostas aos indivíduos de uma organização, real ou virtual, é importante considerar o conceito de autonomia.

2.1.2 Autonomia

O conceito de autonomia foi muito estudado por filósofos, sendo importante também para o domínio de Inteligência Artificial, já que agentes autônomos estão presentes em vários sistemas e são usados em uma grande gama de aplicações. No contexto de SMA, um agente é dito autônomo quando é capaz de tomar suas próprias decisões, sem imposição do meio externo. Deste modo, um agente ter autonomia e pertencer a uma organização podem ser vistos como conceitos opostos: o primeiro permite a um agente agir livremente, enquanto o segundo restringe seu comportamento. Entretanto, eles também podem ser vistos como conceitos complementares, uma vez que tanto a presença como a ausência de autonomia podem ser benéficas.

Permitir a um agente alternar entre um comportamento autônomo ou não pode ser benéfico para o sistema de diversas maneiras, permitindo restringir o comportamento do agente quando necessário ou deixá-lo livre para se adaptar caso existam condições que requeiram tal adaptação. Esse fenômeno é chamado de *autonomia ajustável* [BARBER et al., 2000; VECHT et al., 2008; MAIA; SICHMAN, 2017]. Para permitir essa troca dentro de um sistema, é necessário que exista uma definição e uma maneira de se especificar se um agente possui ou não comportamento autônomo.

Fazendo uma analogia com organizações reais, pode-se perceber que elas normalmente são compostas por pessoas responsáveis por tomar decisões e pessoas responsáveis por executar tarefas. Os responsáveis por tomar decisões normalmente conseguem considerar vários aspectos de uma situação e definir um plano de ações a fim de cumprir um determinado objetivo, caracterizando autonomia de planejamento. Por outro lado,

os responsáveis por executar tarefas apenas têm que cumprir processos já estabelecidos na organização e não têm liberdade para decidir como agir, modificar ou escolher esses processos, caracterizando ausência de autonomia.

Na versão atual do modelo MOISE [HÜBNER, 2003], objeto deste trabalho, não é possível para o projetista da organização atribuir aos agentes participantes um comportamento autônomo ou não autônomo para atingir um certo objetivo, sendo sempre assumida a autonomia dos agentes. Essa autonomia é especificada diretamente nos agentes, não dando ao projetista da organização um controle mais fino sobre a decomposição do objetivo global da organização. Em algumas situações é importante o projetista da organização não deixar essa autonomia ser decidida ad-hoc pelo implementador do agente.

Devido à natureza heterogênea e distribuída de SMAs, o projeto e implementação dos agentes e da organização podem não ser realizados pelo mesmo projetista, o que pode possibilitar que os agentes assumam comportamentos autônomos para determinados objetivos não previstos ou não desejados pelo projetista da organização. Esse trabalho apresenta uma proposta para resolver esse problema.

2.2 Autonomia

Autonomia é um conceito fundamental da área de sistemas multiagentes. Como já mencionado anteriormente, tal conceito trata sobre a capacidade dos agentes de tomarem decisões, levando em conta a influência de estímulos externos. Na literatura, foram apresentadas diversas definições para autonomia. [CARABELEA; BOISSIER; FLOREA, 2003] consideram um agente autônomo como aquele cujo comportamento não é imposto por uma entidade externa e [CASTELFRANCHI, 2000] estabelece autonomia como a não-dependência de um elemento externo. Apesar de similares, ao analisar essas duas definições pode-se perceber que na primeira o conceito é mais relacionado com condições externas, sendo a ausência de autonomia imposta ao agente. Já na segunda definição, a autonomia é relacionada internamente ao agente, através da falta de um recurso ou outro elemento que o agente necessita para cumprir seus objetivos.

O estudo de autonomia é intrinsecamente relacionado a sistemas multiagentes. [CASTELFRANCHI, 1994] afirma que autonomia é uma relação social e que não faz sentido se falar de autonomia quando ela é extrema (um agente completamente autônomo ou autárquico), e que autonomia só é pertinente quando for limitada (um agente pode ter seu objetivo decidido ou influenciado por fontes externas).

O trabalho de [CARABELEA; BOISSIER; FLOREA, 2003] define autonomia em cinco categorias diferentes:

- **Auto-Autonomia:** é definida como “a propriedade que permite um agente ter ou escolher entre diferentes comportamentos possíveis”. Um dos principais aspectos que definem um agente inteligente é a sua capacidade de tomar decisões baseadas a partir de seu modelo mental, logo esse tipo de autonomia está sempre presente implicitamente na definição de todo agente. Um caso especial é o de um agente puramente reativo [WOOLDRIDGE, 2009], cujo comportamento é apenas uma resposta direta ao ambiente e pode ser representado por uma função. Esse tipo de autonomia garante que todos os outros tipos podem existir, sendo as outras definições de autonomia apenas casos particulares desse tipo, nos quais o processo de tomada de decisão do agente é parcialmente ou completamente definido por um elemento externo, seja através de normas sociais, de um usuário, um outro agente ou o ambiente.
- **Autonomia-Ambiental:** caso exista, “o ambiente pode apenas influenciar o comportamento de um agente, ele não pode o impor”. Isso foi primeiramente estudado como o ‘problema de Descartes’: respostas de muitos sistemas vivos “não são causadas por, nem independente de estímulos externos”. Todos os tipos de agentes, assim como seres vivos, utilizam de seu modelo mental para processar estímulos do ambiente e resultar em um comportamento. Entretanto, como esses estímulos passam por uma camada subjetiva, que é o modelo mental, pode-se dizer que todo agente tem Autonomia-Ambiental.
- **Autonomia-Social:** é definida como “um agente X é autônomo em respeito a um outro agente Y para a adoção de um objetivo G se X pode recusar a adoção desse objetivo G de Y”. A autonomia-social é um dos tipos mais comuns de autonomia que se pode encontrar em SMAs. Em um ambiente onde existem muitos agentes, suas interações podem resultar tanto na presença quanto na ausência deste tipo de autonomia. Apesar da definição apresentada para autonomia-social falar apenas de objetivos, pode-se expandi-la para uma definição mais geral a fim de incluir diversos elementos, como recursos, planos, permissões, etc. Um caso especial para a Autonomia-Social é quando existe uma norma que especifica autoridade entre agentes (autoridade pode ser definida como a habilidade de impor algo a alguém); logo, se no sistema existe uma norma que dita autoridade entre agentes, uma relação de autonomia-social está implícita nessa norma.

- **Autonomia-de-Usuário:** é definida como “um agente é autônomo em respeito ao usuário para escolher qual ação irá realizar se ele puder fazer essa escolha sem interferência do usuário”; se a qualquer momento um agente é impedido de alcançar seus objetivos por uma interferência do usuário, o agente não possui autonomia-de-usuário.
- **Autonomia-Normativa:** é definida como “um agente é autônomo com respeito a uma norma se ele pode violar essa norma”. Uma norma é uma regra ou diretriz que o agente deve seguir. Para alcançar a Autonomia-Normativa, um agente necessita reconhecer as normas e ser capaz de raciocinar sobre elas. Agentes cientes de normas [ALECHINA; DASTANI; LOGAN, 2012] foram estudados em trabalhos na área de SMA que tratam especificamente sobre eles. Um dos aspectos mais relevantes nesses estudos é sobre como monitorar violações de normas [BULLING; DASTANI; KNOBBOUT, 2013]. Normas podem ser sumarizadas em dois tipos: regimentadas e reguladas [BOELLA; TORRE, 2004]. Normas regimentadas são normas que não podem ser violadas, e sua forma mais comum é através da codificação no agente ou sistema. Já as normas reguladas podem ser violadas e possivelmente o agente que as obedecem ou violam pode sofrer sanções. Tais normas são normalmente usadas como uma maneira de melhorar o comportamento do agente ainda que lhe permitindo adaptar-se ao realizar violações. Um mecanismo para efetivar tais sanções é apresentado em [NARDIN et al., 2016].

A Figura 1 mostra onde cada um dos tipos de autonomia apresentados anteriormente restringe o comportamento de um agente. A Autonomia-de-Usuário permite ou não a um usuário influenciar as decisões do agente. A Auto-Autonomia é a própria capacidade do agente de tomar decisões. Autonomia-Social restringe o comportamento do agente através de suas interações com outros agentes. Autonomia-Normativa restringe através de normas (na figura representadas por uma organização). Finalmente, Autonomia-Ambiental restringem as ações permitidas e influencia as decisões do agente através de percepções.

Apesar da maioria das definições de autonomia existentes se encaixarem em uma dessas categorias, o trabalho de [CARABELEA; BOISSIER; FLOREA, 2003] fala que “cada e qualquer componente da arquitetura ou condição necessária para se realizar uma ação com sucesso pode definir uma dimensão/parâmetro de autonomia”. Logo, para se avaliar o tipo de autonomia presente ou não em um sistema é necessário primeiro avaliar como ele foi desenvolvido, uma vez que os diferentes elementos da arquitetura podem definir diferentes tipos de autonomia. Percebe-se que quanto maior o número de

Figura 1: Influencia de diferentes tipos de autonomia em um agente.

Fonte: Autor.

processos mentais e físicos que um agente possui, maior a quantidade de possibilidades que um elemento externo pode influenciar. Dependendo da arquitetura do sistema, podem existir relações de causa e consequência entre os diferentes tipos de autonomia presentes, e diferentes granularidades ou subtipos dentro dessas categorias.

Em particular, um tipo de autonomia importante para este trabalho não é definida em [CARABELEA; BOISSIER; FLOREA, 2003], sendo definida a seguir.

Autonomia de Planejamento

Neste trabalho, propõe-se relacionar organizações a um tipo de autonomia chamado autonomia de planejamento [CASTELFRANCHI, 2000]. Esse tipo de autonomia diz respeito à possibilidade do agente criar, executar, modificar ou escolher planos para atingir um determinado objetivo.

Autonomia de planejamento pode estar presente em diferentes contextos. O mais comum é o contexto do próprio agente, quando ele tem autonomia para criar, executar, modificar ou escolher planos para seus próprios objetivos. Entretanto, quando temos um sistema multiagentes, podemos perceber a autonomia de planejamento para objetivos de outras entidades (i.e., outros agente, uma organização, etc.). Logo, ela pode estar também relacionada com as categorias de autonomia-social ou autonomia-normativa.

Um trabalho que relaciona o conceito de autonomia com sistemas multiagentes é

[BRADSHAW et al., 2003]; entretanto, não foi encontrado nenhum trabalho que relaciona autonomia de planejamento dentro do contexto organizacional de forma *explícita* (com a especificação da presença ou ausência de autonomia como uma norma). Na seção 2.6, esse assunto será mostrado de maneira mais detalhada.

2.3 Organizações em SMA

A organização, como uma das principais características de um SMA, não apenas fornece uma maneira de analisar quais os papéis que os agentes têm no sistema como um todo, como também ajuda a analisar o fluxo de comunicação que acontece no SMA e complementa o conceito de agentes, com o objetivo de simplificar seus modelos e reduzir incertezas [DELOACH; MATSON, 2004].

O trabalho de [COUTINHO et al., 2005] propõe um conjunto de dimensões organizacionais que permitem comparar diversos modelos de organização retirados da literatura. Em [COUTINHO; SICHMAN; BOISSIER, 2009], os autores estendem o trabalho anterior, propondo quatro dimensões básicas e outras quatro dimensões complementares para caracterizar os modelos organizacionais de agentes:

- Dimensões Básicas

- *Estrutura Organizacional*: Representa aspectos estruturais da organização normalmente invariantes no tempo;
- *Funções Organizacional*: Representam objetivos globais e decomposição de objetivos que devem ser atingidos pela organização;
- *Interações Organizacionais*: Representam aspectos dependentes do tempo, referentes a interações que envolvam elementos da estrutura e das funções organizacionais;
- *Normas Organizacionais*: Regulam e mostram como a estrutura organizacionais, as funções organizacionais e as interações organizacionais se inter-relacionam.

- Dimensões Complementares

- *Avaliação Organizacional*: Permite medir o desempenho de uma organização de agentes com respeito a cumprir seus objetivos específicos;

- *Evolução Organizacional*: Permite realizar alterações nos elementos da organização, em algum instante de tempo, para adaptá-la a novas demandas do ambiente;
- *Ambiente Organizacional*: Representa o conjunto de recursos formados por entidades não-autônomas, cujos agentes da organização podem perceber e sobre os quais podem agir;
- *Ontologias Organizacionais*: Conceituações do domínio de aplicação da organização de agentes que podem ser compartilhadas de modo consistente pelos agentes que a compõem.

Por exemplo, no que diz respeito às dimensões básicas, o modelo MOISE [HÜBNER, 2003] contempla as três primeiras.

Na sequência serão apresentados os principais modelos organizacionais utilizados na área de agentes.

2.3.1 Modelos Organizacionais Existentes

O trabalho de [HORLING; LESSER, 2004] analisa e compara os paradigmas organizacionais mais comuns: coalizões, equipes, congregações, hierarquias, holarquias, federações, mercados, matrizes e organizações compostas. Cada um desses paradigmas tem suas próprias vantagens e desvantagens, fazendo-se necessário que o projetista da organização avalie o problema que os agentes irão encontrar para que seja possível escolher o melhor modelo organizacional a ser implementado.

Isso levanta a questão de como especificar e disponibilizar uma organização aos agentes de um sistema. Tentando responder a essa questão, diversos modelos de como se especificar uma organização foram propostos, como: MOISE [HÜBNER, 2003], AGR [FERBER; GUTKNECHT; MICHEL, 2003], TÆMS [DECKER, 1996].

Os elementos principais do modelo AGR são Agentes, Grupos e Roles (papéis), e é composto de axiomas e restrições estruturais que dizem como eles se relacionam e como definir uma organização.

Já o modelo TÆMS é mais voltado para o ambiente onde os agentes irão agir, sendo os agentes apenas entidades com crenças e que realizam ações no ambiente. Neste modelo, a coordenação entre os agentes se dá através das estruturas que os agentes usam para a transição de estado, sendo elas controle, coleta de informação, comunicação e execução de método.

2.3.2 Modelo MOISE

O modelo organizacional que será usado nesse trabalho é o modelo MOISE [HÜBNER, 2003]. Ele propõe a especificação de uma organização em três dimensões: Estrutural, Funcional e Deontica¹. A autonomia de um agente pode ser altamente influenciada pela organização. Essa influência se encaixa na categoria de Autonomia-Normativa, uma vez que uma organização é um conjunto de normas.

Uma vez que uma organização MOISE [HÜBNER, 2003] esteja especificada, os agentes participantes dela precisam seguir as restrições de comportamento impostas pela organização para que possam cooperar com outros agentes. A organização não força um comportamento específico ao agente; ao invés disso, ela decompõe objetivos de mais alto nível em subobjetivos, que em seguida são passados para os agentes de um papel através de obrigações ou permissões. Como pode ser visto, o objetivo principal de uma organização é representar de uma maneira indireta a forma como os agente irão cooperar. Mas, enquanto a organização especifica subobjetivos, ela não diz a um agente como cumpri-los ou como agir especificamente. Um modelo organizacional é um conjunto de normas que, quando instanciadas, resultam em uma entidade organizacional específica e concreta destinada a resolver um problema. O fluxo de funcionamento é que os agentes assumem papéis na organização e tentam realizar os objetivos associados a eles através de missões. Dessa maneira, cada agente, durante sua existência, executa o ciclo de receber um objetivo, tentar satisfazê-lo, comunicar o resultado a entidade organizacional, receber um novo objetivo, etc. Nos próximos parágrafos será descrito o modelo formal MOISE [HÜBNER, 2003], o qual é utilizado nesse trabalho.

Uma Especificação Organizacional $EO = (SS, FS, DS)$ é o conjunto das especificações Estrutural (SS), Funcional (FS) e Deontica (DS).

Especificação Estrutural (SS)

A especificação estrutural é responsável por definir os papéis e grupos da organização, assim como as ligações e hierarquias entre eles. O propósito da SS é restringir o comportamento do agente uma vez que ele escolha assumir um papel na organização, limitando a quantidade de interações que ele pode ter com outros agentes participantes da mesma organização. Ela é composta dos seguintes elementos: papéis, ligações (links), compati-

¹As dimensões Estrutural, Funcional e Deontica do modelo MOISE equivalem respectivamente às dimensões básicas de estrutura, funções e normas organizacionais propostas em [COUTINHO; SICHMAN; BOISSIER, 2009].

lidade e grupos.

Papéis: Um papel ρ é um elemento de um conjunto de papéis R . Um papel é um identificador ou etiqueta para um agente na organização. Um agente participa de uma organização escolhendo um papel disponível para assumir. Papéis por si só não limitam o comportamento do agente, entretanto eles são usados por outros elementos da especificação para restringir o comportamento.

Ligações (links): Um link é definido como $link(\rho_1, \rho_2, t)$ onde:

ρ_1 e ρ_2 são papéis tirados de R , e t é o tipo de ligação, pertencente ao conjunto (acq, com, aut) , onde acq é uma ligação de conhecimento, com é uma ligação de comunicação e aut é uma ligação de autoridade. Uma ligação entre dois papéis seguida de um tipo significa que as interações entre esses papéis são restringidas dependendo do tipo dessa ligação.

Compatibilidade A compatibilidade entre dois papéis é definida como $\rho_1 \bowtie \rho_2$. Compatibilidade significa que um agente de um papel ρ_1 pode também assumir um outro papel ρ_2 .

Grupos Um grupo gt de um conjunto de grupos GT é definido como:

$gt = (R_g, SG_g, L^{intra}, L^{inter}, C^{intra}, C^{inter}, np, ng)$ onde:

R_g é um subconjunto de papéis existentes no grupo. Ele é um subconjunto de todos os papéis da especificação, $R_g \subset R$;

SG_g é um subconjunto de possíveis subgrupos nesse grupo, $SG_g \subset GT$;

L^{intra}, L^{inter} são, respectivamente, as ligações internas e externas do grupo;

C^{intra}, C^{inter} são, respectivamente, os conjuntos de compatibilidades internas e externas do grupo;

np é uma função que mapeia um papel para sua cardinalidade mínima e máxima;

ng é uma função que mapeia um subgrupo para sua cardinalidade mínima e máxima.

Um grupo representa um conjunto de agentes que supostamente tem funções similares ou relacionadas na organização. Um grupo pode ser tratado como uma entidade única com um objetivo o qual todos os agentes pertencentes ao grupo irão tentar alcançar. Enquanto grupos não aumentam a representatividade do modelo, eles permitem projetar organizações de uma maneira mais fácil e parecida com as organizações reais.

Especificação Funcional (FS)

A especificação funcional contém os objetivos e missões da organização e como eles devem ser cumpridos em relação a outros. O objetivo principal da FS é definir relações lógicas, hierárquicas ou de ordem entre objetivos a fim de especificar como o objetivo global deve ser atingido. Ela é composta de: missões, objetivos, planos e esquemas sociais.

Metas: Uma meta g é um elemento de um conjunto de metas G . Uma meta representa um estado do mundo que a organização deseja que seja alcançado. Através de esquemas sociais pode-se colocar as metas em diferentes categorias, como será visto adiante.

Missões: Um missão m é um elemento de um conjunto de missões M e é definida como um subconjunto de objetivos de G que deve ser atingido por um único agente ou um grupo de agentes comprometidos com a missão. Uma missão é usada como um identificador para um conjunto de objetivos.

Preferências entre missões: A preferência de uma missão m_1 sobre uma missão m_2 e definida como: $m_2 \prec m_1$.

Plano: Um plano p é um elemento de um conjunto de planos P e é representado na forma $p = (head, op, G^n)$, onde:

$head$ é a meta do plano;

G^n são as submetas do plano

op são os operadores do plano, com $op \in \{, , |, ||\}$.

Um plano representa a decomposição de uma meta em submetas utilizando-se de operadores para definir as relações entre as mesmas.

Esquema Social: Um esquema social sch é um elemento de um conjunto de esquemas sociais SCH e é definido como $sch = (G, P, M, mo, nm)$ onde:

G é o conjunto de metas do esquema;

P é o conjunto de planos que decompõem os objetivos em G ;

M é o conjunto de missões do esquema;

mo é uma função que associa metas a missões do esquema;

nm é uma função que dita o número máximo e mínimo de agentes para cada missão.

Um esquema social é a parte da FS que a organização irá utilizar mais ativamente. A FS de uma organização pode contar vários esquemas sociais e os agentes podem completá-los através da preferência entre missões. Diversos esquemas sociais podem estar ativos ao mesmo tempo, mas existe sempre um esquema padrão que contém todos os objetivos, todas as missões, todos os planos e suas decomposições. A meta no topo da árvore de decomposição desse esquema padrão representa o objetivo global da organização.

Especificação Deontica (DS)

A especificação deontica é responsável por definir a relação entre a FS e a SS. Ela relaciona papéis da SS com missões da FS utilizando-se dos conceitos de permissão e obrigação, que fazem os agentes que desempenham um papel respectivamente escolher qual objetivo desejam cumprir ou quais são forçados a cumprir.

Uma relação deontica é definida como $norm(\rho, m, type)$ onde: ρ é um papel tirado de R , m é uma missão tirada de M e $type$ é o tipo de relação (*obligation*, *permission*), onde *obligation* é uma relação de obrigação e *permission* é uma relação de permissão.

Como o objetivo deste trabalho é incorporar a noção de autonomia de planejamento em modelos organizacionais de agentes, na sequência serão apresentados brevemente os principais conceitos de planejamento que foram utilizados para esta finalidade.

2.3.3 Exemplo: Modelo Organizacional de um Restaurante

A seguir será apresentado a definição de uma organização de um restaurante usando o modelo organizacional MOISE . s

Especificação Estrutural

A Figura 2 mostra a especificação estrutural usada para organizar um pequeno restaurante. Ela é composta de 3 grupos, sendo uma equipe de cozinha (*kitchen group*), uma equipe de salão (*tables group*) e um grupo geral de membros do restaurante (*restaurant members*). A equipe de cozinha (*kitchen group*) é composta de papéis responsáveis por preparar comida (*chef* e *sous-chef*), e a equipe de salão contém papéis responsáveis por interagir com os clientes (*server* e *cashier*). Os cozinheiros podem se comunicar com os garçons (e vice-versa) para informar pedidos e estes últimos podem se comunicar com o

Figura 2: Exemplo de uma especificação estrutural para um restaurante.

Fonte: Autor.

caixa para processar as contas. O *chef* tem autoridade sobre o *sous-chef* e, se necessário, pode assumir seu papel.

A definição formal da especificação estrutural pode ser vista no esquema $ss_{restaurant}$. Ela é composta do grupo $gt_{restaurant}$ que está especificado no esquema com seus subgrupos gt_{tables} e $gt_{kitchen}$, possuindo cardinalidade 1 para cada um desses grupos:

$$ss_{restaurant} = (\{gt_{restaurant}\}, R_{restaurant}, \sqsubseteq)$$

$$gt_{restaurant} = ($$

$\{\},$	$\%R$
$\{gt_{tables}, gt_{kitchen}\},$	$\%SG$
$\{\},$	$\%L^{intra}$
$\{\},$	$\%L^{inter}$
$\{\},$	$\%C^{intra}$
$\{\},$	$\%C^{inter}$
$\{\},$	$\%np$
$\{gt_{tables} \mapsto (1, 1), gt_{kitchen} \mapsto (1, 1)\}$	$\%ng$

O grupo gt_{tables} é composto pelos dois papéis $\rho_{server}, \rho_{cashier}$, possuindo uma ligação

intragrupo do tipo *com* entre ρ_{server} e $\rho_{cashier}$ e uma intergrupo do mesmo tipo entre ρ_{server} e ρ_{cook} . No grupo gt_{tables} pode-se ver as relações de compatibilidade entre os papéis ρ_{chef} , $\rho_{sous-chef}$ com o papel ρ_{cook} e em ambos os grupos é possível ver a cardinalidade de cada papel:

$$\begin{aligned}
 gt_{tables} = (& \\
 & \{\rho_{server}, \rho_{cashier}\}, & \%R \\
 & \{\}, & \%SG \\
 & \{link(\rho_{server}, \rho_{cashier}, com)\}, & \%L^{intra} \\
 & \{link(\rho_{server}, \rho_{cook}, com)\}, & \%L^{inter} \\
 & \{\}, & \%C^{intra} \\
 & \{\}, & \%C^{inter} \\
 & \{\rho_{server} \mapsto (1, 5), \rho_{cashier} \mapsto (1, 1)\}, & \%np \\
 & \{\} & \%ng \\
 gt_{kitchen} = (& \\
 & \{\rho_{chef}, \rho_{sous-chef}\}, & \%R \\
 & \{\}, & \%SG \\
 & \{link(\rho_{chef}, \rho_{sous-chef}, aut), link(\rho_{sous-chef}, \rho_{chef}, com)\}, & \%L^{intra} \\
 & \{\}, & \%L^{inter} \\
 & \{\rho_{chef} \bowtie \rho_{cook}, \rho_{sous-chef} \bowtie \rho_{cook}\}, & \%C^{intra} \\
 & \{\}, & \%C^{inter} \\
 & \{\rho_{chef} \mapsto (1, 1), \rho_{sous-chef} \mapsto (1, 1)\}, & \%np \\
 & \{\} & \%ng
 \end{aligned}$$

Especificação Funcional

A Figura 3 mostra um dos esquemas sociais que compõem a Especificação Funcional de um restaurante. Nesse esquema, o atendimento de uma mesa é o objetivo global, sendo decomposto em passos necessários para atingi-lo (como receber o pedido e prepará-lo) e associado a missões.

O esquema social $sch_{costumer}$ mostra formalmente os elementos do esquema apresentado na Figura 3. Ele é composto das metas *serveCostumer*, *takeOrder*, *sendToKitchen*, *getOrder*, *serveDrinks*, *prepareFood*, *sendToSaloon* e *serveFood*. As metas *serveCostumer* e *getOrder* tem planos associados para atingi-las, sendo que todas as outras metas representadas como folhas do esquema. O esquema define também as missões *m1*, *m2*, *m3* e *m4*,

Figura 3: Exemplo de uma especificação funcional para um restaurante.

Fonte: Autor.

e seus respectivos mapeamentos para as metas $\{takeOrder, sendToKitchen, serveDrinks\}$, $sendToSaloon$, $serveFood$ e $prepareFood$ do esquema. Por fim, pode-se ver respectivamente a quantidade mínima e máxima de agentes para cada uma das missões, como $m1 \mapsto (1, 5)$. Tais definições indicam, por exemplo, que no caso de $m1$ um conjunto de um a cinco agentes podem se comprometer a atingir em sequencia os objetivos $takeOrder$, $sendToKitchen$ e $serveDrinks$.

```

schcostumer =    ({serveCostumer, takeOrder, sendToKitchen,
getOrder, serveDrinks, prepareFood, sendToSaloon, serveFood});           %G
                {"serveCostumer = takeOrder, sendToKitchen, getOrder,
sendToSaloon, serveFood"
                "getOrder = serveDrinks || prepareFood"};               %P
                {m1, m2, m3, m4};                                         %M
                {m1  $\mapsto$  {takeOrder, sendToKitchen, serveDrinks},
m2  $\mapsto$  {sendToSaloon}, m3  $\mapsto$  {serveFood},
m4  $\mapsto$  {prepareFood}};                                                  %mo
                {m1  $\mapsto$  (1, 5), m2  $\mapsto$  (1, 2), m3  $\mapsto$  (1, 5), m4  $\mapsto$  (1, 2)}; ) %nm

```

Especificação Deontica

Tabela 1: Exemplo de uma especificação deontica para um restaurante.

Role	Mission	Link
server	m1	obligation
cook	m2	obligation
server	m3	permission
cook	m4	obligation

Fonte: Autor.

A Tabela 1 mostra a especificação deontica para a organização do restaurante. Nela é possível ver que os agentes no papel de *server* tem obrigação de cumprir a missão *m1* e permissão de cumprir a missão *m3*. Já os agentes no papel de *cook* (e consequentemente dos papéis que herdam suas propriedades) tem obrigação de cumprir as missões *m2* e *m4*.

Como o objetivo deste trabalho é incorporar a noção de autonomia de planejamento em modelos organizacionais de agentes, na sequência serão apresentados brevemente os principais conceitos de planejamento que foram utilizados para esta finalidade.

2.4 Planejamento

O problema de planejar consiste em encontrar as ações que conduzem um estado inicial a um estado final [RUSSELL; NORVIG, 2009]. Essa área foi primeiramente conhecida pela linguagem/planejador STRIPS [FIKES; NILSSON, 1971]. Tipicamente, os algoritmos de planejamento são baseados em buscas em espaços de estados.

A Figura 4 mostra um exemplo de funcionamento de um planejador. A partir de um estado inicial S_0 , as possíveis ações A_0 são as transições para os próximos estados S_1 , e esses últimos são o resultado das respectivas ações aplicadas em S_0 . Um plano é o conjunto de ações que levam do estado inicial para um estado desejado, sendo possível existirem planos ordenados (como na figura) ou planos de ordem parcial.

Figura 4: Como funciona a busca em espaço de estados para planejamento.

Fonte: Autor.

2.4.1 Tipos de Planejamento

Através dos anos foram desenvolvidos diversos formalismos e linguagens para representar diferentes problemas que foram sendo encontrados na área. Dentre eles podem-se citar planejamento clássico, planejamento probabilístico, planejamento temporal e planejamento multiagente.

O planejamento clássico consiste em “encontrar uma sequência de operadores em um espaço de modelos do mundo que transformam um modelo do mundo inicial em um modelo onde uma dada fórmula-objetivo pode ser provada verdadeira” [FIKES; NILSSON, 1971]. Ele surgiu primeiramente como uma maneira de se provar teoremas e evoluiu para ser utilizado nas mais diversas áreas. O planejamento probabilístico [KUSHMERICK; HANKS; WELD, 1995] surgiu da necessidade de se representar as imprecisões existentes em muitos sistemas reais e considerá-las durante o processo de planejamento, resultando na utilização de modelos probabilísticos para se definir como as ações modificam o mundo. Já o planejamento temporal [ALLEN, 1991] inclui conceitos de raciocínio temporal no processo de planejamento, fazendo as ações terem uma duração para sua execução e o tempo do ambiente sendo contínuo, em oposição ao tempo discreto usado em planejamento clássico. No planejamento multiagente, por sua vez, existem diversos agentes alterando o ambiente simultaneamente, podendo ou não agirem em conjunto, e potencialmente interferindo nas ações uns dos outros.

Neste trabalho, será utilizado apenas o planejamento hierárquico.

2.4.2 Planejamento Hierárquico

O conceito de planejamento hierárquico (HTN) foi introduzido em 1975 pelo planejador NOAH (Net of Action Hierarchies) e é estudado até hoje pelo nome de planejamento HTN [EROL; HENDLER; NAU, 1995]. No começo, HTN era mais utilizado como uma técnica ad-hoc, ou um “hack eficiente”, sobre o estilo tradicional de planejamento STRIPS do que um formalismo. Entretanto, no início dos anos 90 alguns trabalhos começaram a incluir definições formais para HTN e analisar sua expressividade, comparando-o a STRIPS. Um desses trabalhos [EROL; HENDLER; NAU, 1995] concluiu que é possível transformar um domínio em STRIPS para um domínio em HTN em um baixo tempo polinomial mas que o inverso não é possível, principalmente por que a noção de objetivo em STRIPS é mais restritiva e não é possível de se expressar restrições em estados intermediários, enquanto que HTN permite que essas restrições existam através de redes de

tarefas.

Visão Geral

As referências principais para a descrição de HTN usadas nesse trabalho são propostas em [EROL; HENDLER; NAU, 1995] e [GEORGIEVSKI; AIELLO, 2015]. Um típico problema HTN consiste em expressar como uma dada tarefa-objetivo pode ser decomposta em tarefas primitivas. *Tarefas primitivas* são definidas como ações em STRIPS: uma transição imediata de estado que consiste no mapeamento parcial de um conjunto de estados para outros. *Tarefas não primitivas*, que são chamadas de tarefas compostas, são aquelas que não podem ser executadas diretamente porque representam uma atividade abstrata ou complexa, e por isso precisam ser decompostas em tarefas primitivas para serem executadas. Essas decomposições podem variar porque para a maioria das atividades complexas, há diferentes maneiras de decompô-las, expressas por redes de tarefas: uma série de tarefas, incluindo primitivas e compostas, que precisam ser completadas seguindo certas restrições. O mapeamento de tarefas compostas para redes de tarefas é chamado de *método*. Uma mesma tarefa composta pode ter diferentes métodos. O algoritmo de planejamento é normalmente inicializado com a tarefa principal que é decomposta recursivamente através dos métodos até que restem apenas tarefas primitivas no plano [KOVACS, 2012].

A seguir são apresentados o modelo formal e os conceitos de HTN.

Elementos

Os elementos básicos do formalismo HTN são:

- *Operador*: Um operador pode ser definido como uma ação atômica, uma transição de estado ou o mapeamento parcial de um conjunto de estados para outro.
- *Tarefa*: Uma tarefa é o elemento principal de planejamento HTN e é usada para representar uma atividade. Assim como em outros contextos, uma atividade pode ser simples ou complexa. O formalismo HTN diferencia essas atividades nomeando-as primitivas ou complexas.
- *Rede de Tarefas*: Uma rede de tarefas define a maneira que um conjunto de tarefas precisa ser executado.

- *Método*: Tendo em vista que uma rede de tarefas é composta de um conjunto de tarefas com restrições, o método associa essas tarefas a uma tarefa complexa. Sendo assim, um método especifica como uma tarefa complexa pode ser decomposta em tarefas menos complexas.
- *Domínio de Planejamento*: Um domínio de planejamento HTN consiste basicamente de todas as tarefas (primitivas e complexas) e métodos que dizem como decompô-las.
- *Problema de Planejamento*: Um problema de planejamento HTN é composto de um estado inicial, uma tarefa-alvo e um domínio. Para resolver um problema, parte-se do estado inicial, utilizam-se as tarefas e suas decomposições presentes no domínio para se decompor a tarefa-alvo até que todas as tarefas-folhas da rede sejam primitivas.

Cada um destes elementos será detalhado formalmente na sequência.

Descrição Formal

Operador: Um operador o pertencente a um conjunto de operadores O é o par $(pre(o), eff(o))$ onde:

$pre(o)$ é o conjunto com as pré-condições para esse operador ser aplicado, sendo que estas devem ser consequências lógicas do estado do mundo atual;

$eff(o)$ são os efeitos do operador. Efeitos são as proposições lógicas que serão adicionadas ou removidas do estado do mundo atual após o operador ser aplicado.

Tarefa: Uma tarefa t de um conjunto de tarefas T é apenas um simbolo ou identificador.

Uma tarefa é considerada primitiva se $t \in O$. Caso contrário, ela é considerada complexa. Denota-se por T_p o subconjunto de T onde todos os elementos são tarefas primitivas. Logo, $T_p = O$. Denota-se como T_c o subconjunto de T onde todos os elementos são tarefas compostas, tal que $T_c = T - T_p$. Note que tarefas primitivas são definidas apenas por um simples operador, que conceitualmente significa que ela pode ser completada através da execução de uma ação.

Rede de Tarefas: Uma rede de tarefas tn é definida como um par $(ts(tn), \psi)$ onde:

$ts(tn)$ é um conjunto de tarefas tal que $ts(tn) \subseteq T$;

ψ é um conjunto de restrições sobre $ts(tn)$.

As restrições ψ são comumente utilizadas como restrições lógicas ou de ordem, e são aplicadas quando a rede de tarefas é decomposta.

- precedência ($t_0 \prec t_1$) significa que a tarefa t_0 precisa ter sua execução precedente à tarefa t_1 ;
- sucessão ($t_0 \succ t_1$) significa que a tarefa t_0 precisa suceder t_1 ;
- logica (t_0, t_1, t_2) significa que a tarefa t_1 precisa ser verdade (ou completada) antes de t_2 e depois de t_0 .

Método: Um método m é um elemento de um conjunto de métodos M que é definido como o par $(ct(m), tn(m))$ onde:

$ct(m)$ é uma tarefa composta pertencente a T_c ;

$tn(m)$ é uma rede de tarefas.

Domínio de Planejamento: Um domínio de planejamento D é composto de (T, M) onde:

T é um conjunto de tarefas;

M é um conjunto de métodos.

Problema de Planejamento: Um problema de planejamento P é definido como uma tripla (D, S_0, T_g) onde:

D é um Domínio de planejamento;

S_0 é o estado inicial do mundo;

T_g é a tarefa-objetivo inicial que deve ser decomposta.

Um problema HTN consiste em decompor T_g através dos métodos presentes em D que podem ser utilizados dado o estado do mundo S_0 . Algoritmos de planejamento HTN têm como entrada um problema de planejamento HTN e tem como saída a sequência de operadores (ou tarefas primitivas) pertencentes a $D(T)$ com restrições associadas às redes de tarefas presentes em $D(M)$, que quando aplicadas em S_0 resultam em T_g . Alguns formalismos HTN apresentam T_g como a rede de tarefas que se deseja alcançar, mas aqui será utilizada apenas a representação de uma tarefa, uma vez que ela pode conter várias redes que a decompõe.

2.4.3 Exemplo: Domínio de Planejamento de um Restaurante

O Domínio 1 ilustra os elementos utilizados para o planejamento de um restaurante no formalismo HTN. Como não existe uma linguagem padrão para HTN, foi utilizada a linguagem pelo planejador SHOP [NAU et al., 1999]. No algoritmo é possível ver que existem quatro operadores: *get – object*, *carry-to*, *place-order* e *cook-food*; e dois métodos: *serve-food* e *serve-drinks*. No formato SHOP, a primeira lista de elementos após o nome do operador são as precondições, seguida dos efeitos negativos do operador e em seguida seus efeitos positivos.

Domínio 1: Exemplo de um domínio de restaurante usando no formato HTN usado pelo algoritmo SHOP

```

1: (defdomain restaurant
2: (
3: (:operator (!get-object ?OBJECT ?PLACE)
4:   ((at ?OBJECT ?PLACE))
5:   ((at ?OBJECT ?PLACE))
6:   ((carrying ?OBJECT))
7: )
8: (:operator (!carry-to ?OBJECT ?PLACE)
9:   ((carrying ?OBJECT))
10:  ((carrying ?OBJECT))
11:  ((at ?OBJECT ?PLACE))
12: )
13: (:operator (!place-order ?MEAL ?TABLE )
14:   ((menu ?MEAL))
15:   ()
16:   ((placed-order ?MEAL ?TABLE))
17: )
18: (:operator (!cook-food ?FOOD ?UTENSIL)
19:   ((has-stock ?FOOD) (in-kitchen ?UTENSIL) (know-cook ?UTENSIL ?MEAL))
20:   ((has-stock ?FOOD))
21:   ((food-prepared ?FOOD ?MEAL))
22: )
23: (:method (serve-food ?TABLE ?MEAL)
24:   ((placed-order ?MEAL ?TABLE) (food-prepared ?FOOD ?MEAL) (at ?RECIPE
    balcon))
25:   ((!get-object ?MEAL balcon) (!carry-to ?MEAL ?TABLE))
26: )
27: (:method (serve-drinks ?TABLE ?DRINK)
28:   ((has-stock ?DRINK) (placed-order ?DRINK ?TABLE))
29:   ((!get-object ?DRINK fridge) (!carry-to ?FOOD ?TABLE))
30: ))

```

2.5 Tipos de Objetivo

É importante mencionar que existem diferentes tipos de objetivos em problemas de Inteligência Artificial e de Planejamento. Riemsdijk, Dastani e Winikoff [2008] apresentam uma taxonomia desses tipos que pode ser vista na Figura 5.

Figura 5: Taxonomia dos tipos de objetivo.

Fonte: [RIEMSDIJK; DASTANI; WINIKOFF, 2008].

A Figura 5 mostra que os tipos de objetivos podem ser representados em uma árvore, sendo a primeira e mais importante a ramificação que diferencia entre objetivos declarativos e objetivos procedurais. Objetivos declarativos são relacionados ao resultado de ações. De uma maneira mais precisa, um objetivo declarativo é estritamente ligado a estados do mundo. Objetivos declarativos podem ser ainda divididos entre objetivos de estado único ou estados múltiplos. Além disso, os tipos representados nas folhas da árvore consistem em: query (consulta), achievement (realização) e maintenance (manutenção).

Por outro lado, objetivos procedurais são relacionados à execução de ações. São completados quando uma ação é executada com sucesso.

Figura 6: Redução do espaço de estados utilizando planejamento hierárquico com múltiplos agentes.

Fonte: [HAYASHI, 2007].

2.6 Trabalhos Relacionados

2.6.1 Organizações e Planejamento

Apesar de organizações e planejamento serem temas muito estudados, não existe uma extensa literatura de trabalhos que relacionam o conceito de organização com o de planejamento. Abaixo serão apresentados alguns dos trabalhos encontrados.

A Figura 6 mostra a abordagem de [HAYASHI, 2007] onde foi utilizada uma abordagem estratificada para planejamento multiagentes (SMAP). Na abordagem, existe uma relação hierárquica entre os agentes. Essa relação permite que um agente em um ponto mais alto da hierarquia possa criar planos mais gerais, enquanto os agentes de nível mais baixo podem criar planos para os subobjetivos do primeiro.

Apesar de não existir uma definição de organização no trabalho apresentado, é possível notar que ela existe mesmo assim, uma vez que existem uma hierarquia de papéis e funções entre os agentes (agentes em posição de criar planos mais gerais e agentes responsáveis por planejar para subobjetivos em diferentes granularidades).

Já o trabalho de [CARDOSO, 2018] integra a organização como elemento presente na hora de planejar, como mostra a Figura 7. O arcabouço, assim como o de [HAYASHI, 2007], permite aos agentes realizarem planejamento individual, mas nesse caso eles possuem diferentes mecanismos de coordenação em diferentes etapas do sistema. Ao contrário da abordagem estratificada, na abordagem de [HAYASHI, 2007] a organização é parte integral do sistema e é considerada na hora de coordenar os agentes.

Figura 7: Funcionamento do arcabouço proposto para planejamento descentralizado.

Fonte: [CARDOSO, 2018].

2.6.2 Organizações e Autonomia

Como já mencionado anteriormente na Seção 2.1, autonomia é um conceito relativo e depende da arquitetura do sistema. Em um SMA onde existe uma organização, pode-se falar sobre a relação de autonomia entre os agentes e a organização. Sendo a organização vista como um conjunto de normas para promover a cooperação entre agentes, esse tipo de autonomia se encaixa no conceito definido anteriormente como Autonomia de Normas. Existem trabalhos anteriores que estudam não diretamente a autonomia em relação às normas, mas as consequências dessa autonomia. Como exemplo pode-se citar [BOELLA; TORRE, 2004] e [NARDIN et al., 2016] que propõem maneiras de monitorar seu cumprimento e recompensar/punir os agentes quando elas são cumpridas/infringidas. Sanções são uma maneira de incentivar o comportamento em acordo com as normas uma vez que os agentes do sistema possuam autonomia para infringi-las. Apesar delas não restringirem a autonomia do agente, sua presença indica um sistema onde é previsto que os agentes sejam autônomos para escolher seguir as normas ou não.

Este trabalho tem uma abordagem similar aos apresentados anteriormente, entretanto partindo de um foco mais centrado na organização. Também realiza uma ligação entre os conceitos de planejamento e organização, usando a definição de autonomia de tais conceitos.

3 AUTONOMIA DE PLANEJAMENTO EM ORGANIZAÇÕES

3.1 Especificações de Domínio

Uma Especificação de Domínio (EDom) é um modelo para representar o domínio onde os agentes da organização irão atuar. A EDom define uma única definição de objetivos e ações, independente dos agentes e/ou do ambiente, potencialmente simplificando o ambiente caso ele contenha mais informações do que o necessário quando se projeta a organização, e garantindo que os objetivos da organização sejam os mesmos, não importando qual agente está tentando cumpri-los. Conceitualmente, as ações e objetivos definidos na EDom são, respectivamente, os objetivos do tipo procedural e declarativo, sendo as ações uma representação dos processos definidos que podem ser encontrados em organizações reais e os objetivos como o conjunto de metas que a organização quer cumprir.

Conceitualmente, a EDom é independente do modelo organizacional utilizado pelos agentes. Existe, no entanto, uma pré-condição: tais modelos devem ter uma representação explícita dos *objetivos* da organização.

Os objetivos do tipo declarativos são objetivos que procuram alcançar um estado definido, atribuindo ao agente que tenta cumpri-lo autonomia para criar seus próprios planos.

Já os objetivos do tipo procedural referem-se à execução de uma ação específica, não dando liberdade ao agente para escolher como fazê-lo, uma vez que se trata de uma ação atômica. A função ω proposta no modelo formal é utilizada para fazer a distinção entre um objetivo procedural ou declarativo.

Para os objetivos declarativos, também é proposta uma maneira de traduzi-los para os diferentes formalismos de planejamento, permitindo assim que agentes capazes de realizar planejamento tentem atingir esses objetivos resultando em um mesmo estado final.

3.2 Modelo Formal

O modelo formal de uma EDom utiliza a noção de *predicado lógico*. Um predicado é uma relação n -ária entre um ou vários termos, que podem ser variáveis ou constantes¹. Formalmente, seja $Pred = \{pred_1, \dots, pred_x, \dots, pred_{N_{pred}}\}$ um conjunto de predicados onde cada predicado $pred_x = (f_x, V_x)$ tal que $f_x \in F_{pred}$ é o nome do predicado, e $V_x \subset V_{pred} \cup C_{pred}$ é o conjunto de termos do predicado, que podem ser variáveis ou constantes.

A título de ilustração, para permitir uma melhor compreensão do modelo, serão utilizados a organização e domínio de um restaurante apresentados respectivamente nas Seções 2.3.3 e 2.4.3. Nestes exemplos, as constantes e predicados serão representadas por letras minúsculas e as variáveis por letras maiúsculas.

Seja M_{org} um modelo organizacional e F_{plan} um formalismo de planejamento, sendo M_{org} um modelo organizacional que possui pelo menos uma definição de objetivos. Uma Especificação de Domínio é definida como uma 9-tupla:

$$DomS = \{Og, P, Pp, A, \theta, G, \omega, Ag, \lambda\}$$

onde:

- $Og = \{og_1, \dots, og_x, \dots, og_{N_{og}}\}$ é um conjunto de objetivos organizacionais tal que $Og \subset M_{org}$ ² e $Og \subset F_{pred}$. Os objetivos são representados por nomes de predicados onde cada um serve como um identificador para um objetivo de M_{org} . Os objetivos organizacionais da EDom são apenas um subconjunto de todos os objetivos da organização, uma vez que podem existir objetivos intermediários que necessitam de ações organizacionais e/ou objetivos que são apenas consequência lógica de outros. Como exemplo, na organização utilizada na Seção 2.3.3, teríamos $og_1 = takeOrder$, $og_2 = sendToKitchen$, $og_3 = serveDrinks$, $og_4 = prepareFood$, $og_5 = sendToSaloon$ e $og_6 = serveFood$.
- $P = \{p_1, \dots, p_x, \dots, p_{N_p}\}$ é um conjunto de predicados do mundo, onde cada elemento $p_x = (h, a)$ é um predicado relevante para o domínio, sendo $h \in F_{pred}$ o nome do predicado e a a aridade do predicado. Em muitas vezes, o ambiente onde os agentes estão situados é composto de muitos estados, entretanto nem todos são

¹Para efeito deste trabalho, não estamos considerando a existência de símbolos funcionais.

²Como a representação formal dos diversos modelos organizacionais é distinta, adotou-se aqui um abuso de notação. Considera-se que qualquer M_{org} tenha em sua representação um subconjunto Og de objetivos organizacionais.

necessários para se alcançar os objetivos da organização. Como exemplo, no domínio do restaurante utilizado na Seção 2.4.3, teríamos $p_1 = at/2$, $p_2 = carrying/1$, $p_3 = menu/1$, $p_4 = placed-order/2$, $p_5 = has-stock/1$, $p_6 = in-kitchen/1$, $p_7 = know-cook/2$, $p_8 = food-prepared/2$.

- $Pp = \{pp_1, \dots, pp_x, \dots, pp_{N_{pp}}\}$ é um conjunto de problemas de planejamento, onde cada problema de planejamento $pp_x = \{I_x, F_x\}$, sendo $I_x \subset P$ um conjunto de estados iniciais e $F_x \subset P$ um conjunto de estados finais. Um problema de planejamento é um dos elementos necessários para se executar os algoritmos de planejamento. No mesmo domínio do restaurante usado anteriormente, poderíamos definir o problema de servir um peixe grelhado (*grilled-fish*) na *mesa2*. Tal problema teria a seguinte representação formal: $pp_1 = (\{menu(grilled-fish), has-stock(fish), know-cook(grill, grilled-fish), in-kitchen(grill), placed-order(grilled-fish, table2)\}, \{at(grilled-fish, table2)\})$.
- $A = \{a_1, \dots, a_x, \dots, a_{N_a}\}$ é um conjunto de ações do domínio, onde $a_x \subset P$. Essas ações representam os processos que a organização considera importantes que sejam cumpridos sem modificações em suas precondições ou seus efeitos, não importando qual agente os execute. No domínio do restaurante utilizado na Seção 2.4.3, teríamos $a_1 = get-object$, $a_2 = carry-to$, $a_3 = place-order$, $a_4 = cook-food$.
- $\theta : A \mapsto C \times E^+ \times E^-$ é uma função que mapeia uma ação a_x em uma tripla de conjuntos de predicados, $\theta(a_x) = (C_x, E_x^+, E_x^-)$, sendo $C_x \subset P$ o conjunto de precondições e $E_x^+ \subset P$ e $E_x^- \subset P$ respectivamente os predicados a serem adicionados ou removidos do mundo como efeito da ação. No domínio do restaurante utilizado na Seção 2.4.3, teríamos $\theta(a_1) = (\{at(OBJECT, PLACE)\}, \{carrying(OBJECT)\}, \{at(OBJECT, PLACE)\})$.
- $G = \{g_1, \dots, g_x, \dots, g_{N_g}\}$ é um conjunto de objetivos do domínio, onde $g_x \subset P$. Cada objetivo do domínio é a conjunção de um ou mais estados do mundo que representam situações que a organização deseja que estejam presentes para que seu objetivo global seja alcançado. Esses objetivos precisam ser definidos pelo projetista. Utilizando o domínio do restaurante, pode-se declarar um objetivo $g_6 = at(MEAL, TABLE)$.
- $\omega : Og \mapsto A \cup G$ é uma função que mapeia um objetivo organizacional para um predicado que representa ação ou objetivo de domínio. A função ω é usada para diferenciar entre os diferentes tipos de objetivos que podem estar presentes. Um objetivo organizacional og_x com $\omega(og_x) \in A$ é um objetivo do tipo procedural, uma vez que ele representa uma ação que deve ser executada, e um objetivo organizacional

com $\omega(og_x) \in G$ é um objetivo do tipo declarativo, já que ele representa estados que são desejados pela organização. Usando os exemplos anteriores, poderíamos ter $\omega(og_6) = g_6$, ou $\omega(serveFood) = at(MEAL, TABLE)$.

- $Ag = \{ag_1, \dots, ag_x, \dots, ag_{N_{ag}}\}$ é um conjunto de agentes, onde cada agente $ag_x = \{BB_x, Garg_x\}$, sendo BB_x a base de conhecimento do agente e $Garg_x$ um conjunto de triplas (og_x, v_x, c_x) , onde $og_x \in Og$; $v_x \in V_x$ sendo V_x as variáveis de $\omega(og_x)$ e c_x um valor constante a ser atribuído a esta variável. Como exemplo, seja o agente $ag_1 = \{BB_1, Garg_1\}$ tal que $BB_1 = \{menu(grilled-fish), has-stock(fish), know-cook(grill, grilled-fish), in-kitchen(grill), placed-order(grilled-fish, table2)\}$ e $Garg_1 = \{(serveFood, MEAL, grilled-fish), (serveFood, TABLE, table2)\}$. Enquanto um agente é composto de diferentes elementos, para a especificação de domínio são necessários apenas a base de conhecimento e um conjunto de triplas que associa um objetivo organizacional e uma variável de seu mapeamento para o domínio à um valor constante. Esse conjunto de triplas é utilizado para associar os valores constantes às variáveis de $\omega(og_x)$. Do ponto de vista de um agente no ambiente, apenas og_x é conhecido, logo é importante utilizar og_x no conjunto $Garg_x$, ao invés de utilizar diretamente $\omega(og_x)$.
- $\lambda : Ag \times G \rightarrow Pp$ é uma função que, para um dado agente ag_x e objetivo de domínio g_y , resulta em um problema de planejamento $p_z = \{I_z, F_z\}$, onde $I_z = BB_x \cap P$ e F_z é o resultado da unificação lógica das variáveis do objetivo g_y com as constantes de $Garg_x$ onde $og_x = \omega^{-1}(g_y)$. Como exemplo, seja o conjunto $Garg_1$ do agente ag_1 definido anteriormente. Esse agente, ao tentar atingir o objetivo do domínio g_6 (também definido anteriormente), unifica as variáveis $MEAL$ e $TABLE$ de $at(MEAL, TABLE)$ respectivamente com os valores $grilled-fish$ e $table2$, uma vez que $\omega^{-1}(g_6) = serveFood$ e $Garg_1 = \{(serveFood, MEAL, grilled-fish), (serveFood, TABLE, table2)\}$. A função λ é utilizada para se obter um problema de planejamento baseado no conhecimento do agente e no objetivo da organização. O estado inicial é composto apenas de conhecimentos do agente que são relevantes para o domínio, já o estado final é formado pelo objetivo do domínio com suas variáveis possuindo os valores definidos no conjunto $Garg$ do agente. Utilizando como exemplo o agente, objetivo e problema de planejamento definidos anteriormente, temos $\lambda(ag_1, g_6) = pp_1$, cujo estado final desejado é $\{at(grilled-fish, table2)\}$.

A Figura 8 mostra como é possível obter os elementos da **EDom** através dos elementos

Figura 8: Relação entre os elementos de um SMA e os elementos da Especificação de Domínio.

Fonte: Autor.

de um SMA. O conjunto Og é formado pelos objetivos da organização do SMA. Os conjuntos E , A e O são formados a partir da representação do ambiente como um domínio que o projetista faz. A função ω associa elementos do conjunto Og a elementos dos conjuntos A e O . O conjunto Ag é formado pelos agentes presentes no ambiente e o conjunto Pp pode ser obtido pela função λ .

A **EDom** é uma especificação para que o projetista da organização insira o conhecimento que ele possui do domínio. Isso permite que, mesmo que não exista uma ligação entre a organização e o ambiente, ele possa fornecer aos agentes dados sobre situações que ele prevê. O fato de existirem objetivos para os quais os agentes podem ou não ter autonomia permite ao projetista dar liberdade (ou retirá-la) aos agentes para se adaptarem (ou serem forçados a agir de uma determinada maneira) sem infringir as normas da organização, mas ainda assim atingindo os resultados esperados pelo projetista.

Note que assim como o agente pode ter autonomia-normativa e optar por não seguir as normas da organização, o agente também pode optar por não seguir as definições da **EDom**, uma vez que elas não são normas regimentadas.

3.3 Uso de Especificação de Domínio

A EDom é utilizada pelos agentes para se obter o tipo de um objetivo organizacional (declarativo ou procedural) assim como a sua definição, permitindo ao agente ter ou não autonomia para esse objetivo. Cada um dos diferentes tipos de objetivos possui maneiras específicas para que os agentes os alcancem. Essas maneiras serão descritas a seguir.

3.3.1 Objetivos Declarativos

Um agente ag , ao participar de uma organização, irá executar os seguintes passos:

1. ag irá escolher qual objetivo organizacional og_x deseja alcançar;
2. Através da EDom, ele obtém o objetivo de domínio $g_x = \omega(og_x)$;
3. Tendo $Garg_x = (og_x, v_x, c_x)$, o agente associa os valores c_x às variáveis v_x de um objetivo de domínio g_x , onde $og_x = \omega^{-1}(g_x)$, obtendo-se assim um g_x concreto com valores constantes em suas variáveis;
4. Em seguida, ag cria o problema de planejamento $pp_x = \lambda(ag, g_x)$;
5. Com o problema de planejamento criado, ag pode utilizar algoritmos de planejamento para buscar uma solução;
6. ag executa a solução do problema.

3.3.2 Objetivos Procedurais

Para alcançar objetivos procedurais, como o agente não ter autonomia de planejamento, não é necessário criar um problema de planejamento ou plano. Assim, um agente ag , ao participar de uma organização, irá executar os seguintes passos:

1. ag irá escolher qual objetivo organizacional og_x deseja alcançar;
2. Através da EDom, ele obtém o objetivo de domínio $g_x = \omega(og_x)$;
3. Tendo $Garg_x = (og_x, v_x, c_x)$, o agente associa os valores c_x às variáveis v_x de um objetivo de domínio g_x , onde $og_x = \omega^{-1}(g_x)$, obtendo-se assim um g_x concreto com valores constantes em suas variáveis;

4. Uma vez que todas as variáveis da ação possuem valores constantes, ag executa a ação.

4 IMPLEMENTAÇÃO

Nesse capítulo será mostrada como foi feita a implementação da EDom na plataforma JaCaMo[BOISSIER et al., 2013].

4.1 JaCaMo

JaCaMo é uma plataforma de desenvolvimento multiagente onde é possível utilizar três paradigmas para a programação multiagente, cada um deles centralizado em uma abstração de primeira classe diferente: na organização, no ambiente ou nos agentes. Para isso, a plataforma JaCaMo utiliza três diferentes arcabouços: **Jason** [BORDINI; HÜBNER, 2005] para agentes, **CArtAgO** [RICCI; VIROLI; OMICINI, 2006] para ambiente e **MOISE** [HÜBNER; SICHMAN; BOISSIER, 2002] para organizações. Cada um desses arcabouços é responsável por definir uma dimensão de um SMA, utilizando-se do seu respectivo paradigma para a programação.

4.1.1 Jason

Jason é a linguagem utilizada para a dimensão de agentes. Ela é uma extensão da linguagem AgentSpeak(L), a qual se baseia na arquitetura BDI (*belief-desire-intention*). A arquitetura BDI é baseada no processo mental que humanos usam. Ela é composta de *beliefs*, que representam as informações que o agente tem sobre o mundo e outros agentes; *desires*, que representam objetivos que o agente deseja alcançar; e *intentions*, que representam o que o agente quer fazer em um determinado momento. Agentes programados em **Jason** atuam no ambiente através da execução dos planos em sua biblioteca e percebem o ambiente através da adição ou remoção de elementos em sua base conhecimentos.

O Algoritmo 2 mostra o exemplo do código de um agente **Jason**. Nela os elementos nas linhas 1 a 3 são a base de crenças do agente. Já o elemento da linha 5 é o ativador do plano, ele representa o plano que será executado quando o agente tiver uma determinada

Algorithm 2: Exemplo de um agente em Jason

```

1: menu(grilled-chicken).
2: my_table(table1)
3:
4: + !order-food(Food)
5:   :   menu(Food) & my_table(Table)
6:   < - + placed-order(Food, Table);

```

Figura 9: Exemplos de artefato.

Fonte: Autor.

intenção; na linha 6 são indicadas as precondições para se executar o plano e na linha 7 e 8 é indicado o corpo do plano que será executado caso as precondições existam.

Uma outra funcionalidade importante, utilizada neste trabalho, é a possibilidade do usuário criar *ações internas* que estendam a funcionalidade original da linguagem.

4.1.2 CArtAgO

CArtAgO é o arcabouço utilizado para a dimensão de ambiente. Ele é utilizado para a programação de ambientes baseado no modelo A&A (Agents and Artifacts) [OMICINI; RICCI; VIROLI, 2008]. Esse modelo separa os elementos do ambiente em duas entidades: agentes e artefatos. Agentes atuam no ambiente através de operações em artefatos e percebem o mesmo através das propriedades observáveis.

Na Figura 9 (esquerda) pode-se ver como um artefato genericamente disponibiliza duas operações e duas propriedades observáveis. Na direita, temos um exemplo de um artefato **Oven** com suas duas propriedades observáveis (*is_cooking* e *is_clean*) e duas operações (*cook* e *boil*).

4.1.3 MOISE

MOISE [HÜBNER, 2003] é o modelo organizacional utilizado para a dimensão da organização. Através dele, é possível especificar uma organização em três dimensões diferentes: estrutural, funcional e deontica. Uma explicação mais detalhada pode ser encontrada na Seção 2.3 deste trabalho.

Dentro da plataforma JaCaMo, a implementação do modelo MOISE [HÜBNER, 2003] se denomina ORA4MAS [KITIO et al., 2008]. Ela define três artefatos para a organização: ORG-BOARD, GROUP-BOARD e SCHEME-BOARD. Cada uma desses artefatos contem as propriedades observáveis que descrevem, respectivamente, os elementos da organização, dos grupos e dos esquemas sociais. Através de operações nesses artefatos, os agentes interagem com a organização quando desejam adotar papéis, participar de um grupo, obter os objetivos disponíveis do esquema social ou obter informações sobre quais grupos estão completos.

4.1.4 Integração das Ferramentas

A plataforma JaCaMo integra todas as ferramentas de maneira bem fluida. As propriedades observáveis de um artefato estão disponíveis como crenças para um agente quando este foca no artefato, sendo estes conhecimentos alterados automaticamente quando a propriedade observável também for alterada. Os agentes executam operações nos artefatos através de predicados no corpo de seus planos, sendo possível interagir com diversos artefatos em um mesmo plano.

A Figura 10 [BOISSIER et al., 2013] mostra como as diferentes dimensões da programação de um SMA estão relacionadas na plataforma JaCaMo. No topo está a especificação de uma organização MOISE [HÜBNER, 2003]. Essa dimensão tem ligação com a dimensão dos agentes, uma vez que estes participam da organização. As relações entre a dimensão dos agentes e a dimensão do ambiente se dão por conta deles executarem ações (operações nos artefatos) e perceberem este último (propriedades observáveis dos artefatos). Vale notar que é possível existir uma ponte entre a dimensão do ambiente na plataforma e um ambiente externo (seja virtual ou real), uma vez que o ambiente especificado na plataforma é apenas uma maneira de se representar o domínio de aplicação.

No âmbito deste trabalho, a EDom foi implementada por meio de um artefato em CArtaGo e uma série de ações internas em Jason. Os detalhes de implementação são apresentados a seguir.

Figura 10: Visão geral da plataforma JaCaMo.

Fonte: [BOISSIER et al., 2013].

4.2 Artefatos

4.2.1 Especificação de Domínio (DomainBoard)

A implementação da especificação de domínio foi feita através de um artefato chamado DomainBoard. O DomainBoard é construído passando como parâmetro um arquivo que contém as definições de parte do modelo da EDom, mais especificamente dos conjuntos A , P e G . O arquivo utilizado encontra-se no formato .asl e foi utilizado o parser do Jason para sua leitura. Esse arquivo será referido como “arquivo de domínio” nas descrições dos conjuntos. Uma vez feita a leitura destes conjuntos, são criadas propriedades observáveis no artefato para que os agentes possam conhecer esses elementos da EDom.

O Arquivo 3 mostra um exemplo de domínio de transporte de passageiros, que será utilizado no estudo de caso descrito no Capítulo 5. Os conjuntos e propriedades observáveis foram representados do seguinte modo:

- *Função ω* : A função ω é apresentada pela anotação *goalid/1*, sendo o termo da anotação o nome do predicado do objetivo organizacional *og*. No Arquivo 3, essa

anotação pode ser vista após alguns elementos, como $[goalid(goToVehicle)]$ e $[goalid(moveAction)]$.

- **Conjunto P :** O conjunto P é representado no arquivo de domínio pelos predicados na base de conhecimento. No Arquivo 3, temos como exemplo os predicados $moveTo/3$, $road/2$ e $at/2$.
- **Conjunto A :** O conjunto A é representado no arquivo de domínio pelos planos na biblioteca de planos. Esses planos seguem a definição de um plano em **Jason**. As precondições do plano são equivalentes às precondições de uma ação e no corpo do plano são utilizados apenas os elementos com operadores $+$ e $-$ como efeitos respectivamente positivos e negativos de ações. No Arquivo 3, temos como exemplo as ações $enterPassenger(PERSON, VEHICLE)$, $enterDriver(PERSON, VEHICLE)$, $leavePassenger(PERSON, VEHICLE)$, $leaveDriver(PERSON, VEHICLE)$ e $directMove(THING, FROM, TO)$.
- **Conjunto G :** O conjunto G é representado no arquivo de domínio pelos predicados específicos denominados $goal/1$. O termo dentro desses predicados é uma lista de estados a serem alcançados. No Arquivo 3, temos como exemplo vários objetivos $goal([moveTo(OBJECT, FROM, TO)])$; entretanto, cada um deles possui $\omega(g)$ (mapeamento para objetivos organizacionais) diferentes.

Figura 11: Artefato DomainBoard usado para a EDom.

Fonte: Autor.

A Figura 11 mostra o artefato **DomainBoard**. Esse artefato possui apenas propriedades observáveis. Uma vez que ele representa conhecimento inserido pelo projetista, não

é possível aos agentes modificá-lo, apenas percebê-lo. Nesse artefato, as propriedades observáveis `domainAction` representam elementos do conjunto A ; as propriedades observáveis `domainGoal` representam os elementos do conjunto G ; e a propriedade observável `domainPredicates` representa o conjunto P .

4.2.2 Planejador (**BasePlanner**)

Uma vez que o planejamento é realizado pelo agente e o arcabouço **JaCaMo** não contém um planejador embarcado, foi implementado um artefato para realizar o planejamento. Esse artefato segue um modelo base de artefato, a fim de permitir que diferentes planejadores possam ser usados sem precisar modificar os agentes. Ele disponibiliza três operações bem simples:

- `criar_dominio/4`: A operação de `criar_dominio/4` recebe como primeiro argumento o nome de um arquivo com o domínio de planejamento; o segundo argumento é o nome do domínio; o terceiro argumento são as ações que serão adicionadas ao domínio; e o quarto é o endereço do arquivo de domínio resultante. Nessa implementação, o nome do arquivo de domínio é baseado no nome do agente, permitindo assim que cada agente possa ter seus próprios métodos e/ou ações inclusos. Seria possível também o agente criar o arquivo com a definição do domínio de planejamento, mas isso limitaria o domínio de planejamento apenas ao conhecimento do agente.
- `criar_problema/5`: A operação de `criar_problema/5` recebe como primeiro argumento o nome do domínio; o segundo argumento é o nome do objetivo; o terceiro argumento é o estado inicial do mundo na forma de uma lista de predicados; o quarto é o estado desejado do mundo também na forma de uma lista de predicados; e o quinto é o endereço do arquivo de problema resultante.
- `criar_plano/4`: A operação de `criar_plano/4` recebe como primeiro argumento o endereço de um arquivo de domínio; o segundo argumento é o endereço de um arquivo de problema; o terceiro argumento é um resultado booleano que indica se existe um plano; e o quarto argumento é o plano criado.

Foi escolhido separar a criação de problemas e domínio no artefato de planejamento para no futuro permitir a diferentes agentes criarem problemas e/ou utilizar problemas de outros agentes para criar seus planos.

A Figura 12 mostra a implementação do artefato base para planejadores. Esse artefato não contém propriedades observáveis, uma vez que ele não fornece percepções para os

Figura 12: Artefato BasePlanner.

Fonte: Autor.

agentes. Ele pode ser usado como base para qualquer linguagem de planejamento que necessite um domínio e um problema, e permite que sejam implementados diferentes algoritmos. Pode-se, por exemplo, ter um artefato chamado **PDDLPlanner** que estende o artefato **BasePlanner** e implementa as operações *criar_dominio* e *criar_problema* para gerar domínio e problema em PDDL [MCDERMOTT et al., 1998], e um outro tipo de artefato **GraphPlanPlanner** que estende o artefato **PDDLPlanner** e implementa na operação *criar_plano* o planejamento utilizando o algoritmo GraphPlan [BLUM; FURST, 1997].

Para o estudo de caso deste trabalho apresentado no capítulo 5, foi implementado um artefato **SHOPPlanner** que utiliza o algoritmo SHOP [NAU et al., 1999], que utiliza formalismo HTN. A tradução entre os elementos da **EDom** e a definição de um domínio HTN [EROL; HENDLER; NAU, 1995] para um agente *ag* se dá da seguinte maneira:

- Para um domínio HTN $D = \{Op, Me\}$, pode-se obter $Op_{ag} = A \cup A_{ag}$, sendo A o conjunto de objetivos procedurais do domínio e A_{ag} os operadores próprios do agente; e Me_{ag} o conjunto de métodos do agente para decompor um objetivo.
- Para um problema HTN $P_{htn} = \{D, I, d\}$, onde D é o domínio definido acima, I e d são respectivamente F e I do problema de planejamento $\lambda(ag, \omega(og_i))$ onde og_i é o objetivo organizacional que o agente tenta atingir.

4.3 Agentes

Os agentes no **JaCaMo** são programados na linguagem **Jason**, e como um dos elementos da **EDom** são os agentes, foi necessário criar planos para que os agentes possam utilizar seus conhecimentos na maneira proposta na **EDom**. O arquivo de inclusão para os agentes serve tanto para obter os elementos do conjunto *Ag* como para a implementação da função λ , e consequentemente a obtenção do conjunto *Pp*. Antes de apresentar como esses conjuntos são obtidos, é necessário primeiro apresentar duas ações internas cuja criação foi necessária para implementar esta proposta.

4.3.1 Ações Internas

Como os objetivos e ações da **EDom** não possuem variáveis instanciadas com valores, foi necessária a criação de duas ações internas para se realizar a atribuição de valores a essas variáveis. Mais precisamente, tal funcionalidade é utilizada pela função λ para realizar a unificação dos termos variáveis. Tais ações internas são:

- *argVarNames/2*: A ação interna *argVarNames/2* é responsável por retornar o nome das variáveis de um predicado. O primeiro argumento é um predicado com variáveis nomeadas e o segundo argumento é uma lista com os nomes das variáveis nomeadas do primeiro termo. Tomando como exemplo o Arquivo 3, a ação interna *argVarNames(moveTo(Driver, Origin, Destination), Result)*, o termo *Result* unifica com *["Driver", "Origin", "Destination"]*.
- *unify/3*: A ação interna *unify/3* é responsável por atribuir valores às variáveis nomeadas de um termo. O primeiro argumento é o termo para ser unificado; o segundo argumento é uma lista de pares *[Nome, Valor]*, onde *Nome* é nome do termo variável para ser unificado e *Valor* é o valor que ele irá receber; e o terceiro argumento retorna um termo cujas variáveis nomeadas do primeiro argumento foram unificadas com os respectivos valores presentes na lista do segundo argumento. Novamente tomando como exemplo o Arquivo 3, a ação interna *unify(moveTo(Driver, Origin, Destination), [{"Driver", driver1}, {"Origin", place1}, {"Destination", place2}], Result)*, o termo *Result* unifica com *move(driver1, place1, place2)*.

4.3.2 Leitura da Especificação de Domínio

Segundo a proposta deste trabalho, para conceber um agente que possa ter autonomia de planejamento deve-se incluir em seu código um arquivo `edom.asl`. Tal arquivo contém somente dois planos para que o agente execute um objetivo organizacional. Entretanto, como os objetivos organizacionais são mapeados para diferentes tipos de objetivos na EDom, faz-se necessário definir dois planos diferentes, uma vez que alcançar um objetivo procedural é diferente de alcançar um objetivo declarativo.

Objetivos Procedurais

Para alcançar objetivos procedurais são feitos os seguintes passos:

1. *Adicionar à biblioteca de planos do agente a ação a (linha 2):* primeiramente, é necessário adicionar a ação que irá ser executada na biblioteca de planos do agente para que ela exista quando ele tente cumpri-la;
2. *Criar um plano para o objetivo organizacional (linhas 3 a 16):* através da função ω , é possível saber a qual objetivo organizacional uma dada ação está associada. Dado isso, cria-se um plano para o objetivo organizacional og que consiste em executar a ação de $\omega(og)$;
3. *Unificar as variáveis da ação a (linhas 6 a 13):* Como a ação possui variáveis e não valores constantes em seus elementos, é preciso unificar essas variáveis com valores definidos. Utiliza-se para isso o conjunto $Garg$ do agente para obter os valores de cada variável. O processo de unificação é similar ao realizado para objetivos declarativos, entretanto o resultado ainda é uma ação;
4. *Executar a ação unificada (linhas 14):* Uma vez que as variáveis estejam unificadas, o agente executa a ação.
5. *Comunicar a organização o resultado (linhas 15):* Após executar a ação, o agente comunica à organização o seu sucesso ou falha.

O Algoritmo 4 apresenta o código Jason para alcançar um objetivo procedural. Nela é possível ver o código para as etapas descritas, sendo elas identificadas pelos números entre parenteses. Na Figura 13, é possível ver os passos previamente apresentados como um diagrama de sequência.

Figura 13: Diagrama de sequência para cumprimento de um objetivo procedural.

Fonte: Autor.

Objetivos Declarativos

Já no caso de objetivos declarativos, é necessário representá-los como um problema de planejamento. Para tal, é necessário obter primeiramente a representação do agente segundo a EDom. O processo é um pouco diferente do caso de objetivos procedurais, tendo os seguintes passos:

1. *Criar um plano para o objetivo organizacional (linha 3)*: assim como para objetivos procedurais, aqui também é criado um plano para o objetivo organizacional *og*;
2. *Unificar as variáveis do objetivo (linhas 8 a 14)*: o processo é similar ao processo para objetivos procedurais, entretanto é feita a unificação de um objetivo de domínio *g*, e não de uma ação *a*;
3. *Criar o problema de planejamento *pp* (linha 15)*: com as variáveis do objetivo unificadas, é criado o problema de planejamento através de função λ ;
4. *Criar plano (linhas 16 a 20)*: tendo o problema de planejamento *pp*, o agente utiliza o artefato planejador para criar os arquivos de domínio e problema, e na sequência utilizar o algoritmo de planejamento para criar um plano;
5. *Executar o plano (linhas 23 e 27 a 36)*: o agente executa cada ação do plano criado;
6. *Comunicar a organização o resultado (linha 24)*: após executar o plano, o agente comunica à organização o seu sucesso ou falha.

Figura 14: Diagrama de sequência para cumprimento de um objetivo declarativo.

Fonte: Autor.

O Algoritmo 5 apresenta o código Jason para alcançar um objetivo declarativo. Nela é possível ver o código para as etapas descritas, sendo elas identificadas pelos números entre parênteses.

A Figura 14 mostra o diagrama de sequência dos passos utilizados pelo agente para cumprir um objetivo declarativo. Note-se que, neste caso, é utilizado o artefato de planejamento para se criar o plano na etapa 4, e que a relação do agente com a organização é a mesma tanto para objetivos declarativos quanto para procedurais.

Arquivo 3: Arquivo usado para construir o artefato DomainBoard.

```

1: //domaingoals
2: goal([moveTo(OBJECT, FROM, TO)])[goalid(passengerAtMeetingPoint)].
3: goal([moveTo(OBJECT, FROM, TO)])[goalid(driveToMeetingPoint)].
4: goal([moveTo(OBJECT, FROM, TO)])[goalid(goToDestination)].
5: goal([moveTo(OBJECT, FROM, TO)])[goalid(goToVehicle)].
6:
7: //domainpredicates
8: moveTo(-, -, -).
9: road(-, -).
10: at(-, -).
11: passenger(-, -).
12: driving(-, -).
13:
14: //domainactions
15: + !enterPassenger(PERSON, VEHICLE)[goalid(enterVehicle)]
16:     :          at(PERSON, PLACE) & at(VEHICLE, PLACE)
17:     < -      - at(PERSON, PLACE);
18:           + passenger(PERSON, VEHICLE);.
19:
20: + !enterDriver(PERSON, VEHICLE)[goalid(driveVehicle)]
21:     :          at(PERSON, PLACE) & at(VEHICLE, PLACE)
22:     < -      - at(PERSON, PLACE);
23:           + driving(PERSON, VEHICLE);.
24:
25: + !leavePassenger(PERSON, VEHICLE)[goalid(leaveVehicle)]
26:     :          passenger(PERSON, VEHICLE) & at(VEHICLE, PLACE)
27:     < -      - passenger(PERSON, VEHICLE);
28:           + at(PERSON, PLACE);.
29:
30: + !directMove(THING, FROM, TO)[goalid(moveAction)]
31:     :          at(THING, FROM) & road(FROM, TO)
32:     < -      - at(THING, FROM);
33:           + at(THING, TO);.
34:
35: + !leaveDriver(PERSON, VEHICLE)[goalid(leaveDriver)]
36:     :          driving(PERSON, VEHICLE) & at(VEHICLE, PLACE)
37:     < -      - driving(PERSON, VEHICLE);
38:           + at(PERSON, PLACE);.

```

Algorithm 4: Código do agente para alcançar um objetivo procedural.

```

1: + domainAction(Domain, goalid(GoalId), Action, -)
2:   < -.addplan(Action);
3:   + {
4:     + !GoalId)
5:     : domainAction(-, goalid(GoalId), -, Trigger)
6:     < -jia.argVarNames(Trigger, GoalArgNames);
7:       for(.member(GoalArgName, GoalArgNames))
8:         {
9:           ?goalArgument(-, GoalId, GoalArgName, GoalArgValue);
10:        }
11:     .findall([ArgName, ArgValue], goalArgument(SchName, GoalId,
12:       ArgName, ArgValue), Values);
13:     jia.unify(Trigger, Values, UnifiedAction);
14:     !UnifiedAction;
15:     goalAchieved(GoalId)
16:   }.

```

Algorithm 5: Código do agente para alcançar um objetivo declarativo.

```

1: + domainGoal(Domain, goal(Goal)[goalid(GoalId)])
2:   < - + {
3:     +!GoalId
4:     : domainPredicates(Domain, DomainPredicates) &
5:     domainGoal(Domain, goal(Goal)[goalid(GoalId)])
6:     < - .my_name(Name);
7:     .concat(Name, ".shp", DomainFileName);
8:     jia.argVarNames(Goal, GoalArgNames);
9:     for(.member(GoalArgName, GoalArgNames))
10:        ?goalArgument(_, GoalId, GoalArgName, GoalArgValue);
11:   }
12:   .findall([ArgName, ArgValue], goalArgument(SchName, GoalId,
13:     ArgName, ArgValue), Values);
14:   jia.unify(Goal, Values, TargetState);
15:   .findall(Belief, Belief[_] & .member(Belief, DomainPredicates),
16:     InitialState);
17:   .findall(Action, domainAction(Domain, _, Action, _), DomainActions);
18:   writedomain(DomainFileName, Domain, DomainActions, DomainPath);
19:   writeproblem(DomainFileName, GoalId, InitialState, TargetState,
20:     ProblemPath);
21:   planningAlgorithm(DomainPath, ProblemPath, Sucess,
22:     CreatedPlan)[artifact_id(PlannerId)];
23:   if(Sucess)
24:   {
25:     !createdPlan(GoalId, CreatedPlan);
26:     goalAchieved(GoalId)
27:   }
28:   + !createdPlan(GoalId, CreatedPlan)
29:   < - if(not.empty(CreatedPlan))
30:   {
31:     for(.member(Intention, CreatedPlan))
32:     {
33:       .print("doingdomainaction : ", Intention);
34:       !Intention[goalid(-)]
35:     }
36:   }
37:   goalAchieved(GoalId).

```

5 ESTUDO DE CASO

5.1 Descrição do Cenário

O cenário utilizado para o estudo de caso, apresentado em [MAIA; SICHMAN, 2018], é composto de várias cidades conectadas por estradas. Nesse cenário, existe um agente que deseja se locomover de uma cidade origem para uma cidade destino. Existem também agentes que possuem veículos para locomoção. A organização utilizada nesse caso funciona como um meio dos agentes que possuem veículo se coordenarem para transportar o agente que deseja se locomover.

O cenário escolhido para o estudo foi baseado em situações do mundo real. Hoje em dia existem diversos aplicativos para transporte que lidam com situações similares a esse cenário: passageiros querendo ser transportados e diversos motoristas querendo transportá-los. Isso torna possível que seja projetada uma organização baseada em organizações reais e que seja possível também especificar um domínio baseado no ambiente real.

Tal cenário é apresentado na Figura 15.

5.2 Descrição da Organização

Esta seção explica o funcionamento da organização utilizada no estudo de caso utilizando o modelo MOISE [HÜBNER, 2003]. Como dito anteriormente, a organização escolhida foi baseada na percepção do projetista sobre organizações reais.

Especificação Estrutural

A Especificação Estrutural utilizada foi baseada na percepção de como são organizados os aplicativos de táxis e caronas. Como pode ser visto na Figura 16, ela é composta de 2 papéis, *driver* e *passenger*, que representam respectivamente motoristas e passageiros.

Figura 15: Cenário de transporte de passageiros.

Fonte: Autor.

Figura 16: Especificação Estrutural para transporte de passageiros.

Especificação Estrutural

Fonte: Autor.

Ela contém ainda um grupo chamado *driverGroup*. Dentro desse grupo, pode existir um *passenger* e pelo menos um *driver*. Existe uma relação de comunicação entre os papéis *passenger* e *driver*.

Especificação Funcional

A Especificação Funcional também foi baseada no funcionamento dos aplicativos mencionados anteriormente. Como pode ser visto na Figura 17, ela é composta de um esquema social nomeado *drivingScheme*, cujo objetivo global *movePassenger* consiste em realizar o transporte de um passageiro. Caso existam mais passageiros no ambiente, pode-se instanciar esse esquema para cada passageiro, tornando cada viagem independente. O objetivo global é decomposto como mostra a árvore na figura, chegando até aos objetivos-folhas. Estes últimos são mapeados para objetivos procedurais ou declarativos através da Especificação de Domínio. Note-se que a especificação funcional da figura 17 segue o mesmo padrão da figura 3, mas foi estendida para representar os diferentes tipos de objetivo: objetivos declarativos são representados por linhas cheias nas folhas, enquanto os objetivos procedurais são representados por linhas pontilhadas. A Especificação Funcional também contém duas missões: *transportPassenger* e *beTransported*, sendo a primeira designada para tarefas do motorista que irá transportar e a segunda para o passageiro que será transportado.

Figura 17: Especificação Funcional para transporte de passageiros.

Fonte: Autor.

Especificação Deontica

A Especificação Deontica faz a ligação entre papéis e missões. Pode-se ver na Tabela 2 que a missão *transportPassenger* é designada para o agente no papel *driver* que irá transportar o passageiro. A relação deontica entre eles é de obrigação, uma vez que para que o objetivo global seja alcançado é necessário que alguém transporte o passageiro. A missão *beTransported*, que foi designada para o agente no papel *passenger*, possui uma relação deontica de obrigação, já que o sucesso do objetivo depende integralmente do passageiro ser transportado.

Tabela 2: Especificação Deontica para transporte de passageiros.

Papel	Missão	Ligação
driver	transportPassenger	obligation
passenger	beTransported	obligation

Fonte: Autor.

5.3 Agente sem planejamento on-line

O primeiro caso estudado foi com os agentes sem a capacidade de planejar. Nesse caso, cada agente possui seus próprios planos para os objetivos da organização, sendo possível definir na programação do agente diversos planos para um mesmo objetivo.

Por não existir a Especificação de Domínio proposta nesse trabalho, não existe uma maneira de regular a autonomia de planejamento dos agentes, logo, todos os agente possuem autonomia de planejamento completa em relação a todos os objetivos.

Por não existir também uma definição comum de ações, é possível ter diferentes agentes que possuam definições diferentes para a mesma ação *goToVehicleLocation*, como mostram os Algoritmos 6 e 7.

Algorithm 6: Plano para o objetivo organizacional *goToVehicleLocation* no agente *driver1*.

```

1: + !goToVehicleLocation(Vehicle)
2:   : .my_name(Me) & at(Me, MyPos) & at(Vehicle, VehiclePos)
3:   < - +!walkTo(VehiclePos).
```

No Algoritmo 6, o plano mostra que para cumprir o objetivo *goToVehicleLocation* o agente *driver1* tem a intenção de andar até a localização do veículo. Já no Algoritmo 7,

Algorithm 7: Exemplo do plano para o objetivo organizacional *goToVehicleLocation* no agente *driver2*.

```

1: + !goToVehicleLocation(Vehicle)
2:   : .my_name(Me) & at(Me, MyPos) & at(Vehicle, VehiclePos)
3:   < - requestTransportation(MyPos, VehiclePos).

```

o agente *driver2* executa uma operação em um artefato para pedir transporte até o local do veículo, sendo este último recebido como argumento do objetivo. É possível notar que ambos os agentes possuem autonomia completa para escolher e executar os planos a fim de cumprir esse objetivo de diferentes maneiras.

5.4 Agente com planejamento hierárquico

No segundo caso estudado foi utilizada a proposta de especificação de domínio em conjunto com planejamento hierárquico.

Para o planejamento hierárquico, foi utilizada a proposta de Especificação de Domínio a fim de especificar a autonomia dos agentes e fornecer um domínio comum para que o planejamento fosse realizado.

O Arquivo 8 detalha a especificação utilizada e ela segue o padrão mencionado anteriormente na implementação para a inicialização do artefato *DomainBoard*. Para o planejamento, como mencionado na Seção 4.2.2, foi implementado um artefato *SHOPPlanner* que utiliza o algoritmo SHOP [NAU et al., 1999] que utiliza o formalismo HTN.

Nesse caso, os agentes estão aptos a atingirem tanto objetivos declarativos quanto procedurais, uma vez que ambos os tipos estão representados na *EDom*.

Aqui, os agentes possuem autonomia para criar diferentes planos para os objetivos organizacionais *driveToMeetingPoint*, *goToDestination*, *goToVehicle* e *passenger AtMeetingPoint*. Já para os outros objetivos, eles realizam uma ação específica de maneira igual independente de qual agente tenta atingi-los. A definição de cada ação pode ser vista no Arquivo 8.

5.5 Análise Comparativa

No arcabouço *JaCaMo*, os agentes BDI, que são implementados em *Jason*, possuem capacidade nativa de selecionar um entre os diversos planos em sua biblioteca para atingir

Arquivo 8: Especificação de Domínio usado no estudo de caso.

```

1: goal([moveTo(OBJECT, FROM, TO))][goalid(passengerAtMeetingPoint)].
2: goal([moveTo(OBJECT, FROM, TO))][goalid(driveToMeetingPoint)].
3: goal([moveTo(OBJECT, FROM, TO))][goalid(goToDestination)].
4: goal([moveTo(OBJECT, FROM, TO))][goalid(goToVehicle)].
5:
6: moveTo(-, -, -).
7: road(-, -).
8: at(-, -).
9:
10: + !enterPassenger(PERSON, VEHICLE)[goalid(enterVehicle)]
11:   : at(PERSON, PLACE) & at(VEHICLE, PLACE)
12:   < -- at(PERSON, PLACE);
13:   + passenger(PERSON, VEHICLE); .
14:
15: + !enterDriver(PERSON, VEHICLE)[goalid(driveVehicle)]
16:   : at(PERSON, PLACE) & at(VEHICLE, PLACE)
17:   < -- at(PERSON, PLACE);
18:   + driving(PERSON, VEHICLE); .
19:
20: + !leavePassenger(PERSON, VEHICLE)[goalid(leaveVehicle)]
21:   : passenger(PERSON, VEHICLE) & at(VEHICLE, PLACE)
22:   < -- passenger(PERSON, VEHICLE);
23:   + at(PERSON, PLACE); .
24:
25: + !directMove(THING, FROM, TO)[goalid(moveAction)]
26:   : at(THING, FROM) & road(FROM, TO)
27:   < -- at(THING, FROM);
28:   + at(THING, TO); .
29:
30: + !leaveDriver(PERSON, VEHICLE)[goalid(leaveDriver)]
31:   : driving(PERSON, VEHICLE) & at(VEHICLE, PLACE)
32:   < -- driving(PERSON, VEHICLE);
33:   + at(PERSON, PLACE); .

```

um mesmo objetivo. Entretanto, eles não possuem capacidade para criar novos planos, já que estes são definidos a priori no código de cada agente.

Comparando um agente **JaCaMo** tradicional com os agentes utilizados no estudo de caso, é possível perceber que há uma distinção entre suas capacidades, sendo o primeiro apenas capaz de escolher entre planos pré-definidos, enquanto o último também é capaz também de criá-los. Esses agentes, apesar de seguirem o paradigma BDI, utilizam-se de um artefato para criar planos HTN para alcançar objetivos declarativos. Tal abordagem também foi utilizada em outros trabalhos que integram planejamento HTN em agentes BDI, como [CARDOSO; BORDINI, 2016; SARDINA; SILVA; PADGHAM, 2006]. Pode-se citar também [MENEGUZZI; LUCK, 2007] que propõe um implementação para

permitir agentes BDI criarem planos utilizando STRIPS [FIKES; NILSSON, 1971].

Em CANPlan [SARDINA; SILVA; PADGHAM, 2006] é feita uma proposta para incluir planejamento em agentes BDI, tornando-os capazes de planejar. Entretanto, o trabalho não menciona se os agentes tem ou não autonomia de planejamento, abordagem similar é adotada [MENEGUZZI; LUCK, 2007], entretanto utilizando outro formalismo de planejamento. Já o trabalho de DOMAPS [CARDOSO; BORDINI, 2016] traz uma abordagem semelhante à utilizada neste trabalho, fazendo uso tanto de organizações quanto de planejamento, mas nesse trabalho não se tem uma definição explícita diferenciando os tipos de objetivos entre declarativos ou procedurais na organização, deixando para os agentes decidirem.

Além de adicionar a capacidade de criar planos para os agentes BDI como alguns dos trabalhos citados anteriormente, este trabalho acrescenta também uma distinção entre tipos de objetivos no contexto organizacional. Tal distinção permite ao projetista da organização primeiro especificar os objetivos organizacionais para os quais os agentes devem formular e solucionar problemas, e segundo especificar a definição de planejamento desse problema.

Tabela 3: Tabela mostrando diferenças entre os trabalhos.

	CANPlan	Meneguzzi e Luck [2007]	DOMAPS	EDom
Criação de Planos	Sim	Sim	Sim	Sim
Objetivos Declarativos	Explicito	Explicito	Implícito	Explicito
Objetivos Procedurais	Explicito	Implícito	Implícito	Explicito
Objetivos Organizacionais	Não	Não	Sim	Sim
Definição dos Objetivos	Agente	Agente	Agente	Organização
Agentes	BDI	BDI	BDI	BDI
Planejamento	HTN	STRIPS	MA-HTN	HTN

Fonte: Autor.

Na Tabela 3, ilustram-se as diferenças entre os trabalhos acima mencionados e o proposto neste trabalho. Em [SARDINA; SILVA; PADGHAM, 2006] é possível distinguir explicitamente entre objetivos declarativos e procedurais pelo uso dos construtores **Plan** e **Goal**. Em [MENEGUZZI; LUCK, 2007] todos os planos são por padrão procedurais, sendo utilizada a intenção $+goal_conj/1$ para representar objetivos declarativos caso necessário. Já em [CARDOSO; BORDINI, 2016] essa distinção é feita implicitamente, dependendo do agente estar tentando alcançar uma tarefa complexa ou tarefa simples no formalismo MA-HTN [CARDOSO; BORDINI, 2017]. Objetivos organizacionais estão presentes em [CARDOSO; BORDINI, 2016], mas a definição destes não se encontra na organização, e

sim nos problemas que cada agente tenta resolver (para o caso declarativo), como também no seu domínio (para o caso procedural).

Cabe ainda resaltar que no caso de DOMAPS, o modelo proposto não se restringe ao uso de agentes BDI. Já no caso de EDom, esta proposta não está restrita nem ao formalismo BDI nem tampouco ao uso de HTN (como ilustrado neste trabalho), sendo possível integrar até mesmo agentes de diferentes tipos e agentes utilizando diferentes formalismos em uma mesma organização.

Embora os agentes propostos neste trabalho possam utilizar a EDom para criar planos para atingir um objetivo, eles não são obrigados a fazê-lo: podem exercer sua autonomia para escolher diferentes planos pré-definidos em sua biblioteca, ou criar um novo caso assim o desejem.

Portanto, neste trabalho permite-se diversos graus de autonomia de planejamento do agente: (i) um agente recebe um objetivo procedural e apenas executa o plano pré-definido para atingi-lo; (ii) um agente recebe um objetivo declarativo, e baseado em informações locais e circunstanciais, escolhe um dentre vários planos de uma biblioteca de planos que foram pré-definidos em sua biblioteca e (iii) um agente recebe um objetivo declarativo, e utilizando a EDom, cria um plano em tempo de execução para atingi-lo.

A EDom não trata do problema de como especificar, projetar e programar um agente com capacidade para realizar planejamento como fazem, como proposto em [SARDINA; SILVA; PADGHAM, 2006] e [SILVA, 2017]. Tal problema se refere ao projeto de agentes, e não de especificação organizacional, e encontra-se fora do escopo desse trabalho. A EDom trata apenas de como especificar no nível organizacional os possíveis tipos de autonomia de planejamento dos agentes para atingir diferentes objetivos.

Seu uso é apropriado para projetos de SMA nos quais o projetista não pode prever com precisão as circunstâncias nas quais os agentes irão atuar. Essa implementação do agente se aproxima mais de um verdadeiro agente de resolução de problemas, como descrito em [RUSSELL; NORVIG, 2009].

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 Conclusões

Nesse trabalho, foi apresentada uma definição para Especificação de Domínio (EDom) que permite definir diferentes graus de autonomia de planejamento para objetivos organizacionais. Para tal, foi utilizado o conceito de tipos de objetivo propostos por [RIMSIDIJK; DASTANI; WINIKOFF, 2008]. A EDom foi implementada e integrada com o modelo organizacional MOISE [HÜBNER, 2003], utilizando a plataforma JaCaMo.

Foi mostrado que a EDom permite distinguir os diferentes tipos de objetivos apresentados, dando assim a possibilidade de especificar a presença ou ausência de autonomia de planejamento para um dado objetivo organizacional presente no modelo MOISE [HÜBNER, 2003].

Agentes com capacidade de planejamento podem utilizar a EDom para a tradução de objetivos organizacionais para problemas de planejamento. A definição de objetivos procedurais como ações de planejamento garante que eles serão atingidos de uma maneira pré-definida e fixa, não importando qual agente tenta atingí-los, restringindo assim a sua autonomia.

Por outro lado, objetivos declarativos foram criados usando uma descrição similar à descrição de problemas de planejamento já existentes [GEORGIEVSKI; AIELLO, 2015]. Em conjunto com o formalismo HTN, isso permite aos agente usarem suas próprias ações e/ou métodos para procurar uma solução.

Este trabalho considera relevante o estudo de autonomia organizacional, em particular a de planejamento, para agentes que atuam em um SMA. A abordagem proposta neste trabalho permite ao projetista da organização ter um controle mais rígido ou mais flexível sobre as ações dos agentes quando necessário, sem modificar o código agente.

6.2 Trabalhos Futuros

No futuro, pode-se utilizar a **EDom** em diferentes trabalhos que utilizem os conceitos de planejamento e organizações. Uma possibilidade seria introduzir uma descrição formal de organizações em algoritmos de planejamento, possibilitando assim que estes possam considerar restrições organizacionais.

Um caso interessante seria utilizar a **EDom** em um cenário com diferentes tipos de agentes, não apenas BDI; além disso, poderiam ser utilizados diferentes formalismos de planejamento dentro de uma mesma organização, uma vez que a **EDom** foi projetada para ser utilizada com qualquer modelo organizacional, formalismo de planejamento e paradigma de agentes.

Outra possibilidade seria integrar as definições de tipos de objetivos aqui apresentadas à verificação de fatos no ambiente, como proposto nas *Situated Artificial Institutions* propostas por [BRITO; HÜBNER; BOISSIER, 2018]); pode-se assim criar um modelo mais rico capaz de inferir o estado de um objetivo através dos fatos ambientais.

Trabalhos futuros podem também propor uma linguagem mais robusta para representar os diferentes tipos de objetivos, como a definição proposta por [YAO; SILVA; LOGAN, 2016], ou ainda incorporar a arquitetura proposta por [RIEMSDIJK; DASTANI; WINIKOFF, 2008] para determinar o estado de um objetivo à **EDom**. Também seria possível aumentar a granularidade de representação de autonomia, definindo subtipos de objetivos e relacionando-os a objetivos organizacionais.

REFERÊNCIAS

- ALECHINA, N.; DASTANI, M.; LOGAN, B. Programming norm-aware agents. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. **Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems**. [S.l.], 2012. v. 2, p. 1057–1064.
- ALLEN, J. F. Planning as temporal reasoning. **KR**, v. 91, p. 3–14, 1991.
- BARBER, K. S. et al. Dimensions of adjustable autonomy. In: SPRINGER. **Pacific Rim International Conference on Artificial Intelligence**. [S.l.], 2000. p. 353–361.
- BLUM, A. L.; FURST, M. L. Fast planning through planning graph analysis. **Artificial intelligence**, Elsevier, v. 90, n. 1-2, p. 281–300, 1997.
- BOELLA, G.; TORRE, L. W. van der. Regulative and constitutive norms in normative multiagent systems. **KR**, v. 4, p. 255–265, 2004.
- BOISSIER, O. et al. Multi-agent oriented programming with jacamo. **Science of Computer Programming**, Elsevier, v. 78, n. 6, p. 747–761, 2013.
- BORDINI, R. H.; HÜBNER, J. F. Bdi agent programming in agentspeak using jason. In: SPRINGER. **International Workshop on Computational Logic in Multi-Agent Systems**. [S.l.], 2005. p. 143–164.
- BRADSHAW, J. M. et al. Dimensions of adjustable autonomy and mixed-initiative interaction. In: SPRINGER. **International Workshop on Computational Autonomy**. [S.l.], 2003. p. 17–39.
- BRITO, M. D.; HÜBNER, J. F.; BOISSIER, O. Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 32, n. 2, p. 219–251, 2018.
- BULLING, N.; DASTANI, M.; KNOBBOUT, M. Monitoring norm violations in multi-agent systems. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. **Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems**. [S.l.], 2013. p. 491–498.
- CARABELEA, C.; BOISSIER, O.; FLOREA, A. Autonomy in multi-agent systems: A classification attempt. In: SPRINGER. **International Workshop on Computational Autonomy**. [S.l.], 2003. p. 103–113.
- CARDOSO, R. C. **A decentralised online multi-agent planning framework for multi-agent systems**. Tese (Doutorado) — Escola Politécnica - Pontifícia Universidade Católica do Rio Grande do Sul, 2018.

- CARDOSO, R. C.; BORDINI, R. H. A distributed online multi-agent planning system. In: **Proceedings of the Workshop on Distributed and Multiagent Planning (ICAPS)**. [S.l.: s.n.], 2016. p. 15–23.
- CARDOSO, R. C.; BORDINI, R. H. A multi-agent extension of a hierarchical task network planning formalism. **ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal**, v. 6, n. 2, p. 5–17, 2017.
- CASTELFRANCHI, C. Guarantees for autonomy in cognitive agent architecture. In: SPRINGER. **International Workshop on Agent Theories, Architectures, and Languages**. [S.l.], 1994. p. 56–70.
- CASTELFRANCHI, C. Founding agents’ autonomy on dependence theory. In: **ECAI**. [S.l.: s.n.], 2000. v. 1, p. 353–357.
- COUTINHO, L. R.; SICHTMAN, J. S.; BOISSIER, O. Modelling dimensions for agent organizations. In: **Handbook of research on multi-agent systems: Semantics and dynamics of organizational models**. [S.l.]: IGI Global, 2009. p. 18–50.
- COUTINHO, L. R. et al. Modeling organization in mas: A comparison of models. In: **First Workshop on Software Engineering for Agent-oriented Systems**. [S.l.: s.n.], 2005. p. 1–10.
- DECKER, K. S. **Task environment centered simulation**. [S.l.]: Simulating Organizations: Computational Models of Institutions and Groups. AAAI Press/MIT Press, 1996.
- DELOACH, S. A.; MATSON, E. An organizational model for designing adaptive multiagent systems. In: **The AAAI-04 Workshop on Agent Organizations: Theory and Practice (AOTP 2004)**. [S.l.: s.n.], 2004. p. 66–73.
- EROL, K.; HENDLER, J. A.; NAU, D. S. **Semantics for hierarchical task-network planning**. [S.l.], 1995.
- FERBER, J.; GUTKNECHT, O. A meta-model for the analysis and design of organizations in multi-agent systems. In: IEEE. **Multi Agent Systems, 1998. Proceedings. International Conference on**. [S.l.], 1998. p. 128–135.
- FERBER, J.; GUTKNECHT, O.; MICHEL, F. From agents to organizations: an organizational view of multi-agent systems. In: SPRINGER. **International Workshop on Agent-Oriented Software Engineering**. [S.l.], 2003. p. 214–230.
- FIKES, R. E.; NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. **Artificial intelligence**, Elsevier, v. 2, n. 3-4, p. 189–208, 1971.
- GEORGIEVSKI, I.; AIELLO, M. Htn planning: Overview, comparison, and beyond. **Artificial Intelligence**, Elsevier, v. 222, p. 124–156, 2015.
- HAYASHI, H. Stratified multi-agent htn planning in dynamic environments. In: SPRINGER. **KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications**. [S.l.], 2007. p. 189–198.

- HORLING, B.; LESSER, V. A survey of multi-agent organizational paradigms. **The Knowledge Engineering Review**, Cambridge Univ Press, v. 19, n. 04, p. 281–316, 2004.
- HÜBNER, J. F. **Um modelo de reorganização de sistemas multiagentes**. Tese (Doutorado) — Escola Politécnica - Universidade de São Paulo, 2003.
- HÜBNER, J. F.; SICHTMAN, J. S.; BOISSIER, O. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: **Advances in artificial intelligence**. [S.l.]: Springer, 2002. p. 118–128.
- KITIO, R. et al. Organisational artifacts and agents for open multi-agent organisations: “giving the power back to the agents”. In: **Coordination, Organizations, Institutions, and Norms in Agent Systems III**. [S.l.]: Springer, 2008. p. 171–186.
- KOVACS, D. L. A multi-agent extension of pddl3. **WS-IPC 2012**, p. 19, 2012.
- KUSHMERICK, N.; HANKS, S.; WELD, D. S. An algorithm for probabilistic planning. **Artificial Intelligence**, Elsevier, v. 76, n. 1-2, p. 239–286, 1995.
- LIMA, I. C. A. de; NARDIN, L. G.; SICHTMAN, J. S. Gavel: A sanctioning enforcement framework. In: SPRINGER. **Proceedings of EMAS 2018, Stockholm, Sweden, July, 2018**. [S.l.], 2018.
- MAIA, A. V.; SICHTMAN, J. S. Extending the moise model with adjustable autonomy. In: **Proceedings of WESAAC 2017**. [S.l.: s.n.], 2017. p. 181–186.
- MAIA, A. V.; SICHTMAN, J. S. Explicit representation of planning autonomy in moise organizational model. In: **Proceedings of BRACIS 2018**. [S.l.: s.n.], 2018.
- MCDERMOTT, D. et al. Pddl-the planning domain definition language. 1998.
- MENEGUZZI, F.; LUCK, M. Composing high-level plans for declarative agent programming. In: SPRINGER. **International Workshop on Declarative Agent Languages and Technologies**. [S.l.], 2007. p. 69–85.
- MENEGUZZI, F.; SILVA, L. D. Planning in bdi agents: a survey of the integration of planning algorithms and agent reasoning. **The Knowledge Engineering Review**, Cambridge University Press, v. 30, n. 1, p. 1–44, 2015.
- MENEGUZZI, F. R.; ZORZO, A. F.; MÓRA, M. C. Mapping mental states into propositional planning. In: IEEE COMPUTER SOCIETY. **Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems**. [S.l.], 2004. v. 3, p. 1514–1515.
- NARDIN, L. G. et al. Classifying sanctions and designing a conceptual sanctioning process model for socio-technical systems. **The Knowledge Engineering Review**, Cambridge University Press, v. 31, n. 2, p. 142–166, 2016.
- NAU, D. et al. Shop: Simple hierarchical ordered planner. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the 16th international joint conference on Artificial intelligence**. [S.l.], 1999. v. 2, p. 968–973.

OMICINI, A.; RICCI, A.; VIROLI, M. Artifacts in the a&a meta-model for multi-agent systems. **Autonomous agents and multi-agent systems**, Springer, v. 17, n. 3, p. 432–456, 2008.

RAO, A. S. Agentspeak (1): Bdi agents speak out in a logical computable language. In: SPRINGER. **European Workshop on Modelling Autonomous Agents in a Multi-Agent World**. [S.l.], 1996. p. 42–55.

RICCI, A.; VIROLI, M.; OMICINI, A. Cartago: A framework for prototyping artifact-based environments in mas. In: SPRINGER. **International Workshop on Environments for Multi-Agent Systems**. [S.l.], 2006. p. 67–86.

RIEMSDIJK, M. B. V.; DASTANI, M.; WINIKOFF, M. Goals in agent systems: a unifying framework. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. **Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems**. [S.l.], 2008. v. 2, p. 713–720.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [S.l.]: Prentice Hall, 2009. v. 1.

SARDINA, S.; SILVA, L. de; PADGHAM, L. Hierarchical planning in bdi agent programming languages: A formal approach. In: ACM. **Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems**. [S.l.], 2006. p. 1001–1008.

SILVA, L. de. Bdi agent reasoning with guidance from htn recipes. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. **Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems**. [S.l.], 2017. p. 759–767.

VECHT, B. V. D. et al. A dynamic coordination mechanism using adjustable autonomy. In: **Coordination, Organizations, Institutions, and Norms in Agent Systems III**. [S.l.]: Springer, 2008. p. 83–96.

WOOLDRIDGE, M. **An introduction to multiagent systems**. [S.l.]: John Wiley & Sons, 2009.

YAO, Y.; SILVA, L. de; LOGAN, B. Reasoning about the executability of goal-plan trees. In: SPRINGER. **International Workshop on Engineering Multi-Agent Systems**. [S.l.], 2016. p. 176–191.