

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E  
INFORMÁTICA INDUSTRIAL**

**NOME DO AUTOR**

**TÍTULO EM PORTUGUÊS**

**DISSERTAÇÃO**

**CURITIBA**

**2019**

**NOME DO AUTOR**

**TÍTULO EM PORTUGUÊS**

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de “Mestre em Ciências” – Área de Concentração: Informática Industrial.

Orientador:      Nome do Orientador

**CURITIBA**

**2019**

---

Dados Internacionais de Catalogação na Publicação

---

T137    Sobrenome, Nome

          Título em português/ Nome do Autor. – 2019.

          66 f. : il. ; 30 cm

          Orientador: Nome do Orientador.

          Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2019.

          Bibliografia: f. 62-64.

          1. Teoria do controle. 2. Redes de comutação. 3. TCP/IP (Protocolo de rede de computação), ...

CDD (22. ed.) 621.3

---

Biblioteca xxxxxx

Título da Dissertação Nº 596:

**“Esquema de Controle de Congestionamento para  
TCP Baseado na Banda Disponível”.**

por

**Marcos Talau**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM CIÊNCIAS – Área de Concentração: Telemática, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Curitiba, às 09h30min. do dia 04 de maio de 2012. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:



Visto da coordenação:



Texto da dedicatória.

## **AGRADECIMENTOS**

Texto dos agradecimentos.

Texto da epígrafe.

## **RESUMO**

SOBRENOME, Nome. TÍTULO EM PORTUGUÊS. 66 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Texto do resumo (máximo de 500 palavras).

**Palavras-chave:** Palavra-chave 1, Palavra-chave 2, ...



## **ABSTRACT**

SOBRENOME, Nome. TITLE IN ENGLISH. 66 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Abstract text (maximum of 500 words).

**Keywords:** Keyword 1, Keyword 2, ...

## LISTA DE FIGURAS

FIGURA 1	– Arvore de objetivos definido pelo modelo Moise (HÜBNER et al., 2002)	23
FIGURA 2	– Linguagem para descrever um programa de multiagentes normativos com a possibilidade de violações e sanções na notação EBNF segundo o texto (DASTANI et al., 2009). Nesta notação, <i>&lt; ident &gt;</i> é usado para denotar uma <i>string</i> e <i>&lt; int &gt;</i> inteiros. Os termos <i>&lt; b – prop &gt;</i> e <i>&lt; i – prop &gt;</i> são usados para designar dois tipos de conjuntos de proposições que são disjuntos entre si	25
FIGURA 3	– Um programa descrito na linguagem proposta neste estudo onde um agente representa um passageiro em uma estação de trem que pode entrar com ou sem um <i>ticket</i> na plataforma e no trem (DASTANI et al., 2009).	26
FIGURA 4	– A estrutura geral das classes do modelo	33
FIGURA 5	– Diagrama de classes do Modelo	53
FIGURA 6	– Um programa descrito na linguagem proposta neste estudo onde um agente representa um passageiro em uma estação de trem que pode entrar com ou sem um <i>ticket</i> na plataforma e no trem (DASTANI et al., 2009).	56
FIGURA 7	– Ontologia que descreve <i>Domain Model</i> no model V3S (BAROT et al., 2013)	57

## LISTA DE TABELAS

TABELA 1	– Construtores da linguagem <i>ACTIVITY-DL</i> (BAROT et al., 2013) .....	58
TABELA 2	– As pré-condições possíveis para as atividades (BAROT et al., 2013) .....	59

## **LISTA DE QUADROS**

## **LISTA DE SIGLAS**

## LISTA DE SÍMBOLOS

## SUMÁRIO

<b>1 INTRODUÇÃO</b>	<b>13</b>
1.1 MOTIVAÇÃO	13
1.2 RELEVÂNCIA	13
1.3 OBJETIVOS	13
1.3.1 Objetivo Geral	13
1.3.2 Objetivos Específicos	13
<b>2 FUNDAMENTAÇÃO TEÓRICA</b>	<b>14</b>
2.1 AGENTES	14
2.2 ARTEFATOS	16
2.3 SISTEMA MULTIAGENTE	18
2.3.1 Conceitos Gerais de uma Organização de Sistemas Multiagentes	19
2.3.2 Formalização de Conceitos Específicos para SMA	21
2.4 NORMAS	24
2.5 RISCOS	26
2.6 LÓGICA MODAL	28
<b>3 METODOLOGIA</b>	<b>31</b>
3.1 ANÁLISE DOS MODELOS	31
3.2 ACOMPANHAMENTO DE PROFISSIONAIS EM ATIVIDADE DE RISCO	31
3.3 CONSTRUÇÃO DO MODELO	31
3.4 IMPLEMENTAÇÃO	31
<b>4 RESULTADOS</b>	<b>32</b>
4.1 ESTRUTURA CONCEITUAL	33
4.1.1 Módulos	33
4.1.2 Predicados	39
4.1.3 Regras	43
4.2 UML	53
4.3 CASO DE ESTUDO	55
4.4 RACIOCÍNIO	55
4.5 VALIDAÇÃO	55
<b>5 ANÁLISE COMPARATIVA</b>	<b>56</b>
5.1 MOISE+	56
5.1.1 Estrutura	56
5.1.2 Análise comparativa	56
5.2 DASTANI	56
5.2.1 Estrutura	56
5.2.2 Análise comparativa	57
5.3 V3S	57
5.3.1 Estrutura	57
5.3.2 Análise comparativa	59
5.4 NORMMAS	59
5.4.1 Estrutura	59

5.4.2 Análise comparativa .....	60
5.5 PERSPECTIVA GENÉRICA .....	60
<b>6 CONCLUSÃO .....</b>	<b>61</b>
6.1 AVALIAÇÃO DOS OBJETIVOS .....	61
6.2 TRABALHOS FUTUROS .....	61
<b>REFERÊNCIAS .....</b>	<b>62</b>
<b>Apêndice A – NOME DO APÊNDICE .....</b>	<b>65</b>
<b>Anexo A – NOME DO ANEXO .....</b>	<b>66</b>



## **1 INTRODUÇÃO**

### **1.1 MOTIVAÇÃO**

### **1.2 RELEVÂNCIA**

### **1.3 OBJETIVOS**

#### **1.3.1 OBJETIVO GERAL**

#### **1.3.2 OBJETIVOS ESPECÍFICOS**

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 AGENTES

Não existe uma definição universal para tratar o conceito de agente sendo que esse tópico se encontra em meio a debates e controvérsias. Contudo, existe um entendimento generalizado de que um comportamento *autonomo* é o cerne de noção que se tem por agente (JENNINGS; LESPÉRANCE, 2000). Apesar disso, a construção de um modelo computacional não pode ser feito sem uma definição. Assim sendo, nesse texto um agente é um sistema computacional que está situado em um dado ambiente e que apresenta comportamento autonomo (JENNINGS; LESPÉRANCE, 2000) (JENNINGS; LESPÉRANCE, 2000). Não apenas isso, mas um agente faz uso do seu comportamento autonomo com o propósito de atingir objetivos que a ele é designado (JENNINGS; LESPÉRANCE, 2000) (JENNINGS; LESPÉRANCE, 2000).

Como a definição de agente faz uso do conceito de ambiente, não é possível tratar esse tópico de forma consistente sem considerar a semântica deste termo. Assim sendo, ambiente é aquilo que apresenta as propriedades listadas a seguir (RUSSELL; NORVIG, 2003) (JENNINGS; LESPÉRANCE, 2000);

- *Acessibilidade vs Inacessibilidade*; Um ambiente acessível é aquele onde um agente consegue ter informações claras, precisas e atualizadas no que tange a característica do ambiente.
- *Determinístico vs Não-Determinístico*; Um comportamento determinístico é aquele onde uma ação possui um efeito claro e garantido, sem incertezas sobre o estado que irá resultar.
- *Episódico vs Não-Episódico*; Um ambiente tende a ser o mais episódico possível tanto quando o desempenho do agente estiver associado a um episódio discreto e específico no ambiente.

- *Estático vs Dinâmico*; Um ambiente é estático se não houver outros processos em paralelo aos eventos associados ao agente.
- *Discreto vs Contínuo*; Um ambiente é discreto se existe um número finito de ações e percepções.

Outro aspecto que está contido na definição consiste no conceito expresso pelo termo *autônomo*. Assim sendo, a pesquisa retratada neste texto trabalha esse termo da seguinte forma; Uma entidade que possui essa natureza tem a capacidade agir por si mesmo. Essa entidade não precisa de nenhuma outra entidade externa (ex. ser humano) para realizar decisões (JENNINGS; LESPÉRANCE, 2000) (JENNINGS; LESPÉRANCE, 2000).

Há uma série de exemplos que se enquadram dentro dessa definição. Sistemas de controle é um desses exemplos. Um *termostato* é um sistema de controle que está em um dado ambiente (como um quarto ou uma sala) (JENNINGS; LESPÉRANCE, 2000), gera dois sinais de saída (um desses sinais indica que a temperatura está baixa demais (ou alta demais dependendo da aplicação) e o outro sinal demonstra que a temperatura está no nível aceitável). O termostato tem o seu comportamento autônomo baseado em duas regras (JENNINGS; LESPÉRANCE, 2000):

- Se a temperatura estiver abaixo (ou acima) do nível de temperatura definido, então ligar o atuador.
- Se a temperatura estiver dentro do nível estabelecido, então desligar o atuador.

Outro exemplo de agentes consiste nos programas *Daemons* em sistemas *UNIX*. Esses algoritmos trabalham em segundo plano e monitoram um dado ambiente de *software*. Com base em certas regras, na ocorrência de um dado evento no ambiente, esses programas realizam uma dada atuação (JENNINGS; LESPÉRANCE, 2000).

Os exemplos presentes neste texto são apropriados dentro do conceito de agentes. Contudo, esses exemplos não se enquadram dentro da definição de agentes inteligentes (JENNINGS; LESPÉRANCE, 2000). Uma entidade que se enquadra dentro das características de um agente inteligente deve necessariamente respeitar a definição já apresentada e deve apresentar as seguintes propriedades; reatividade (capaz de perceber as mudanças que ocorrem no ambiente e responder a elas de maneira apropriada no que condiz aos objetivos do agente), pro-atividade (apresenta comportamento orientado a objetivos sendo que o agente toma as decisões a fim de satisfazer os objetivos em interesse) e habilidades sociais (capacidade de

interagir com outros agentes (e possivelmente humanos) a fim de poder satisfazer os próprios objetivos) (JENNINGS; LESPÉRANCE, 2000) (RUSSELL; NORVIG, 2003).

Os agentes podem ser definidos em categorias. Um desses são agentes puramente reativos. Esses agentes tomam decisões considerando apenas informações que estão no instante presente. Por consequência, o comportamento deles ocorre por respostas diretas ao ambiente (JENNINGS; LESPÉRANCE, 2000).

Outra categoria de agentes são aqueles que possuem estados. Essas entidades possuem uma dada estrutura de dados interna que são considerados quando agente toma uma certa decisão (JENNINGS; LESPÉRANCE, 2000).

Uma outra maneira de analisar os agentes se dá por meio das arquiteturas e modelos disponíveis para representar os agentes (tomada de decisão, estado interno). Para o propósito do estudo que está sendo apresentando neste texto, é o suficiente considerar de forma sucinta quatro dessas arquiteturas. A primeira consiste nos agentes baseados em lógica onde os agentes realizam deduções lógicas para tomar uma decisão (GENESERETH; NILSSON, 1987), a segunda arquitetura consiste nos agentes reativos que tomam decisões com base em um dado mapeamento de uma certa situação em uma dada ação (BONASSO et al., 1995), a terceira arquitetura é *BDI* cuja decisão ocorre da manipulação de estruturas de dados que representam crenças, desejos e intenções do agente (RAO; GEORGEFF, 1991) e a quarta arquitetura consiste em uma estrutura em camadas onde a tomada de decisão acontece por intermédio de diversas camadas abstração a cerca do ambiente (WOOLDRIDGE; JENNINGS, 1995) (JENNINGS; LESPÉRANCE, 2000).

## 2.2 ARTEFATOS

A definição de um artefato pode ser feita por analisar, antes, o comportamento de um agente. De maneira genérica, existe duas formas que podem ser usadas para caracterizar o comportamento de um agente, e essas são *goal-governed* e *goal-oriented* (CASTELFRANCHI et al., 2018) (RICCI et al., 2006).

Um agente que é caracterizado como *goal-governed* são aqueles que apresentam capacidades cognitivas, que podem representar os seus respectivos objetivos e, portanto, são capazes de definir seus objetivos em interesse (CASTELFRANCHI et al., 2018) (RICCI et al., 2006). Em contraste com isso, *goal-oriented* consiste nos agentes que são programados com a finalidade de alcançar um determinado objetivo (CASTELFRANCHI et al., 2018) (RICCI et al., 2006). Em um sistema multiagente muitas vezes uma dada entidade não é adequadamente

representada por nenhuma dessas duas categorias.

Assim sendo, artefatos são entidades que não se enquadram em *goal-governed*, não se enquadram em *goal-oriented*, são explicitamente projetados com o propósito de serem explorados pelos agentes para que possam alcançar seus objetivos individuais e sociais (RICCI et al., 2006) (RICCI et al., 2007).

Para ilustrar usando como referência a sociedade humana; se agentes (entidades autônomas) estão para seres humanos então artefatos são as ferramentas (não autônomas) que são usadas para uma determinada finalidade (ex. um marceneiro (agente), usa um martelo (artefato) para pregar um prego (artefato)) (RICCI et al., 2006).

Uma outra distinção entre agentes e artefatos se torna claramente explícita quando verificar sobre o ponto de vista conceitual e filosófico. Isso, pois agentes apresentam a capacidade de se comunicar com outros agentes, em contraste com isso artefatos não apresentam essa condição (RICCI et al., 2006).

Existem quatro elementos que podem ser usados para caracterizar um artefato (RICCI et al., 2006), que são; *interface de uso*, *instruções de operação*, *funcionalidade* e *estrutura-comportamento*.

A *interface de uso* consiste no conjunto de operações que podem ser invocadas pelo agente a fim explorar as suas funcionalidades (RICCI et al., 2006). A *instruções de operação* consiste na descrição de como fazer o uso das funcionalidades do artefato. Isso implica protocolos de uso das operações que devem ser invocadas pelo agente (RICCI et al., 2006). A *Funcionalidade* do artefato consiste no propósito definido pelo programador do sistema (RICCI et al., 2006). A *estrutura-comportamento* consiste nos aspectos internos do artefato, que é como o artefato é implementado a fim de providenciar as suas funcionalidades (RICCI et al., 2006).

Cartago (*Common "Artefacts for Agents" Open Framework*) é um framework usado para especificar as relações entre agentes e artefatos. Tendo em vista o fato de ser um dos principais *frameworks* na descrição das interações entre agentes e artefatos, não é possível falar no tema de artefatos em SMA sem ao menos comentar a estrutura conceitual presente no Cartago. Esse modelo é composto de três blocos, que são; *agent bodies*, *artifact* e *workspace* (RICCI et al., 2007).

*Agent bodies* é o que possibilita a inteligência do agente se relacionar com o ambiente. Para cada agente criado, há um *agent bodies* construído. Um *agent bodies* possui *effectors* (efetores) que tem o propósito de executar ações sobre o ambiente de trabalho e possui sensores (captam sinais do ambiente em sua volta). A relação completa entre agente-*agent*

*bodies*-artefato, no Cartago, é dada da seguinte forma; eventos observáveis são gerados pelos artefatos, sensores (componentes presentes no *agent bodies*) são sensibilizados, essa informação é transmitida para a inteligência do agente (esta, por sua vez, realiza os raciocínios e toma as decisões necessárias), o agente comunica ao *agent bodies* a ação que deve ser feita ao meio e, por último, através do *effectors* uma dada ação é produzida no ambiente de trabalho.

*Artifacts* (artefatos) são os tijolos básicos na gerência do Cartago. Cada artefato apresenta um nome lógico e número de identificação (id) único que são definidos pelo *artefat creator* no momento da instanciação. O nome lógico consiste num caminho ágil para o agente poder referenciar e compartilhar o respectivo artefato com os demais agentes. O id é requisitado na identificação dos artefatos quando uma dada ação é executada. Os artefatos também apresentam nome completo que inclui o nome do(s) *workspace(s)* onde está logicamente localizados (RICCI et al., 2007).

*Workspaces* é a localização lógica dos artefatos ocorre dentro de *workspaces*, que pode ser usado para definir a topologia do ambiente de trabalho. Neste âmbito, o *workspaces* são feitos com a finalidade de especificar nome lógico e id. Está dentro do escopo deste item definir uma topologia de ambiente possibilitando que uma estrutura de interação entre agentes e artefatos. Assim sendo o agente só pode usar um artefato que está no mesmo *workspace* onde ele está contido.

## 2.3 SISTEMA MULTIAGENTE

Um sistema multiagente(SMA) organizado é aquele constituído por agentes autônomos que interagem visando um propósito em comum tendo como consequência um comportamento global (HÜBNER et al., 2002) (ABBAS et al., 2015). Assim sendo, uma organização com essas características deve ser capaz de manifestar conhecimento em comum, cultura, memória, história, distribuição de atividades e a capacidade de distinguir um agente em específico (ABBAS et al., 2015). Deste fato é possível identificar o fenômeno "supra-individual" que implica em um comportamento que existe além dos comportamentos e atributos particulares no que diz respeito as entidades constituintes do sistemas.

Uma organização de um sistema multiagente deve conter relações sociais no que tange a agentes, institutos e grupos sociais (ABBAS et al., 2015). Ainda sobre isso, uma organização SMA deve apresentar uma *extensão de um espaço abstrato*. Isso implica uma representação dos seguintes conceitos; estrutura espacial, estrutura temporal, símbolos, semântica e capacidade de dedução. Há organizações que não se enquadram em todas essas restrições, contudo são

suficientes para tratar o problema dentro de uma perspectiva computacional (ABBAS et al., 2015).

O subtópico *Conceitos Gerais de uma Organização de Sistemas Multiagentes* tem como finalidade detalhar melhor os elementos presentes da Teoria da Organização dentro do contexto de SMA em relação a esse estudo. Já o Subtópico *Formalização de Conceitos Específicos para SMA* tem como por objetivo realizar uma verificação analítica dos elementos presentes no modelo de SMA denominando por *MOISE+*. Os conceitos que serão analisados são; objetivos, planos e papéis *organisation of multiagents system*.

### 2.3.1 CONCEITOS GERAIS DE UMA ORGANIZAÇÃO DE SISTEMAS MULTIAGENTES

A finalidade desta subseção consiste trabalhar com uma maior riqueza de detalhes todos os conceitos que constituem a ideia de uma organização de um sistema multiagente.

**Divisão em tipos de atividades:** Uma organização não é uniformemente estruturada. Isso, pois as atividades são distribuídas de forma desigual entre as diferentes entidades. Dentro do ponto de vista fenomenológico as atividades são sujeitas a classificação e ocorrem com diferentes frequências e em diferentes regiões dentro das definições espaciais da organização (ABBAS et al., 2015).

**Integração:** Dentro de uma organização ocorre a presença de interdependência entre diferentes espaços de atividades. Essas, por sua vez, estão relacionadas em uma estrutura única definida dentro de um contexto alinhado e integrado (ABBAS et al., 2015).

**Composição** Uma organização é composta por elementos menores. No caso dos multiagentes, os elementos atômicos que estruturam a organização são os agentes (ABBAS et al., 2015).

**Estabilidade/Flexibilidade:** Uma organização apresenta padrões de atividades. Esses padrões possuem características que podem ser enquadradas em dois aspectos; estáveis e flexíveis. No que tange as características estáveis, essas são constituídas por elementos/processos que definem o padrão em si mesmo. Em contraste com isso um comportamento flexível acontece quando o sistema é submetido a situações incomuns (ABBAS et al., 2015) ?.

**Coordenação:** Todo sistema é dependente de algum dado recurso. Assim sendo, se faz necessário que esse recurso seja utilizado de forma inteligente a fim de que possa se manter ao longo do tempo. Para que isso, se faz necessário que a organização se comporte como uma amplificadora de recursos a fim de que as estruturas operacionais tenham um comportamento

cada vez mais organizado (ASHBY, 1962), (FOERSTER, 2003), (LENDARIS, 1964). Contudo as incertezas relacionando aos efeitos combinados resultam influenciam negativamente nas eficiencias. Portanto, para manter a eficiência organizacional se faz necessário a existência de elementos otimizadores sobre os padrões de atividades (ABBAS et al., 2015).

**Recursividade:** Uma organização é constituída por sub-organizações. Isso ocorre em multiplos níveis de estrutura e se dá por intermédio de um padrão recursivo (ABBAS et al., 2015).

**Representação Multi-Nível e Causalidade:** Uma organização é estruturada por suborganizações em diferentes níveis estruturas. Isso, por sua vez, resultam em atividades ocorrendo em diferentes escalas espaciais, temporais e estruturais. Como consequencia disto, as cadeias causais presentes em estruturas organizacionais são processos multi-níveis (ABBAS et al., 2015).

**Potenciais e Diferenciais:** Diversos são os sistemas físicos onde as forças entre partículas são decorrentes de balanços de potenciais. Como esse comportamento está presente em diversos sistemas físicos, existe modelos abstratos de sistemas auto-organizaveis que levam em consideração a presença de forças potenciais e diferenciais em organizações (PRIGOGINE; NICOLIS, 1985). Esse conceito é trabalhado dentro de sistemas multiagentes. Um exemplo notório a respeito disto consiste no conceito de *Poder* o qual é entendido como a capacidade de influenciar uma dada organização (ABBAS et al., 2015)

**Regras e Gramáticas:** Organizações podem ser compreendidas como potenciais configurações de atividades e processos. Essas configurações podem ser descritas usando gramática (PENTLAND, 1995) (PENTLAND; RUETER, 1994). Tanto as gramáticas como as regras que compõem um organização apresentam três interpretações, essas são; como estruturas (especificações procedurais do que deve ser feito), como coação as ações definindo o que pode e não pode ser feito e como um compilado das experiências (ABBAS et al., 2015).

**Incerteza:** Não é possível conceber o conceito de uma organização sem ao menos entende-la como uma estrutura que distribui informação em si mesma. Sobre essa ótica, a distribuição de informação inequivocamente implica geração de incerteza o que por sua vez se manifesta como um complicante no que tange a comunicação entre as partes bem como a atividade organizacional em si mesma.



### 2.3.2 FORMALIZAÇÃO DE CONCEITOS ESPECÍFICOS PARA SMA

A apresentação desses conceitos será feita por intermédio de estudos relacionados ao *MOISE+*. Apesar de ser um *framework*, o *MOISE+* trata a rigor acadêmico na ótica da computação clássica a tratativa dada para os conceitos em interesse a esse estudo. Assim sendo, um estudo aprofundado do modelo, bem como dos textos em referência, satisfaz com exatidão os fundamentos teóricos para as análises em interesse.

A constituição do *MOISE+* é estruturada em três categorias de especificação, essas são; estrutural, funcional e deontica. O texto a seguir exibe com maior requie de detalhes cada uma dessas especificações.

A especificação estrutural acontece em três níveis, individual, social e coletivo. O nível individual trata de definir os papéis  $\rho$  dos agentes. Uma possível entre os papéis acontece por intermédio da hereditariedade em que se  $\rho'$  é filho de  $\rho$ . Isso implica afirmar que  $\rho'$  é uma especialização de  $\rho$ . Um exemplo apropriado para isso é o jogo de futebol onde existe o papel jogador dado por  $\rho$  e existe o papel atacante dado por  $\rho'$  (HÜBNER et al., 2002) (FERBER; GUTKNECHT, 1998) (FOX et al., 1996) (CARRON; BOISSIER, 2001). Em termos formais, essa relação é dada por;

$$\rho_a \sqsubset \rho_b$$

O nível social estabelece relações de ligação dado pelo predicado  $link(\rho_s, \rho_d, t)$ . Existe três possíveis valores para  $t$ , os quais são  $t = \{aut, com, acq\}$ . O valor *auth* significa autoridade (neste caso  $\rho_s$  exerce autoridade sobre  $\rho_d$ ), o valor *com* significa comunicação (neste caso  $\rho_s$  pode se comunicar com  $\rho_d$ ) e o valor *acq* significa conhecimento ( $\rho_s$  tem conhecimento da existência de  $\rho_d$ ) (HÜBNER et al., 2002) (HÜBNER et al., 2002) (CARRON; BOISSIER, 2001). O *MOISE+* define as seguintes relações de implicabilidade

$$\begin{aligned} link(\rho_s, \rho_d, auth) &\rightarrow link(\rho_s, \rho_d, com) \\ link(\rho_s, \rho_d, com) &\rightarrow link(\rho_s, \rho_d, acq) \end{aligned} \tag{1}$$

O modelo também determina como se dá as relações de hereditariedade para o predicado de *link*, é dado por (HÜBNER et al., 2002) (CARRON; BOISSIER, 2001);

$$\begin{aligned}
& link(\rho_s, \rho_d, t) \wedge \rho'_s \sqsubset \rho'_s \rightarrow link(\rho'_s, \rho_d, t) \\
& link(\rho_s, \rho_d, t) \wedge \rho'_d \sqsubset \rho'_d \rightarrow link(\rho_s, \rho'_d, t)
\end{aligned} \tag{2}$$

O nível coletivo determina a existência de compatibilidade entre os papeis (HÜBNER et al., 2002). Essa é uma relação reflexiva e transitiva de determina que se um papel  $\rho_a$  possui a capacidade de realizar um determinado objetivo, então o papel  $\rho_b$  também tem essa capacidade. Em termos formais, essa relação se dá da seguinte forma (HÜBNER et al., 2002) (CASTELFRANCHI, 1995).;

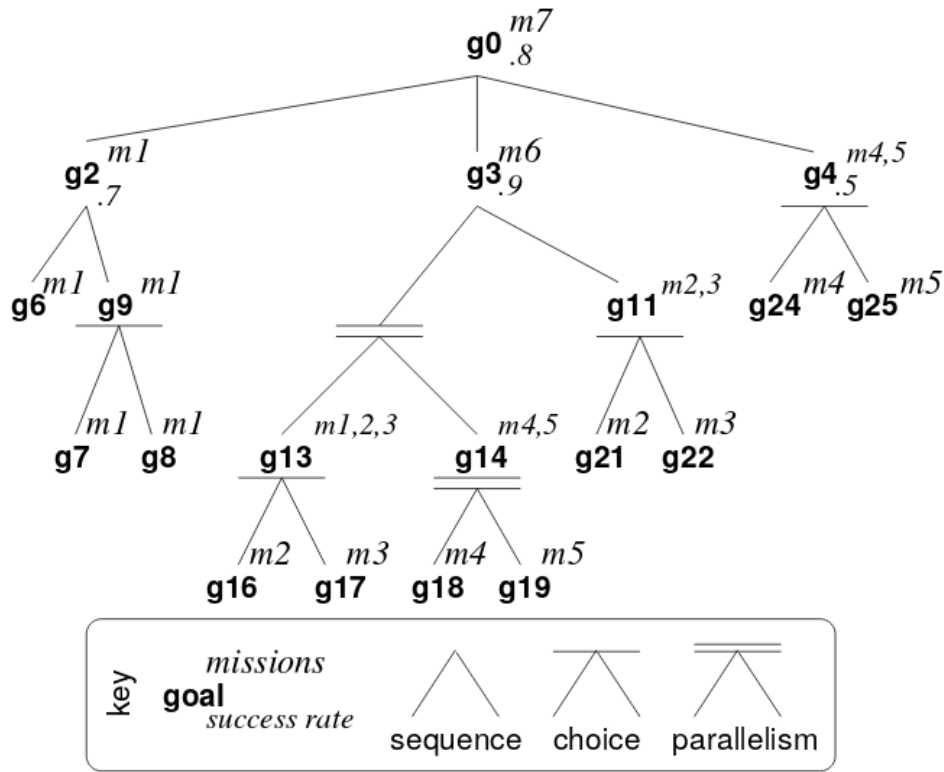
$$\rho_a \bowtie \rho_b \wedge \rho_a \neq \rho_b \wedge \rho_a \sqsubset \rho' \rightarrow \rho' \bowtie \rho_b$$

O nível coletivo também apresenta o conceito de grupo dado por  $gt$  e constituído por;

$$gt = \langle R, SG, L^{intra}, L^{inter}, C^{intra}, C^{inter}, np, ng \rangle$$

Em que  $R$  é o conjunto dos papeis não abstratos,  $SG$  são subgrupos que estão contidos neste grupo,  $L^{intra}$  consiste dos *links* intra-grupos,  $L^{inter}$  dos links inter-grupos,  $C^{intra}$  das relações de compatibilidade intra-grupos e  $C^{inter}$  das relações de compatibilidade inter-grupos. O símbolo  $np$  denota a cardinalidade mínima e máxima para uma dada função e o símbolo  $ng$  realiza o mesmo para os subgrupos (HÜBNER et al., 2002).

A Especificação Funcional tem como por finalidade descrever os objetivos a serem atingidos dentro de uma estrutura de árvore. A figura a seguir define como se dá esse tipo de especificação;



**Figura 1: Arvore de objetivos definido pelo modelo Moise (HÜBNER et al., 2002)**

A figura 1 define três tipos de relação de subobjetivos; *sequence* onde todos os subobjetivos devem necessariamente ser concluídos em sequência, *choice* onde o agente tem a possibilidade de escolher qual objetivo ele deseja seguir e *parallelism* onde todos os objetivos devem ser concluídos, contudo sem uma sequência definida (EXPLORING..., 1996) (SO; DURFEE, 1996). Esta parte do modelo é baseada em um *framework* de distribuição de atividades denominado por *TAEMS* (GARVEY; LESSER, 1996). Como é possível observar na figura, os objetivos são agrupados em conjuntos de missões *m* (CARRON; BOISSIER, 2001). A relação a seguir define isso melhor;

$$m_k = \{g_n, \dots, g_m\}$$

A Especificação Deontica define predicados para estabelecer permissões e obrigações entre os papéis e as missões. Toda obrigação implica necessariamente em uma permissão. A relação a seguir estabelece isso (HÜBNER et al., 2002) (CASTELFRANCHI, 1995);

$$obl(\rho, m, tc) \rightarrow per(\rho, m, tc)$$

$$obl(\rho, m, tc) \wedge \rho \sqsubset \rho' \rightarrow obl(\rho', m, tc) \quad (3)$$

$$per(\rho, m, tc) \wedge \rho \sqsubset \rho' \rightarrow per(\rho', m, tc) \quad (4)$$

$$(5)$$

Onde o predicado *obl* define uma obrigação e o predicado *per* define permissão. O argumento *tc* define uma periodicidade de tempo para o qual a relação deôntica é válida.

## 2.4 NORMAS

Quando se trata de normas em sistemas multiagentes é de crucial importância definir claramente este termo. Isso, pois há diversos estudos que tratam o conceito de norma sobre perspectivas diferentes. Por exemplo, os estudos (ESTEVA et al., 2002) (MOSES; TENNENHOLTZ, 1995) apresentam normas, em sistemas multiagentes, para representar a presença de sociedades, institutos e organizações. Há estudos que tratam normas como maneiras dos agentes trabalharem de forma coordenada com propósito de cumprir um objetivo global e também como uma maneira de obedecer as autoridades do sistema (LOPEZ; LUCK, 2003) (LÓPEZ; LUCK, 2004). No *MOISE+* normas são tratadas sobre a ótica da lógica deontica e é usada para especificar aos agentes com dado papel as suas obrigações em relação as missões (HÜBNER et al., 2002) (HÜBNER et al., 2002).

O estudo (DASTANI et al., 2009) apresenta uma linguagem formal para especificar sistemas multiagentes normativos. Essa linguagem contém os conceitos de normas que são os mesmos usados neste texto. A figura 2 apresenta a linguagem em notação EBNF (DASTANI et al., 2009).

```

N-MAS_Prog  := "Agents: " (<agentName> <agentProg> [<nr>])+ ;
              "Facts: " <bruteFacts>
              "Effects: " <effects>
              "Counts-as rules: " <counts-as>
              "Sanction rules: " <sanctions>;

<agentName> := <ident>;
<agentProg> := <ident>;
<nr>        := <int>;
<bruteFacts>:= <b-literals>;
<effects>    := ({<b-literals>} <actionName> {<b-literals>})+;
<counts-as>  := ( <literals>  $\Rightarrow$  <i-literals> )+;
<sanctions>  := ( <i-literals>  $\Rightarrow$  <b-literals> )+;
<actionName>:= <ident>;
<b-literals>:= <b-literal> {"," <b-literal>};
<i-literals>:= <i-literal> {"," <i-literal>};
<literals>   := <literal> {"," <literal>};
<literal>    := <b-literal> | <i-literal>;
<b-literal>  := <b-prop> | "not" <b-prop>;
<i-literal>  := <i-prop> | "not" <i-prop>;

```

**Figura 2:** Linguagem para descrever um programa de multiagentes normativos com a possibilidade de violações e sanções na notação EBNF segundo o texto (DASTANI et al., 2009). Nesta notação,  $\langle ident \rangle$  é usado para denotar uma *string* e  $\langle int \rangle$  inteiros. Os termos  $\langle b-prop \rangle$  e  $\langle i-prop \rangle$  são usados para designar dois tipos de conjuntos de proposições que são disjuntos entre si

Os *Facts* implementados em forma de *brute facts* definem os estados iniciais do sistema dentro de um ambiente compartilhado por todos os agentes. O termo *Effects*, implementado pro meio de *effects* define como se dá a transição de estados do sistema. A tag  $\langle actionName \rangle$  representa os eventos que geram transição dos estados. Uma norma, portanto, é definida em termos de *Counts-as rules*. O termo  $\langle counts-as \rangle$  aponta para transição entre  $\langle literals \rangle$  e  $\langle i-literals \rangle$ . Isso representa os fatos que resultam em violações. Portanto, em (DASTANI et al., 2009) as normas são descritas por intermédio de suas violações. Violação, dentro deste contexto, é dado como o descumprimento da norma (WRIGHT, 1969).

O termo *Sanction Rules* aponta para  $\langle sanctions \rangle$  e esses, por sua vez, para a transição entre  $\langle i-literals \rangle$  e  $\langle b-literals \rangle$  sendo que esses  $\langle i-literals \rangle$  advem de *Counts-as Rules*. Assim sendo, *Sanction Rules* define as consequências da violação. Essas consequências denotam caráter negativo ao agente tendo como foco uma natureza de ordem punitiva (DASTANI et al., 2009).

A figura 6 apresenta um programa escrito na linguagem proposta em

```

Agents:      passenger  PassProg  1
Facts:       {-at_platform, -in_train, -ticket}
Effects:      {-at_platform} enter {at_platform},
              {-ticket} buy_ticket {ticket},
              {at_platform, -in_train} embark {-at_platform, in_train}
Counts_as rules: {at_platform, -ticket} ⇒ {viol1},
                  {in_train, -ticket} ⇒ {viol1}
Sanction rules: {viol1} ⇒ {fined10}

```

**Figura 3: Um programa descrito na linguagem proposta neste estudo onde um agente representa um passageiro em uma estação de trem que pode entrar com ou sem um *ticket* na plataforma e no trem (DASTANI et al., 2009).**

Esse programa apresenta contém um agente chamado de *passenger* (especificações desse agente são detalhadas com maior rigor em *PassProg*). Os *Facts* deste programa são - *at\_plataform* (agente não está na plataforma), -*in\_train* (agente não está no trem) e -*in\_ticket* (agente não possui o -*in\_ticket*). As regras de *Effects* apresentam dois *< actionName >*. O primeiro é denominado por *enter* que tem por finalidade alternar entre os estados -*at\_plataform* e *at\_plataform*, ou seja, é uma ação onde o agente muda o estado de não está na plataforma para está na plataforma. O segundo estado é o *buy\_ticket* que gera a transição entre os fatos -*ticket* para *ticket*, ou seja - na ocorrência *buy\_ticket* o agente passa a ter um *ticket* (DASTANI et al., 2009).

O programa especifica duas regras em *Counts\_as rules*. A primeira regra denota a ocorrência de uma violação para um agente que entra numa plataforma se um *ticket*. A segunda define que uma violação acontece para o caso de um agente entrar em um trem sem um *ticket* (DASTANI et al., 2009).

O programa também define uma regra para *Sanction rules*. Essa regra se aplica a *viol<sub>1</sub>*, que - se verdade - então resulta no fato *fined<sub>10</sub>*, cuja semântica denota um pagamento no valor de 10 unidades de moeda (DASTANI et al., 2009).

## 2.5 RISCOS

O primeiro estudo teórico sobre acidentes de trabalho se dá no texto (RASMUSSEN, 1997). Essa pesquisa conclui que os erros em indústria não podem ser definidos apenas nas falhas de humanos, mas sim como consequência de um comportamento global da instituição. Ainda dentro deste âmbito, esse comportamento advém de uma forte pressão tendo em vista eficiência e otimização dos processos de produção (RASMUSSEN, 1997) (FADIER et al., 2003).

Com base neste entendimento, o estudo (FADIER et al., 2003) apresenta um

*framework* a fim de identificar as redes de causalidade que resultam em acidentes de trabalho. Assim sendo, a gestão de segurança se dá com base nos seguintes fatores;

- Políticas; *leis, diretrizes, padrões e regras*,
- Corporativo; *regras, estratégias, políticas internas, gerenciamento*
- Projeto de Equipamentos de Trabalho; *especificação, integração de segurança*

O item Política é mais relevante que os itens Corporativo e Projeto de Equipamentos de Trabalho. Esses dois últimos apresentam a mesma importância para uma estrutura de prevenção de acidentes bem sucedida.

A primeira análise a ser feita diz respeito ao nível Corporativo-Projeto de Equipamentos de Trabalho. Muitas vezes a equipe adota um atividades paleativas a fim de otimizar os processos de produção. Isso envolve assumir níveis de tolerância no que diz respeito ao desempenho e a segurança. Essa situação está dentro do conceito, para o *framework*, de *atividades limites*, isso pois tratam de situações que trabalham no limiar com os riscos. Assim sendo, as decisões feitas pelo profissional podem resultar muito facilmente em acidentes ou incidentes (FADIER et al., 2003). Dentro desta perspectiva que se apresenta o ator *BATU* - *Boundary Activities Tolerated during Use* (Atividades Limites Toleradas Durante o Uso).

Existe dois tipos de *BATU* que devem ser verificados tendo como base os processos de trabalho. Esses são; operacional e gerencial (administrativo - termo identificado no texto original; *managerial*). Aquele faz referência as atividades relacionadas a melhoria da produtividade com o propósito de resultar em melhorias de produtividade tendo em vista as metas de produção no que tange a produção, qualidade e segurança. Este diz respeito a decisões administrativas independentes dos processos operacionais mas que os impacta.

Outro conceito presente em (FADIER et al., 2003) é o de *Boundary Conditions Tolerated by Use* - *BCTU* (Condições Limites Toleradas Durante o Uso). O termo condição faz referência a uma situação, um estado, circunstâncias externas às quais pessoas ou até mesmo entidades são afetados no que diz respeito a uma certa decisão. Assim sendo, *BCTU* consiste em uma série de elementos e circunstâncias (ambiental, material, humana, produtos) que por conta de sua natureza ou de como se relaciona com as demais entidades e processos apresenta um certo potencial na geração de situações particulares, tendo em vista causas decorrentes de operações dinâmicas. Tanto os *BATUs* bem como os *BCTUs* não podem ser analisados diretamente, mas devem ser analisados por intermédio das ações e escolhas dos operadores e dos atores que constituem esse trabalho (FADIER et al., 2003).

Existe dois tipos de *BCTU*. O primeiro consiste no *BCTU* interno que se apresenta como uma concepção global de trabalhos e situações no que tange as relações de política da empresa. Neste concepção, *BCTU* interno faz referência as diferenças hierarquicas em termos de nível e decisões centrais. Em contraste com esse ponto, o *BCTU* externo aponta para o projeto da instalação. Como resultado, há o surgimento de quatro derivações, que são; soluções de segurança - funções de segurança (diz respeito as questões que podem fazer com que um dispositivo de segurança venha a falhar) , soluções técnicas - requisições de trabalho (quando as soluções técnicas são incompatíveis com as requisições de trabalho), modelo de projeto - modelo de instalação (se da quando a solução final não é ótima ou está degradada quando comparada com a solução inicial) e condições nominais preventivos - condições reais de operação (FADIER et al., 2003).

As relações entre *BATUs* e *BCTUs* são dinâmicas e são dependentes do processo. Para exemplificar, pode-se considerar o seguinte cenário; O projeto de uma máquina de dobra de papel obriga o operador a adotar uma dada posição que o faz assumir o riscos para acessar determinados pontos da máquina. Assim sendo, as escolhas do projeto da máquina (relacionada ao *BCTU*) não leva em consideração todos os aspectos relacionados a dinâmica profissional-máquina fazendo o que o profissional envolvido tenha que atuar dentro de um certo intervalo de tolerância no diz respeito a segurança profissional *BATU*.

## 2.6 LÓGICA MODAL

A lógica modal consiste em uma linguagem para tratar proposições que necessariamente ocorrem e proposições que possivelmente ocorrem. Os proposições dados como necessarios são aqueles que necessariamente são verdade. Por exemplo, A água sobre 1 atm e entre 0,1 °C - 99 °C se apresenta no estado líquido. O conceito de possibilidade é totalmente dependente do conceito de necessidade. Isso pois uma proposição possível é aquela que necessariamente não é falsa (GARSON, 2018).

A lógica modal é do tipo **K** e isso significa que nela está condita simbolos  $\sim$  para não,  $\rightarrow$  para "se ... então" e  $\Box$  para "Isto é necessário". De **K** e  $\Box$ , tem-se as seguintes regras;

Sendo que  $isTheorem(A, \mathbf{K})$  representa "Se A é teorema de **K**".

$$isTheorem(A, \mathbf{K}) \rightarrow \Box A \quad (6)$$



$$\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B) \quad (7)$$

O operador  $\Diamond$  apresenta o seguinte correspondente semântico; "Isto é possível". A relação entre  $\Box$  e  $\Diamond$  é dada pela regra que se segue.

$$\Diamond A = \sim \Box \sim A \quad (8)$$

As relações a seguir apresentam outras regras válidas para essa lógica;

$$\Box(A \wedge B) \rightarrow \Box A \wedge \Box B \quad (9)$$

$$\Box A \vee \Box B \rightarrow \Box(A \vee B) \quad (10)$$

$$\Box A \rightarrow A \quad (11)$$

$$\Box A \rightarrow \Box \Box A \quad (12)$$

$$\Diamond A \rightarrow \Box \Diamond A \quad (13)$$

$$\Box \Box \dots \Box = \Box \quad (14)$$

$$\Diamond \Diamond \dots \Diamond = \Diamond \quad (15)$$

$$A \rightarrow \Box \Diamond A \tag{16}$$

### **3 METODOLOGIA**

#### **3.1 ANÁLISE DOS MODELOS**

#### **3.2 ACOMPANHAMENTO DE PROFISSIONAIS EM ATIVIDADE DE RISCO**

#### **3.3 CONSTRUÇÃO DO MODELO**

#### **3.4 IMPLEMENTAÇÃO**

## 4 RESULTADOS

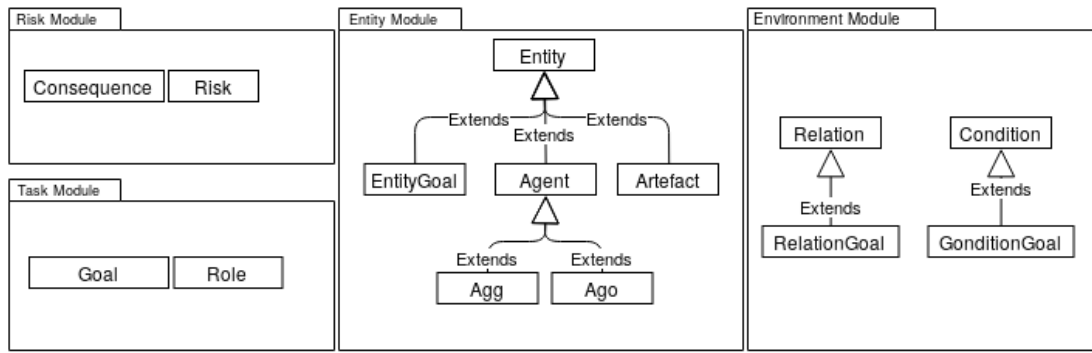
Essa capítulo tem como por propósito apresentar o modelo, sua utilização para um dado estudo de caso e sua implementação em Prolog (linguagem de programação com viés na Lógica Matemática). O modelo foi construindo usando Teoria de Conjuntos e Lógica de Predicado.

Os conjuntos são usados para representar um dado conceito. Os elementos de um conjunto representam os objetos atrelados ao conceito. Por exemplo, supondo que uma lógica de Carros venda os seguintes veículos; *Jetta*, *Gol*, *Uno*. Nesta situação, o conceito de carro é representado pelo conjunto  $C$  e os modelos são elementos do mesmo. Portanto, numa linguagem matemática formal tem-se a seguinte situação  $C = \{Jetta, Gol, Uno\}$ .

Dentro do conceito matemático, uma relação é uma correspondencia entre elementos de conjuntos não vazio, sendo dada por  $R \subseteq A \times B = \{(a, b) | a \in A \wedge b \in B\}$ . A Lógica de Predicados foi usada para representar essas relações. Para representação das regras foi feito uso das relações de inferência  $\rightarrow$  "Se ..., então ...".

O *UML* também é uma ferramenta que foi usada para criar representações do modelo. O propósito disto consiste nos seguintes aspectos; apresentar perspectiva global do modelo, definir melhor os critérios existenciais (agregação, composição), tornar o processo de apresentação mais didático e aproxima-lo de mecanismos de implementação (ex. linguagens de programação).

A figura 4 apresenta a estrutura de módulos (a fim de evitar poluição visual, as relações serão apresentadas em outra figura).



**Figura 4: A estrutura geral das classes do modelo**

Assim sendo, assumindo que existe  $\Omega_{Model}$  (um conjunto global onde todos os outros conjuntos do modelo estão contidos nele), os módulos são representados da seguinte maneira;

$$\Omega_{Model} = \{M_{Risk}, M_{Task}, M_{Entity}, M_{Environment}\} \quad (17)$$

A seção 4.1.1 seguirá exibindo toda a estrutura do modelo, trabalhando os módulos, conceitos, predicados, regras e raciocínios. A seção 4.2 exibe o diagrama *UML* de classes e o diagrama *UML* de atividades deste estudo. A seção 4.3 apresenta o caso de estudo escrito no modelo proposto neste estudo. A seção 4.4 exibe os raciocínios possíveis para o modelo implementado no estudo de caso. A seção 4.5 demonstra a modelagem do caso em estudo implementada em Prolog e exibe as soluções encontradas ao reproduzir os raciocínios presentes em 4.4

## 4.1 ESTRUTURA CONCEITUAL

Essa parte do texto é estruturada em; Módulo (exibe informações sobre os módulos e os conceitos neles contidos), Predicados (apresenta os predicados e justifica a sua existência) e as regras (exibe as regras e explica porque sua existência dentro do modelo é necessário).

### 4.1.1 MÓDULOS

A organização estrutural dos conceitos em módulos se justificativa por intermédio da grande quantidade de conceitos presentes dentro deste modelo. Um desses é o módulo da entidade *Entity Module* representado por  $M_{Entity}$ .

No mundo sobre o qual este modelo pretende representar trabalha com o fato de que tanto agentes como artefatos possuem algumas propriedades em comum, que é; existem, ocupam lugar no espaço, estão sujeitos ao tempo, apresentam estados e participam de processos. Essa premissa possuem os seus fundamentos alicerçados em 2.1 e 2.2 e isso será demonstrado com maior rigor no texto que se segue. Tendo em vista a ocorrência de certos conceitos necessários para lidar com essas questões, se fez necessário definir um módulo de entidades para agrupá-los em uma estrutura única. Esse módulo é composto pelos seguintes conceitos;

$$M_{Entity} = \{Entity, Agent, Artefact, EntityGoal, Agg, Ago\} \quad (18)$$

**Entity** - O termo entidade é sujeito a profundos debates filosóficos, porém neste texto o termo é usado para referenciar uma "coisa" que pode ser identificada, como uma pessoa, companhia ou um evento (CHEN, 1976). É dado que as propriedades anteriormente mencionadas caracterizam as "coisas" que podem ser identificadas, logo entidades. É digno de nota a existência de entidades que não se adequam a todas essas propriedades. Contudo, essas propriedades fazem referência ao que se caracteriza por entidade respeitando o conceito padrão (CHEN, 1976) e restringindo para o escopo deste modelo. Isso contempla tanto os agentes assim como artefatos, como fica claro na relação 19. O texto a seguir demonstra como essas propriedades se aplicam a agentes e a artefatos (se isso ficar demonstrado, logo fica demonstrado que são entidades).

*Existe* - implica numa propriedade que se justifica por si mesma, pois uma vez que o objecto ocorre, então existe. Tanto agentes como artefatos, existem dentro do mundo a ser representado e podem ser identificados.

*Ocupa lugar no espaço, estão sujeitos ao tempo* - Como definido em 2.1 e 2.2, ambos são situados em ambientes. Isso possibilita inferir que se faz necessário a presença de um conceito que se apresente como uma propriedade de estado para agentes e artefatos. Um ambiente, no contexto onde os agentes e artefatos são usados para representar atividades das pessoas, condiz com a relação de espaço e tempo.

*Participa de Processos* - Processos podem constituir entidades bem como entidades necessariamente constituem processos por intermédio a ação que aquelas manifestam nesses. No que se verifica ao primeiro caso, é possível usar o ser humano como exemplo - onde a entidade humano é formulada por uma série de processos bio-químicos. Sobre o segundo caso, relações climáticas explicam isso, onde a água é uma entidade presente em processos

termodinâmicos. Essa explicação justifica o texto; *Participam de Processos*.

*Apresenta estados* - O fato de que artefatos bem como agentes apresentam atributos (que podem mudar e podem assumir diferentes valores no que tange aos eventos externos e internos), então ambos também apresentam a concepção de estados (sendo esse termo usado diretamente em certos pontos dos textos presentes tanto em 2.1 2.2).

$$(Agent \cup Artefact) \subset Entity \quad (19)$$

**Agent** - Esse estudo adota a definição de agentes presentes no primeiro parágrafo da seção 2.1. Isso implica entidades autônomas, ou seja - que apresenta a capacidade de agir por si mesmo quando diante de condições onde isso é necessário. A seção 2.1, apresenta o conceito de agentes inteligentes e esse mesmo conceito é adotado neste modelo.

Não é preocupação deste estudo delimitar as representações do agente bem como definir algoritmos para verificar como se dá as relações de tomada de decisão. Assim sendo, fica em aberto para o modelador definir como se dá os processos de tomada de decisão, estados internos e modelos de representação que serão usados para definir o comportamento do agente.

**Artefact** - são entidade que existem para que os agentes possam cumprir com os seus objetivos e que apresentam interface de uso, instruções de operação, funcionalidade e estrutura-comportamento. Essas entidades não são orientados a objetivos e não apresentam capacidade de comunicação como definido na seção 2.2. Predicados que contemplam esses aspectos do artefato serão apresentados mais adiante ao decorrer do texto.

Os agentes são autônomos e orientados a objetivos sendo esses dois elementos descaracterizantes do que se definiu como por artefato. Logo, apesar de agentes e artefatos serem entidades, não é possível existir um agente que seja artefato ou um artefato que seja agente, o que é dado por 20.

$$Agent \cap Artefact = \emptyset \quad (20)$$

**EntityGoal** - Condiz a subconjuntos de **Entity** que, por intermédio de um predicado que será apresentado posteriormente, se relacionam com elementos do conjunto **Goal** (representam os objetivos). Assim sendo, consiste nas entidades necessárias que devem estar presentes no ato da execução de um certo objetivo para que este possa ser alcançado. Para exemplificar é possível conceber o seguinte cenário;

**Exemplo da Redação:** "O Professor Aristóteles definiu uma atividade; Escrever uma

redação sobre o livro Metafísica. Para isso, o aluno Alexandre o Grande deve escrever um dado texto, deve ler o livro sobre o tópico em definido, deve pegar uma folha, deve pegar um lapis e escrever a redação”.

Neste modelo, esse cenário pode ser especificado da seguinte forma;  $E = \{aristoteles, alexandre, folha, lapis, livro\}$ , em termos de objetivo (esse conceito será descrito com maior detalhe no texto em diante) há três  $G = \{g_0, g_1, g_2\}$  onde  $g_0$  corresponde ao ato do professor definir a atividade,  $g_1$  corresponde ao ato de ler o livro e  $g_2$  ao ato de escrever a redação. Então, é possível definir três subconjuntos de  $E$ , esses são os conjuntos **EntityGoal**,  $E_g = \{eg_0, eg_1, eg_2\}$ , onde  $eg_0 = \{aristoteles\}$   $eg_1 = \{alexandre, livro\}$  e  $eg_2 = \{aluno, folha, lapis\}$ . Em termos de relação, que será melhor trabalhado em partes futuras deste texto, considera-se que  $eg_1$  se relaciona com  $g_1$  e  $eg_2$  com  $g_2$ .

**Agg, Ago** - ambos corresponde ao conjunto de agentes que atingiram um determinado objetivo. Contudo, **Agg** faz referência aos gentes que atingiram o objetivo sem serem obrigados a isso, e **Ago** condiz aos agentes que atingiram o objetivo sendo obrigados a isso. Em um primeiro momento essa diferença para ser desnecessária, mas é relevante para criação de regras que serão apresentadas futuramente. A necessidade deste conjunto pode por ser demonstrada com o **Exemplo da Redação** onde antes de executar ambos os objetivos *Ago*, pois Alexandre e Aristóteles são obrigados a alcançar os objetivos. Então, é definido por  $Ago = \{ago_0, ago_1, ago_2\}$  onde  $ago_0 = \emptyset$ ,  $ago_1 = \emptyset$  e  $ago_2 = \emptyset$ . Ao atingir  $g_0$  o  $ago_0 = \{aristoteles\}$   $g_1$ , então  $ago_1 = \{alexandre\}$ , pois o aluno finalizou o primeiro objetivo. Ao término de  $g_2$ ,  $ago_2 = \{alexandre\}$ . Existe predicados que relacionam *Agg*, *Ago* com *Goal* e serão apresentados futuramente.

**O Módulo de Atividades** - *Task Module* representado por  $M_{Task}$  condiz com os conceitos relacionados aos objetivos que devem ser atingidos bem como aos papeis que são assumidos pelos agentes.

$$M_{Task} = \{Goal, GoalPreRequisite, Role\} \quad (21)$$

**Goal** - faz referência aos objetivos que devem ser atingidos pelos agentes. Os fundamentos semanticos deste conjunto dos estudos presentes em 2.3 2.3.2, mais especificamente do *MOISE*. Neste modelo, um objetivo é descrito em termos de *eg* e *rg*, que são as entidades e os relacionamentos (será explicado) que devem ser feitos para que o objetivo possa ser dado como concluído. Com o propósito de explorar as com maior granularidade as relações entre o agente e o objetivo, neste modelo o conceito de missão foi removido. Como será apresentado posteriormente, as relações deonticas entre os papeis se dão com o objetivo



e não com a missão. O estudo presente no *MOISE* também grafos para representar objetivos-subobjetivos. Esse modelo não importou a estrutura em grafos para descrever o comportamento de objetivos.

**Role** - apresenta o papel que um agente pode adotar dentro de um *SMA*. Esse conceito também é importado do *MOISE* 2.3.2 e define as relações deonticas entre os agentes e os objetivos. Para exemplificar, pode-se considerar o **Exemplo da Redação** onde existe dois agentes  $Agent = \{aristoteles, alexandre\}$ , existe dois papeis  $Role = \{professor, aluno\}$ . Neste caso, o agente *aristoteles* é o *professor* e o agente *alexandre* é o *aluno*.

**O Módulo de Ambiente** - *Environment Module* - consiste em conjuntos que representam relações e condições ambientes. Os conjuntos que descrevem esse ambiente são dados pela relação que se segue;

$$M_{Environment} = \{Relation, ReationGoal, Condition, ConditionGoal\} \quad (22)$$

**Relation** - Uma entidade estabelece relações com outras entidades ao seu redor (CHEN, 1976). No modelo proposto neste texto, os pesquisadores optaram por definir um conjunto que representa essas relações. Os pesquisadores optaram por um conjunto para representar os relacionamentos entre as entidades porque isso possibilita identifica-los. Isso facilita o desenvolvimento de raciocínos. O uso dos relacionamentos pode ser exemplificado por meio do **Exemplo da Redação**. Ao definir uma atividade, o professor aristoteles , que é uma entidade, estabeleceu um relacionamento com o seu aluno Alexandre, representado aqui por *relAristotelesAlexandre*. Para cumprir com essa tarefa o aluno teve de ler o livro - *relAlexandreLivro*, teve de pegar de pegar uma folha - *relAlexandreFolha*, teve de pegar um lapis *relAlexandreLapis* e teve de escrever a redação o que implica em uma relação entre lapis e folha *relLapisFolha*. Portanto, o conjunto de relacionamentos se dá da seguinte maneira;

$$M_{Environment} = \{relAristotelesAlexandre, relAlexandreFolha, relAlexandreLivro, relAlexandreLapis, relLapisFolha\}$$

Obviamente, cada entidade do grupo *Relation* tem uma vínculo com elementos do grupo *Entity*, como por exemplo *relAlexandreFolha* apresenta um dado vinculo com as entidades *Alexandre* e *Folha*. Há um predicado que trata disto e será exibido posteriormente. Assim como o conjunto *EntityGoal*, existe relacionamentos que devem estar presentes para

que dados objetivos possam ser atingidos, revelando - portanto - a necessidade de um conjunto para representar esse tipo de situação, que neste caso é *RelationGoal*. O exemplo em análise é descrito por três objetivos. O objetivo  $g_0$  corresponde ao ato do professor definir a atividade. Esse objetivo não pode ser cumprido sem relacionamento *relAristotelesAlexandre*. Portanto, existe  $rg_0$  associado a  $g_0$  onde  $rg_0 = \{relAristotelesAlexandre\}$  e que  $rg_0 \subset RelationGoal$ . O conjunto  $g_1$  está relacionado com  $rg_1$  e esse, por sua vez, é composto por  $\{relAlexandreFolha\}$ . O conjunto  $g_2$  está relacionado com  $rg_2$  e os elementos correspondem a  $\{relAlexandreLapis, relLapisFolha, relAlexandreFolha\}$ .

**Condition** - Esse conjunto representa as condições que devem ser mantidas para que um dado objetivo possa ser alcançado. Em analogia a *EntityGoal* e a *RelationGoal*, há um conjunto denominado de *ConditionGoal* que define essas condições para os objetivos. Para exemplificar o uso deste conjunto é possível considerar o **Exemplo da Redação**. Com certeza nenhum dos três objetivos se torna viável se não houver luz suficiente para que todos possam ver. O ambiente deve ser mantido em um certo silêncio, do contrário não há possibilidade do professor e do aluno exercer suas atividades intelectuais. Assim sendo é possível definir duas entidades;  $Condition = \{luz, silencio\}$ . Ambas as condições são válidas para todos os objetivos (para esse exemplo), portanto existe um  $cg_1 = \{luz, silencio\} | ConditionGoal = \{cg_1\}$ , sendo que  $cg_1$  estabelece relações com  $g_0$ ,  $g_1$  e  $g_2$ .

**Módulo de Risco** - *Risk Module* contem conjuntos que correspondem a conceitos relacionados a temática da segurança. O módulo de risco é dado pela relação que se segue;

$$M_{Risk} = \{Risk, Consequence\} \quad (23)$$

**Risk** - Na seção 4.1.1 o termo risco é usado para referenciar a um evento que apresenta um potencial de ocorrer e que gera consequências negativas as pessoas associadas quando acontece. Para exemplificar pode-se considerar uma condição onde um eletricista está trocando um disjuntor de um quadro elétrico. Nesse processo, o eletricista está sujeito ao risco de ser eletrocutado. Essas consequências negativas também são representadas por um conjunto, e esse é **Consequence**. O uso deste conjunto pode ser apresentada utilizando esse mesmo exemplo do eletricista, pois a consequência de se subter a um evento desses implica morte (nem sempre é assim, mas para efeitos didáticos pode-se considerar que o quadro elétrico é de certa potência que a morte é certa para o profissional que for eletrocutado).

#### 4.1.2 PREDICADOS

O predicado  $thereIsRelation(r_l, e_i, e_k) | r_l \in Relation \wedge e_i, e_k \in Entity$  é usado para tratar as questões de identificar a relação entre duas entidades com a sua relação. Esse predicado se lê da seguinte forma: O relacionamento  $r_l$  possui a entidade  $e_i$  e a entidade  $e_k$ . Para demonstrar como se dá o uso desse predicado pode-se considerar o **Exemplo da Redação**. A entidade *alexandre* apresenta uma relação com *folha* que é identificada como *relAlexandreFolha*. Portanto, com o predicado, a representação fica;  $thereIsRelation(relAlexandreFolha, alexandre, folha)$

O predicado  $hasRole(ag_n, \rho_m) | ag_n \in Agent \wedge \rho_m \in Role$  que tem sua origem nos estudos do *MOISE+* onde cada agente tem uma função dentro do contexto do *SMA*. Esse predicado se lê da seguinte forma: O agente  $ag_n$  tem um papel  $\rho_m$ . Usando o **Exemplo da Redação** tem-se o seguinte:  $hasRole(artistoteles, professor)$ .

O predicado  $hasObligation(\rho_m, g_j) | \rho_m \in Role, g_j \in Goal$  tem suas origens nos estudos da lógica deôntica também presentes no modelo *MOISE+*. Esse predicado pode ser lido da seguinte maneira: O agente que assumir o papel  $\rho_m$  tem a obrigação de concluir o objetivo  $g_j$ . No exemplo padrão deste texto (**Exemplo da Redação**), o primeiro objetivo é uma obrigação do professor, portanto:  $hasObligation(professor, g_0)$

O predicado  $hasPermission(\rho_m, g_j) | \rho_m \in Role, g_j \in Goal$  também está associado aos estudos da lógica deontica relacionada ao *MOISE+*. A leitura se dá desta maneira: O agente que  $\rho_m$  tem a permissão de concluir o objetivo  $g_j$ . Usando o exemplo padrão como base, onde *professor* tem a obrigação para executar  $g_0$ , então também tem permissão (isso será melhor explicado em uma regra que será apresentada mais tarde). Portanto;  $hasPermission(professor, g_0)$ .

O predicado  $isReached(g_k) | g_k \in Goal$  Todos os agentes, que eram obrigados a concluir o objetivo  $g_k$ , concluíram. A existência desse predicado se dá apenas ao fato de que em certos raciocínios é necessário indentificar que um dado objetivo foi atingido.

O predicado  $stopIn(g_n, ag_m) | g_n \in Goal, ag_m \in Agent$  apresenta a seguinte leitura: Toda a atividade foi encerrada no obeitivo  $g_n$  por uma ação associada ao agente  $ag_m$ . Para certos faciocínios se faz necessário identificar o encerramento das atividades como um todo e por quem isso aconteceu. Em certos casos não há a necessidade de identificar o agente  $ag_m$ . Para isso há uma outra versão deste mesmo predicado escrito da seguinte forma:  $stopIn(g_n)$  e sua semantica indica que a atividade foi encerrada do ojetivo  $g_n$ .

O predicado  $nextGoal(g_i, g_j) | g_i, g_j \in Goal$  possui a seguinte semantica: O objetivo  $g_i$

tem um próximo o objetivo que é  $g_j$ . Sua necessidade advém do fato de que este modelo trabalha os mesmos conceitos presentes em *MOISE+* porém sobre uma abordagem diferente. Em vez de usar estrutura de super-sub objetivos e definindo operadores de série e paralelo, os objetivos não possuem estruturas e suas relações se dão por um objetivo apontado para o próximo. Então, para exemplificar pode-se considerar uma atividade descrita por quatro objetivos, sendo que  $g_0$  é pré-requisito para  $g_1$  e  $g_2$ . Em contrapartida  $g_3$  só começar a ser atingido depois da finalização de  $g_1$  e  $g_2$ . Assim sendo, na linguagem proposta neste estudo, esse problema é escrito da seguinte forma:  $nextGoal(g_0, g_1), nextGoal(g_0, g_2), nextGoal(g_1, g_3), nextGoal(g_2, g_3)$ .

O predicado  $hasCondition(g_i, cg_n) | g_i \in Goal, cg_n \subset GoalCondition$  é lido da seguinte forma: O objetivo  $g_i$  só pode ser iniciado se todas as condições presentes em  $cg_n$  estiverem presentes quando o agente iniciar a tentativa de alcançá-lo. O uso deste predicado pode ser exemplificado por meio do exemplo padrão. Tendo em vista que o objetivo  $g_0$  pode começar apenas se as condições  $cg_1 = \{luz, silencio\}$ , então para este caso, é possível escrever o seguinte predicado:  $hasCondition(g_0, cg_1)$ .

O predicado  $hasEntity(g_i, eg_m) | g_i \in Goal, eg_m \subset EntityGoal$  é lido da seguinte forma: Para que o objetivo  $g_i$  seja atingido, as entidades presentes em  $eg_m$  devem estar presentes. O propósito deste predicado reside na necessidade de identificar quais são as entidades que devem estar presentes para que um dado objetivo possa ser executado. No **Exemplo da Redação**, um objetivo  $g_2$  só pode ser atingido se as entidades  $eg_2 = \{aluno, folha, lapis\}$  estiverem presentes. Para essa situação, esse predicado é escrito da seguinte forma  $hasEntity(g_2, eg_2)$ .

O predicado  $hasRelation(g_i, rg_m) | g_i \in Goal \wedge rg_m \subset RelationGoal$  é lido da seguinte forma: Para que o objetivo  $g_i$  seja atingido se faz necessário a realização de todos os relacionamentos contidos em  $rg_m$ . Dentro do exemplo padrão, o objetivo  $g_2$  só pode ser atingido se os relacionamentos  $rg_2$  estiverem presentes. Portanto, a especificação para esse situação é escrita da seguinte forma:  $hasRelagion(g_2, rg_2)$ .

O predicado  $isPresent(X) | X = cg_n \vee X = c_k \vee X = rg_k \vee X = r_k \vee X = eg_k \vee X = e_k$ . Esse predicado é lido da seguinte forma:  $X$  está presente neste instante. Em alguns raciocínios e de impotancia verificar se um elemento ou um conjunto de elementos está presente durante a tentativa de um dado agente tentar executar algum objetivo.

O predicado  $tryReach(ag_i, g_j) | ag_i \in Agent \wedge g_j \in Goal$  é lido da seguinte forma: Um dado agente  $ag_i$  está tentando atingir o objetivo  $g_j$ . Para algumas situações é de crucial importancia identificar quando um agente está tentando atingir um dado objetivo, fazendo-se necessário a existência de um predicado apenas para esse propósito. Para exemplificar pode-se considerar o exemplo presente no quadro do professor Aristoteles com o seu aluno Alexandre.

Para que o professor *Aristoteles* possa alcançar o objetivo  $g_0$ , ele precisa tentar fazer isso. Neste modelo, essa situação é representada da seguinte maneira:  $tryReach(aristoteles, g_0)$ . Seguindo a linha desse predicado, há também o  $ableTryReach(ag_i, g_j) | ag_i \in Agent \wedge g_j \in Goal$  cuja semântica expressa que o agente  $ag_i$  está habilitado para tentar buscar o objetivo  $g_j$ . Contudo, não significa que o agente fará o correspondente de  $tryReach(ag_i, g_j)$ . O que definirá a transição de estado: habilitado a alcançar um dado objetivo para o estado: tentando alcançar um dado objetivo consiste em estados internos do agente. Contudo, não é do interesse deste estudo aprofundar na dinâmica do agente em si - deixando esse processo em aberto para o programador decidir como resolverá essa questão.

O predicado  $violationCondition(ag_i, g_j, c_k)$  sendo que  $ag_i \in Agent \wedge g_k \in Goal \wedge c_k \in Condition \wedge c_k \in cg_n \wedge cg_n \subset ConditionGoal$ . Esse predicado deve ser lido da seguinte maneira: O agente  $ag_i$  comete uma violação de condição no objetivo  $g_j$  por tentar realizar uma determinada atividade em que a condição  $c_k$  era essencial porém não estava presente. Os fundamentos deste predicado está vinculado com a seção 2.4. No modelo em 2.4 para representar aspectos normativos dos agentes, a regra do tipo *Count-as* apresenta quais são as circunstâncias que ocasionam numa violação. Esse predicado cumpre com esse propósito para o conjunto **Conditions**. Para exemplificar, no **Exemplo da Redação**, se o professor Aristoteles lecionar sem que haja luz suficiente para isso, então ele cometeu uma violação de condição caracterizada da seguinte maneira:  $violationCondition(aristoteles, g_0, luz)$

O predicado  $violationRelation(ag_i, g_j, r_k)$  sendo que  $ag_i \in Agent \wedge g_k \in Goal \wedge c_k \in relation \wedge r_k \in rg_n \wedge rg_n \subset RelationCondition$ . Esse predicado possui a mesma situação presente em  $violationCondition(ag_i, g_j, c_k)$  contudo o foco diz respeito aos relacionamentos. A leitura se dá desta forma: O agente  $ag_i$  pratica uma violação de relação no objetivo  $g_j$  por executar a atividade sem que a relação  $r_k$  esteja presente. O uso deste predicado por ser feito considerando o exemplo em análise com a adição de uma breve descrição de uma situação que possa ocorrer que é o seguinte; A ponta do lápis que Alexandre tenta usar para escrever a redação está quebrada. Portanto, a relação  $relLapisFolha$  não pode ser feita. Se o agente *Alexandre* alcançar o objetivo  $g_2$  sem ter as circunstâncias necessárias para isso, então cometeu uma violação de relação sendo escrito da seguinte forma:  $violationRelation(alexandre, g_2, r_2)$ .

O predicado  $violationEntity(ag_i, g_j, e_k)$  sendo que  $ag_i \in Agent \wedge g_j \in Goal \wedge e_k \in Entity \wedge e_k \in eg_n \wedge eg_n \subset EntityGoal$ . Esse predicado advém das mesmas situações dos dois predicados acima. A leitura deste predicado se dá da seguinte forma: O agente  $ag_i$  cometeu uma violação por tentar alcançar o objetivo  $g_j$  sem que a entidade  $e_k$  esteja presente. No exemplo padrão o uso deste predicado consiste em Alexandre tentar executar  $g_2$  sem ter a entidade *lapis*

o que resulta no seguinte:  $\text{violationEntity}(\text{alexandre}, g_2, \text{lapis})$ .

O predicado  $\text{hasRisk}(X, \text{risk}_j, \text{cs}_k) | \text{risk}_k \in \text{Risk} \wedge \text{cs}_k \in \text{Consequence} \wedge (X = c_k \vee X = r_k)$  é baseado nos estudos presentes na seção 2.5 e tem a finalidade de finir os riscos associados a tentar executar alguma atividade sem que  $c_k$  ou  $r_k$  esteja presente. Portanto, a leitura deste predicado é dada da seguinte forma; A ocorrência de uma violação onde  $X$  está envolvido ocasiona em um evento associado ao  $\text{risk}_j$  com a consequência  $\text{cs}_k$ . Ao explicar sobre o conjunto **Consequence**, foi dado um exemplo sobre um eletricitista que tem o potencial de ser eletrocutado. Para esse exemplo, o uso deste predicado apresenta o seguinte formado:  $\text{hasRisk}(\text{relFerramentaIsolanteBarramento}, \text{eletrocutado}, \text{morte})$  que traduz o ato do eletricitista não efetuar o relacionamento de uma ferramenta isolante para entrar em contato com o barramento  $\text{relFerramentaIsolanteBarramento}$ , é eletrocutado e acaba se submetendo a *morte* por conta disto.

O predicado  $\text{possibilityHappensBadEvent}(r_l) | r_l \in \text{Relation}$  tem seus fundamentos associados ao estudo da lógica modal, presente na seção 2.6, no operador  $\Diamond$  cuja semântica denota possibilidade. Contudo, neste estudo esse termo apresenta o seguinte conceito semântico: há a possibilidade de acontecer um evento ruim associado ao risco  $r_l$  vinculado ao agente que está associado a essa relação mesmo que esse agente não tenha cometido nenhum erro durante o procedimento. Esse predicado tem como finalidade representar situações onde um evento ruim acontece não pelo erro do profissional diretamente associado a situação, mas sim por outras cadeias causais complexas de serem identificadas e justamente por isso são abstraídas por conceitos de aleatoriedades, idem possibilidades. Para exemplificar pode-se considerar a situação onde um eletricitista de linha viva usa um bastão isolante para acessar um barramento altamente energizado. Contudo, esse bastão pode estar com isolamento comprometido. Em testes que deveriam ser feitos antes do eletricitista fazer uso desta ferramenta, esse comportamento pode ser revelado eliminando qualquer possibilidade de que os riscos venham a se tornar eventos reais, pois se o bastão estiver em bom estado então o eletricitista não se envolverá em um acidente por esse fator, se o bastão estiver em mal estado - isso será identificado e a ferramenta será adequadamente substituída. Contudo, considerando um cenário onde por negligência de profissionais a medição não é feita, surge uma possibilidade do isolamento estar comprometido. Essa situação torna verdadeira o seguinte predicado  $\text{possibilityHappensBadEvent}(\text{relBastaoBarramento})$ .

O predicado  $\text{affectsOtherRelations}(r_k, r_n) | \{r_k, r_n\} \subset \text{Relation}$  trabalha em conjunto com  $\text{possibilityHappensBadEvent}(r_l)$ . Esse predicado se lê da seguinte forma: Se  $r_k$  não foi realizado, ou se for realizado de forma inapropriada, isso afeta  $r_n$  tornando verdadeira

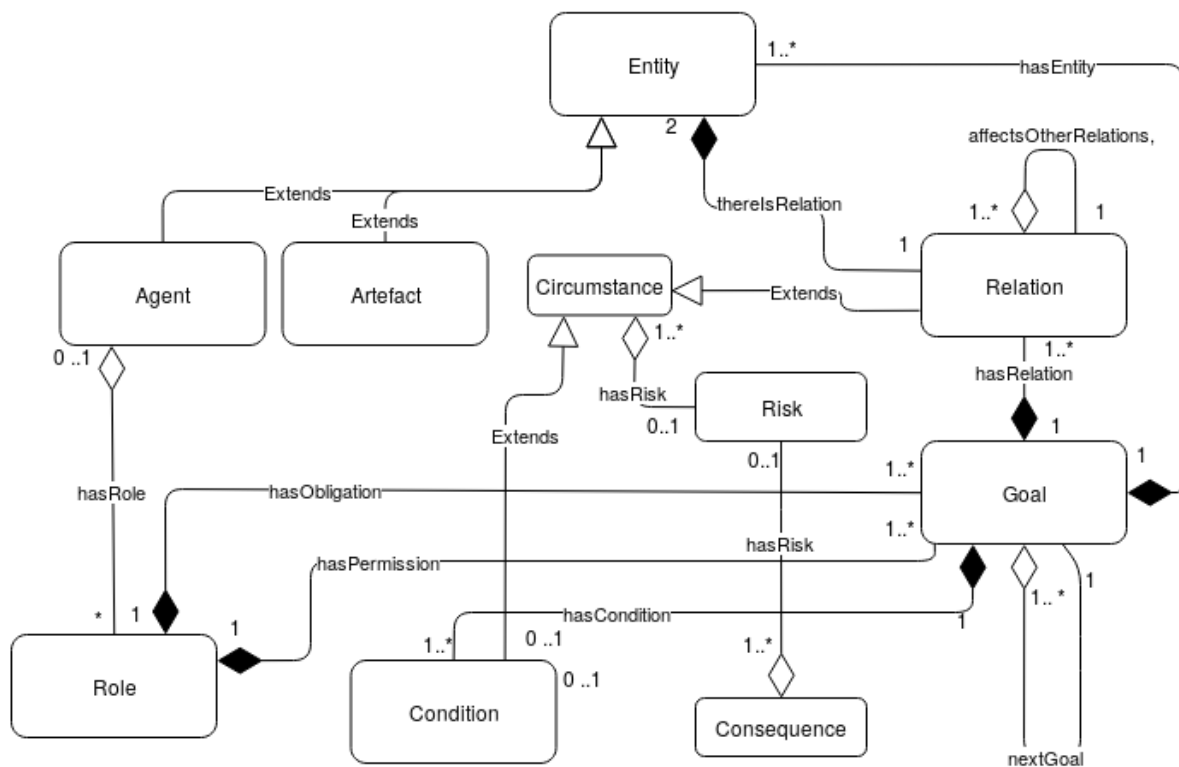
*possibilityHappensBadEvent(r<sub>n</sub>)*. Ambos são importantes em raciocínios onde se deseja mostrar que a não execução de um dada relação não gera consequências negativas imediatas a ninguém, mas resulta em consequências futuras inclusive sobre pessoas que não compartilham da mesma situação. No exemplo do eltricista, a relação *relBastaoBarramento* é afetada pela não execução da relação *relBastaoMedidor* que define a relação entre o aparelho medidor de corrente de fulga e o bastão isolante, sendo este evento escrito da seguinte maneira: *affectsOtherRelations(relBastaoMedidor,relBastaoBarramento)*

O predicado *consequenceOfBadEvenet(g<sub>k</sub>,ag<sub>i</sub>,risk<sub>k</sub>,cs<sub>m</sub>)* pode ser lido da seguinte maneira; ocorreu um evento ruim no objetivo *g<sub>k</sub>* associado ao agente *ag<sub>i</sub>* e associado ao risco *risk<sub>k</sub>* que atribuiu ao agente *ag<sub>i</sub>* consequências *cs<sub>m</sub>*. Esse predicado tem por finalidade trazer semanticamente as consequências ruins sobre alguém quando ocorre o pior caso. Sobre o exemplo do eletricitista que acessa um barrameto de alta tensão, esse predicado é escrito da seguinte forma: *consequenceOfBadEvenet(g<sub>acessoBarramento</sub>,eleticista,eletrocutado,morte)*. Associado a isso há o predicado *happensBadEvent(r<sub>m</sub>)* cujo propósito define que um evento ruim aconteceu no relacionamento *r<sub>m</sub>*.

O predicado *lastGoal(g<sub>i</sub>,ρ<sub>m</sub>)|g<sub>i</sub> ∈ Goal,ρ<sub>m</sub> ∈ Role* apresenta um ultimo objetivo *g<sub>i</sub>* que deve ser alcançado por agentes com determinado papel *ρ<sub>m</sub>*. Esse predicado tem sua existência justificada em certos raciocínios, que serão demonstrados mais adiante, onde esse tipo de informação é relevante.

#### 4.1.3 DIAGRAMA DE CLASSES

Diferente da figura 4, o foco da figura 5 consiste em apresentar como se dá a estrutura de classes e dos relacionamentos. Ambas situações poderiam ser representadas em uma única figura, contudo os pesquisadores decidiram por seccionar em duas a fim de tornar o processo mais didático. Por esse mesmo motivo não está apresentado neste *UML* todas as classes e propriedades.



**Figura 5: Diagrama de classes do Modelo**

Uma vantagem deste tipo de diagrama em relação a representação por conjuntos consiste na ocorrência de uma semântica específica para tratar dois pontos relevantes dentro do contexto computacional: cardinalidade e relações existenciais. Um dos predicados interessantes de serem analisados, neste contexto, é *hasRole* que define um relacionamento fracamente agregado entre *Agent* e *Role*. Isso, pois (dentro do escopo deste modelo) um agente pode existir sem ter um papel, portanto este não é um critério necessário para definir aquele. A cardinalidade se justifica tendo como base o fato de que um agente pode ter um ou mais papéis.

As relações *hasObligation*, *hasPermission* se dão por meio de agregações fortes tendo vista que não há sentido semântico para um papel  $\rho$  existir sem que esteja vinculado a ao menos um objetivo. Como um papel se relaciona com diversos objetivos, os engenheiros adotaram a cardinalidade de 1 para 1..\*.

A relação *hasRisk* ocorre entre *Consequence*, *Condition*, *Relation* e *Risk*. Contudo, uma esse relacionamento não se dá entre as quatro classes no mesmo instante, sendo assim: *Consequence*, *Condition*, *Risk* ou assim: *Consequence*, *Relation*, *Risk*. A representação desta situação presente na subseção 4.1.2 se deu de forma relativamente simples, criando uma variável que poderia assumir instâncias de *Relation* ou de *Condition*. Contudo, isso não se aplica em UML. Para resolver esse problema, os engenheiros criaram uma classe chamada *Circumstance*,



que é pai de *Relation* e de *Condition*. Essa classe não possui sentido semântico dentro do modelo e sua existência se deve apenas para resolver esse problema neste tipo de linguagem de representação. De 0 a 1 risco pode ter 1 ou mais relacionamentos com *Circumstance* e o mesmo ocorre com *Consequence*.

Um objetivo não pode ser definido sem saber quais são as entidades *Entity*, relações *Relation* e consequências *Consequence* necessários para que seja alcançado. Por isso os predicados *hasCondition*, *hasEntity* e *hasRelation* estabelecem composição forte de suas respectivas classes com *Goal*. Como um objetivo pode apontar para diversas instancias dessas classes, os pesquisadores optaram - para cada uma das relações - trabalhar com a cardinalidade  $1 - 1..*$ .

No modelo roda relação *Relation* deve estar relacionada com duas entidades. Por esse motivo o predicado *thereIsRelation* faz composição forte com *Entity* e a sua cardinalidade é dada  $1 - 2$ .

A classe *Goal* possui uma relação consigo mesma dada por *nextGoal*. Essa é uma agregação fraca, pois do contrário seria impossível haver uma única instância desta classe. Isso se deve ao fato de que a primeira instância necessitaria de uma instância de *Goal* para existir. Contudo, como não há um elemento de *Goal* antes do primeiro elemento de *Goal*, logo esse primeiro elemento não pode existir. Um objetivo poder ter como próximo um ou mais objetivos, justificando a ocorrência da cardinalidade  $1..n$ .

O predicado *affectsOtherRelations*, por motivos similares a *nextGoal* deve ter agregação fraca. Como uma relação pode afetar uma ou mais, a cardinalidade adequada para essa circunstância é dada por  $1 - 1..*$

#### 4.1.4 REGRAS

A regra 24 tem os fundamentos teóricos na lógica deontica e em modelos como *MOISE+*. Assim sendo, todas as relações de obrigação implicam relações de permissão. O que essa regra determina consiste no fato de que se um agente  $g_j$  é obrigado a trabalhar sobre o objetivo  $g_j$ , então esse agente também tem a permissão de para trabalhar sobre o objetivo  $g_j$

$$\begin{aligned} hasObligation(\rho_m, g_j) &\rightarrow hasPermission(\rho_m, g_j), \\ \rho_m &\in Role \wedge g_j \in Goal \end{aligned} \tag{24}$$

As regras 25, 26, e 27 são fundamentadas em 2.4 onde o conceito do que pode ou não pode ser feito é definido em termos das regras *Count-as*. Essas regras determinam quais são os elementos que resultam em violação. Inspirando nesse tipo de estrutura é que os pesquisadores trataram de trabalhar as referidas regras.

A complexidade de estudo é extramente ampla, e com certeza existe mais tipos de violações do que as consideradas a seguir, contudo optou-se por estudar essas violações porque são essenciais para os objetivos deste estudo. Outro questionamento que pode surgir consiste no porque definir três tipos de violações? Isso reside no fato de que essas violações resultam em consequências diferentes, por conta disto em um primeiro momento os engenheiros do modelo decidiram trata-las em estruturas diferentes.

A explicação das regras será feita sempre analisando o sentido semântico do predicado que é implicado e em relação aos estudos pelos quais elas se fundamentam. Partindo desta permissa o entendimento da relação 25 só pode ser feito na ocorrência de uma investigação sobre quais são os elementos que semanticamente correspondem ao predicado  $violationCondition(ag_m, g_i, c_k)$ . O primeiro ponto reside em verificar quais são as condições necessárias de  $g_i$ . Quem tem essa finalidade é o predicado  $hasCondition(g_i, cg_n)$ . Contudo, saber todas as condições não são o suficiente, pois a violação acontece na ausência de uma condição  $c_k$  e isso deve ser verificado nesta relação de implicabilidade. Então se faz necessário considerar um predicado que analisa se  $c_k$  está presente no ato da manutenção e é com esse propósito que  $isPresent(c_k)$  faz parte da relação. Contudo, as informações ficam desconstruídas se  $c_k$  não estiver contido em  $cg_n$ , por isso é de importância fazer essa análise também. Para isso, neste estudo, os pesquisadores fizeram uso de um operador de conjuntos cujo símbolo é  $\in$  que nessas relações tem a finalidade de verificar se  $ck$  está contido em  $cg_n$ . Esses são componentes essenciais, porém não são suficientes porque não consideram a condição do agente. Isso, pois afirmar sobre a ocorrência de uma violação sobre um agente sem considerar se ele está efetivamente tentando alcançar um objetivo consiste em desconsiderar a semântica daquilo que está sendo implicado. Isso é resolvido por considerar o termo  $tryReach(ag_m, g_i)$ .

$$\begin{aligned}
 &hasCondition(g_i, cg_n) \wedge \neg isPresent(c_k) \wedge (c_k \in cg_n) \wedge tryReach(ag_m, g_i) \rightarrow \\
 &\quad violationCondition(ag_m, g_i, c_k) \\
 &g_i \in Goal, cg_n \subset ConditionGoal, c_k \in Condition, ag_m \in Agent \quad (25)
 \end{aligned}$$

O propósito da regra 25 quando definido em termos de linguagem natural tem o propósito de exprimir o seguinte: Se um agente tentar executar um dado objetivo sem que haja

todas as condições ambientes necessárias para isso, então esse agente comete uma violação de condição neste respectivo objetivo.

A regra 26 define as condições que resultam em uma violação de relação. O predicado  $violationRelation(ag_m, g_i, r_k)$  considerada que a violação se dá por um agente  $ag_m$  em um objetivo  $g_i$  na relação  $r_k$ . Portanto, para respeitar a semântica deste predicado se faz necessário considerar ao menos um termo que vincule o objetivo  $g_i$  com a relação  $r_k$ . Para esse propósito é que se considera o termo  $hasRelation(g_i, rg_n)$  pois define quais são as relações que devem estar presentes para que o objetivo  $g_i$  possa ser alcançado. Contudo, só isso não é o suficiente, pois se faz necessário analisar se  $r_k$  está presente ou não e se pertence ao conjunto  $rg_n$ . A primeira situação é resolvida considerando o termo  $isPresent(r_k)$  e a segunda situação é resolvida verificando se  $r_k \in rg$ . A semântica de  $violationRelation$  só é conservada em sua inteireza se a presença do agente também for analisada. Para esse propósito é que se verifica a necessidade do uso de  $tryReach(ag_m, g_i)$  que deverá retornar se o agente está tentando alcançar o objetivo  $g_i$ .

$$\begin{aligned}
 &hasRelation(g_i, rg_n) \wedge \neg isPresent(r_k) \wedge (r_k \in rg_n) \wedge tryReach(ag_m, g_i) \rightarrow \\
 &\quad \quad \quad violationRelation(ag_m, g_i, r_k) \\
 &g_i \in Goal, rg_n \subset RelationGoal, r_k \in Relation, ag_m \in Agent \quad (26)
 \end{aligned}$$

Traduzindo a regra 26 para linguagem natural obtém-se a seguinte expressão: Se um agente tentar alcançar um dado objetivo sem que todas as relações necessárias para isso estejam presentes (considerando as relações sobre o domínio dele, tal como manuseio de uma dada ferramenta específica, e considerando as relações que são independente dele), então esse agente comete uma violação de relação.

A regra 27 tem o propósito de definir quais são as condições que resultam em uma violação de entidade. O predicado  $violationEntity(ag_m, g_i, e_k)$  define semanticamente que uma violação de entidade corresponde em relação a um dado  $ag_m$  em um  $g_i$  vinculado a entidade  $e_k$ . Assim como nas demais regras de violação, o sentido semântico deste termo considera a necessidade de verificar a existência de uma entidade necessária para um dado objetivo, por isso é essencial verificar o predicado  $hasEntity(g_i, eg_n)$ , verificar se  $e_k \in eg_n$  e verificar se está ou não presente no momento da manutenção  $isPresent(e_k)$ . A semântica do predicado também considera o momento em que o agente está atuando sobre o objetivo  $g_i$ , por isso o predicado  $tryReach(ag_m, g_i)$  também é posto na relação de implicabilidade.

$$\begin{aligned}
& hasEntity(g_i, eg_n) \wedge \neg isPresent(e_k) \wedge (e_k \in eg_n) \wedge tryReach(ag_m, g_i) \rightarrow \\
& \quad violationEntity(ag_m, g_i, e_k) \\
& g_i \in Goal, eg_n \subset EntityGoal, e_k \in Entity, ag_m \in Agent
\end{aligned} \tag{27}$$

Em termos de linguagem natural, 27 se apreseta da seguinte forma: Se um agente tentar alcançar um dado objetivo sem ter todas as entidades presentes para isso, então esse agente cometeu uma violação de entidade.

As regras 28 e 29 são inspiradas nos estudos presentes na seção 2.4 onde as consequências de uma violação são definidas como sanções no que é denominado por *SanctionRule*. A estrutura dessas regras, em 2.4 e em (DASTANI et al., 2009) é dada como *violation*  $\rightarrow$  .... Contudo, este estudo leva em consideração não apenas o termo que se refere a violação, mas também as circunstâncias que são consideradas juntos, que neste caso advêm do predicado *hasRisk*. Assim como em 2.4, o modelo deste estudo define que uma sanção corresponde a uma penalidade que o agente deve pagar. Na estrutura da problemática em análise, a penalidade ocorre pelo fato do agente sofrer fisicamente os efeitos dos seus erros. Esse comportamento é dado pelo predicado *consequenceOfBadEvent*( $g_i, ag_m, risk_j, cs_m$ ) cujo correspondente semantico define que o agente  $ag_m$  sofre sobre o evento associado em  $risk_j$ , no objetivo  $g_i$  a consequência  $cs_m$ . Se os engenheiros deste modelo considerarem apenas *violationCondition*( $ag_m, g_i, c_k$ ) para a relação 28 e *violationRelation*( $ag_m, g_i, r_k$ ), o correspondente semantico de *consequenceOfBadEvent*( $g_i, ag_m, risk_j, cs_m$ ) é desrespeitado, não especificando  $risk_j, cs_m$ . Contudo, isso é resolvido por levar em consideração o predicado *hasRisk*( $c_k, risk_j, cs_m$ ) para 28 e o predicado *hasRisk*( $r_k, risk_j, cs_m$ ) para 29.

$$\begin{aligned}
& violationCondition(ag_m, g_i, c_k) \wedge hasRisk(c_k, risk_j, cs_m) \rightarrow \\
& \quad consequenceOfBadEvent(g_i, ag_m, risk_j, cs_m) \\
& ag_m \in Agent, g_i \in Goal, c_k \in Condition, risk_k \in Risk, cs_m \in Consequence
\end{aligned} \tag{28}$$

Em termos de linguagem natural, a relação em 28 é definida da seguinte maneira: "Uma violação de condição sobre um determinado agente, em um dado objetivo ocasiona em uma consequência ruim a ele. Essa consequência ruim está associada ao risco da condição violada".

$$\begin{aligned}
& violationRelation(ag_m, g_i, r_k) \wedge hasRisk(r_k, risk_j, cs_m) \rightarrow \\
& \quad consequenceOfBadEvent(g_i, ag_m, risk_j, cs_m) \\
& ag_m \in Agent, g_i \in Goal, r_k \in Relation, risk_k \in Risk, cs_m \in Consequence
\end{aligned} \tag{29}$$

A regra 29, quando posto em linguagem natural é definido desta forma: "Uma violação de relação sobre um determinado agente, em um dado objetivo resulta em uma consequência ruim a ele. Essa consequência está atrelada ao risco da relação violada".

Neste estudo o termo *risco* deve ser analisado com muito cuidado. Isso, pois, dependendo do contexto, a complexidade deste termo é praticamente infinita e neste estudo a concepção deste termo se reduz a dois dos muitos possíveis usos. Neste modelo, risco é analisado como um evento que tem potencial de acontecer, contudo - nas relações de implicação um dos usos do termo risco advem de considera-lo como evento que acontece apenas na ausência de uma dada condição ou de uma dada relação. Os pesquisadores optaram por essa tratativa ao estudar os conceitos presentes no referencial teórico em 2.5 e ao analisar o caso de estudo (que será apresentado mais tarde). Com base nestes estudos verificou-se que acidentes acontecem porque profissionais tentam executar uma dada atividade sem ter as condições apropriadas para isso e é a essa circunstância sobre o qual o risco está associado (em (FADIER et al., 2003), isso é explicado visando a melhoria da eficiencia e da produção). Por exemplo, para poder navegar em alto mar a fim de poder pescar, um barco pesqueiro deve ter a sua disposição uma dada condição climática. Se a tripulação decidir por navegar sem a presença da condição climática apropriada, então o barco está submetido ao risco de naufragar sobre as consequências de morte da tripulação inteira. Portanto é com essa semantica que as relações de implicação 28 e 29 empregam o conceito de risco.

Obviamente, existe a possibilidade do barco poder desbravar um mar sem as apropriadas condições e voltar em terra salvo. Contudo, considerar situações assim, apesar de serem interessantes, levam a um aprofundamento da complexidade deste modelo. Não que isso seja uma justificativa coerente para não se fazer isso, contudo - neste estudo o interesse reside em uma primeira versão que torne possível a modelagem de condições assim por meio de um vocabulário mais específico. Assim sendo, os pesquisadores decidiram por simplificar essa situação e considerar que toda a ação tomada por um agente sem que as condições necessárias estejam presentes ou as relações apropriadas sejam feitas resultam em penalidades associadas ao risco da ausência desses elementos.

Dentro do que condiz ao conceito de sanção que é tratado neste estudo, apenas as

regras 28 e 29 são sanções. Isso se deve ao fato de que essas regras consideram que o equívoco do agente gerou penalidades a ele mesmo. Apesar de levar em consideração predicados associados a violação, as demais regras não são consideradas como regras de sanção porque elas apresentam uma condição onde o comportamento inapropriado de um dado agente A resulta em consequências ruins a outros agentes. Como o erro do agente A não recai sobre si, é um inequívoco - dentro do escopo deste estudo - afirmar que ele sofreu uma sanção por conta disto.

A regra 30 é usada com o propósito de demonstrar que uma dada violação em uma certa relação afeta outras relações. Isso, pois muitas vezes o ato de não executar uma dada relação não gera consequências imediatas no instante a ser considerado, contudo essas consequências se manifestam em relações futuras. Não somente isso, mas a regra 30 também considera um dado componente de aleatoriedade que está atrelado com este tipo de raciocínio. O predicado *possibilityHappensBadEvent*( $r_n$ ) semanticamente corresponde que existe a possibilidade de acontecer algo errado associado ao relacionamento  $r_n$ . O sentido semântico deste termo é correspondido quando se verifica os elementos que causam este tipo de condição - que no caso desta regra isso envolve a ocorrência de uma violação em  $r_k$ , sendo que esse relacionamento afeta  $r_n$ .

$$\begin{aligned}
 & \text{violationRelation}(ag_m, g_i, r_k) \wedge \text{affectsOtherRelations}(r_k, r_n) \\
 & \rightarrow \text{possibilityHappensBadEvent}(r_n) \\
 & ag_m \in \text{Agent}, g_i \in \text{Goal}, r_k, r_n \in \text{Relation},
 \end{aligned} \tag{30}$$

O entendimento desta regra pode ser feito ao considerar um exemplo que já foi mencionado neste texto ao apresentar o correspondente semântico do predicado *possibilityHappensBadEvent* e o predicado *affectsOtherRelations*, onde um eletricitista usa um bastão isolante para acessar um dado barramento. Naquela parte do texto o problema é modelado por meio de duas relações *relBastaoMedidor* que define a relação que deve ser feita entre o bastão isolante com um dado aparelho medidor de corrente de fulga. A outra relação é *relBastaoBarramento* que consiste na relação entre o bastão com o barramento elétrico do quadro de energia. Tendo em vista que a ausência de uma medida em  $g_{medida}$  afeta a possibilidade de ocorrer algum evento grave em *relBastaoBarramento*, é dado - para esse caso - como verdade o seguinte predicado *affect*(*relBastaoMedidor*, *relBastaoBarramento*). Assim sendo, em um cenário onde a violação de relacionamento em  $g_{medida}$  o seguinte raciocínio pode ser feito: *violationRelation*(*eletricista<sub>medidor</sub>*,  $g_{medida}$ , *relBastaoMedidor*)  $\wedge$

$affect(relBastaoMedidor, relBastaoBarramento)$   
 $\rightarrow possibilityHappensBadEvent(relBastaoBarramento).$

A regra 30 demonstra como um agente pode ser submetido a consequências ruins sem necessariamente ser culpado por isso. Contudo, essa regra denota apenas possibilidade, não demonstrando o que acontece efetivamente quando o agente é submetido ao lado não favorável da possibilidade. Essa situação está atrelada a 31. Para lidar com as situações onde um agente é submetido a condições ruins, fez-se o uso do predicado  $consequenceOfBadEvent(g_i, ag_m, risk_j, cs_m)$ . Entretanto, diferente das regras 28, 29, essas consequências negativas tem seus correspondentes semanticos em outros predicados. O predicado  $possibilityHappensBadEvent(r_k)$  é invocado com o propósito de mostrar que  $r_k$  apresenta a possibilidade da ocorrência de um evento ruim mesmo que o agente que esteja executando essa relação não faça nada de errado. Contudo, esse predicado só denota a possibilidade, para que o sentido semântico de que a possibilidade de um evento ruim realmente acontece foi considerado o uso do predicado  $happensBadEvent(r_k)$ . Para o contexto desta regra, a semântica deste predicado exhibe o seguinte significado: "O evento ruim associado a essa relação realmente aconteceu". Nesta situação que se faz necessário adotar a outra concepção associada ao termo risco que é adotado a este modelo. Nesta regra, esse termo é adotado como um evento em potencial devido a incerteza associada ao evento.

Para compreender melhor essa situação, é possível voltar ao exemplo do eletricitista-bastão isolante-quadro de energia. Como já citado anteriormente, o fato do agente medidor não executar sua atividade, criou uma incerteza sobre a condição do isolamento do bastão. Se a medida fosse executada com sucesso (partido do pressuposto de que o medidor está em condições apropriadas de funcionamento), a condição do bastão seria revelada eliminando qualquer incerteza a respeito disto. Contudo, como está sendo considerado um cenário onde isso não foi feito, a não execução de  $relBastaoMedidor$  resultou no surgimento do risco *eletrocutado* com uma consequência de morte. Esse risco é dado como um potencial evento até que o eletricitista de acesso ao barramento faz uso da ferramenta. Por conta disto, se faz o uso do predicado  $hasRisk(r_k, risk_j, cs_m)$ . Tendo em vista que isso se dá por uma relação que está atrelada a um objetivo, se faz necessário considerar  $hasRelation(g_i, rg_n) \wedge (r_k \in rg_n)$ . Para verificar a ação do agente nesta situação, o predicado  $tryReach(ag_m, g_i)$  também deve compor a regra.

$$\begin{aligned}
& possibilityHappensBadEvent(r_k) \wedge happensBadEvent(r_k) \wedge hasRelation(g_i, rg_n) \wedge (r_k \in rg_n) \\
& \quad \wedge hasRisk(r_k, risk_j, cs_m) \wedge tryReach(ag_m, g_i) \\
& \quad \rightarrow consequenceOfBadEvent(g_i, ag_m, risk_j, cs_m) \\
& r_k \in Relation, g_i \in Goal, rg_n \subset RelationGoal, risk_k \in Risk, cs_m \in Consequence(31)
\end{aligned}$$

O exemplo em voga pode ser implementado nesta regra da seguinte forma:

$$\begin{aligned}
& possibilityHappensBadEvent(relBastaoBarramento) \\
& \quad \wedge happensBadEvent(relBastaoBarramento) \\
& \quad \quad \wedge hasRelation(g_{acessoBarramento}) \\
& \quad \quad \wedge hasRelation(g_{acessoBarramento}, rg_{acessoBarramento}) \\
& \quad \quad \quad \wedge (relBastaoBarramento \in rg_{acessoBarramento}) \\
& \quad \quad \wedge hasRisk(relBastaoBarramento, eletrocutado, morte) \\
& \quad \quad \quad \wedge tryReach(eletricista_{executor}, g_{acessoBarramento}) \\
& \rightarrow consequenceOfBadEvent(g_{acessoBarramento}, eletricistaExecutor, eletrocutado, morte) \quad (32)
\end{aligned}$$

O exemplo se traduz na situação onde um bastão apresenta uma possibilidade de estar com o seu isolamento comprometido e isso resulta em um risco de eletrocutar o profissional que usa-lo resultando na morte dele. Contudo, o momento da ferramenta ser usada e ao fazer isso o eletricista morre eletrocutado por que esse bastão pertencia as ferramentas cujo isolamento estava deteriorado.

A violação de entidade, dada pela regra 33 demonstra que a violação de entidade, diferente das demais, resulta apenas no encerramento da atividade no objetivo onde ocorreu. Os engenheiros deste modelo definiram essa regra partindo do pressuposto que a ausencia de uma ferramenta, profissional, peça de substituição ou máquina simplesmente gera o impedimento do prosseguimento das atividades. Voltando ao exemplo do eletricista, se o profissional não tiver o bastão isolante para executar a ação, ele simplesmente não consegue dar prosseguimento ao objetivo fazendo com que o procedimento seja encerrado naquele exato instante.



$$\begin{aligned}
& violationEntity(ag_m, g_i, e_k) \rightarrow stopIn(g_i) \\
& ag_m \in Agent, g_i \in Goal, e_k \in Entity
\end{aligned} \tag{33}$$

Em termos de linguagem natural, a regra 33 é definida da seguinte forma: Se acontecer uma violação de entidades, então o procedimento é encerrado no objetivo onde aconteceu.

A regra 34 advem do pressuposto de que na ocorrência de uma calamidade onde um profissional sai extramamente ferido ou morto, os demais envolvidos na manutenção não continuaram por executar os procedimentos.

$$\begin{aligned}
& consequenceOfBadEvent(g_k, ag_m, risk_j, cs_m) \rightarrow stopIn(g_k) \\
& g_k \in Goal, risk_j \in Risk, cs_m \in Consequence
\end{aligned} \tag{34}$$

Essa regra, no escopo da linguagem natural, pode ser lida desta forma: Se acontecer um evento ruim onde um profissional sai morto ou gravemente ferido, então a manutenção é encerrada no objetivo onde a fatalidade aconteceu. Não há como afirmar que as regras 33 e 34 se aplicam para todo tipo de situação em qualquer procedimento. Operações militares, por exemplo, não se enquadram em situações assim. Isso, pois a morte de um soldado ferido não impede que o resto do batalhão continue em conflito. Contudo, os pesquisadores deste estudo entenderam que o pressuposto dessas duas regras englobam diversos cenários que implica no interesse deste estudo, tais como; cenário industrial, subestação, usinas de produção de energia, certas atividades hospitalares e entre outras da mesma natureza.

A regra 35 define o critério para quando um dado objetivo é considerado como atingido. O primeiro predicado,  $stopIn(g_k, ago_n)$  no contexto desta regra define que todos os agentes que tinham permissão para alcançar  $g_k$  fizeram isso sem a ocorrência de alguma interrupção (ou seja, não houve nenhum evento onde alguém saiu ferido, morto ou onde alguma dada ferramenta tenha sido esquecida). O termo  $(ago_n \subset agg_n)$  denota que os agentes que são obrigados a concluir esse objetivo devem estar contidos em  $agg_n$ . Se isso não for considerado, a regra 24 entra em contradição com a 35, pois caso contrário haveria a possibilidade de um objetivo ser considerado como concluído quando agentes que são obrigados a concluir não o fizeram.

$$\neg stopIn(g_k, agg_n) \wedge (ago_n \subset agg_n) \rightarrow isReached(g_k)$$

$$g_k \in Goal, agg_n \in Agg, ago_n \in Ago \quad (35)$$

A linguagem natural essa expressão é dada da seguinte forma: Se todos os agentes que têm permissão para alcançar um dado objetivo fizeram sem que esse tenha sido interrompido e considerando que um subgrupo deles é constituído por agentes que são obrigados a isso, então o objetivo é dado como alcançado.

A regra 36 apresenta a condição adequada para quando um agente está habilitado para atingir novos objetivos. Para isso, ele deve possuir um papel onde existe uma permissão para que ele possa atingir o próximo objetivo. Isso é traduzido por  $hasRole(ag_n, \rho_m) \wedge hasPermission(\rho_m, g_j)$ . Não apenas isso, mas o objetivo atual do agente deve ter sido atingido  $isReached(g_i)$  e o objetivo em interesse deve estar associado com o recém finalizado  $nextGoal(g_i, g_j)$ . O termo  $ableTryReach(ag_i, g_j)$  corresponde semanticamente apenas que o agente está habilitado para buscar novos objetivos mas não significa que isso implicará em  $tryReach(ag_i, g_j)$  pois o que decide esses processos de transição consiste em aspectos que não correspondem a esse modelo. Essa dinâmica é discutida mais tarde na seção de *Predicados Abertos*.

$$hasRole(ag_n, \rho_m) \wedge hasPermission(\rho_m, g_j) \wedge nextGoal(g_i, g_j) \wedge isReached(g_i)$$

$$\rightarrow ableTryReach(ag_i, g_j)$$

$$ag_i, ag_n \in Agent, \rho_m \in Role, g_j \in Goal, g_i \in Goal \quad (36)$$

Em linguagem natural, a regra 36 exhibe o seguinte: "Se um agente que alcançou um objetivo atual tem um papel que lhe dá permissão para buscar o próximo objetivo, então esse agente está habilitado isso"

A regra 37 apresenta a condição de parada do agente em relação ao seu papel. Isso, pois se o agente, que tem um dado papel, cumpriu com todos os objetivos designados a ele, então ele deve encerrar sua operação. A verificação do papel é dado por  $hasRole(ag_n, \rho_m) \wedge hasPermission(\rho_m, g_i)$ , a análise semântica sobre o ultimo objetivo associado a um dado papel é dado por  $lastGoal(g_i, \rho_m)$  e a verificação se aquele ultimo objetivo foi atingido é dado por  $isReached(g_i)$ .

$$\begin{aligned}
& hasRole(ag_n, \rho_m) \wedge hasPermission(\rho_m, g_i) \wedge lastGoal(g_i, \rho_m) \wedge isReached(g_i) \\
& \qquad \qquad \qquad \rightarrow stopIn(g_i) \\
& ag_n \in Agent, \rho_m \in Role, g_i \in Goal \qquad (37)
\end{aligned}$$

Portanto, a regra 37 em linguagem natural é definida da seguinte maneira; Se um agente cumpriu com todos os objetivos ao qual o papel ele tinha permissão de atuar, então esse agente deve encerrar suas atividades em relação ao seu papel.

#### 4.1.5 DIAGRAMA DE ATIVIDADES

#### 4.2 CASO DE ESTUDO

#### 4.3 RACIOCÍNIO

#### 4.4 VALIDAÇÃO

## 5 ANÁLISE COMPARATIVA

### 5.1 MOISE+

O Moise+ é usado para realizar a especificação de sistemas multiagentes. Para cumprir com essa finalidade, existe três tipos de especificação que são; Estrutural, Funcional, e Deontica.

#### 5.1.1 ESTRUTURA

#### 5.1.2 ANÁLISE COMPARATIVA

### 5.2 DASTANI

#### 5.2.1 ESTRUTURA

```

Agents:      passenger  PassProg  1
Facts:       {-at_platform, -in_train, -ticket}
Effects:     {-at_platform} enter {at_platform},
             {-ticket} buy_ticket {ticket},
             {at_platform, -in_train} embark {-at_platform, in_train}
Counts_as rules: {at_platform , -ticket} ⇒ {viol1},
                 {in_train , -ticket} ⇒ {viol⊥}
Sanction rules: {viol1} ⇒ {fined10}

```

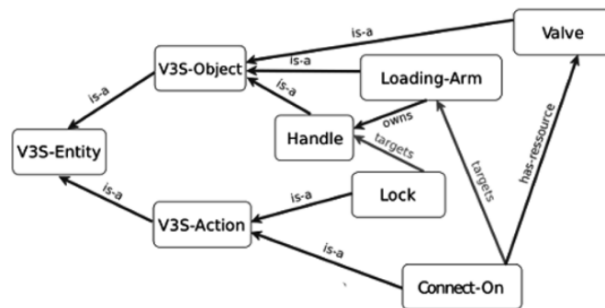
**Figura 6: Um programa descrito na linguagem proposta neste estudo onde um agente representa um passageiro em uma estação de trem que pode entrar com ou sem um *ticket* na plataforma e no trem (DASTANI et al., 2009).**

A figura 6 apresenta um programa que contém um agente com nome *passenger*. O agente pode estar ou não na plataforma e no trem, sem ou com *ticket*. Se o agente entrar na plataforma ou no trem sem o *ticket*, então esse agente cometeu uma violação. Para este programa, a sanção da violação que ocorre por entrar na plataforma sem o *ticket* resulta em uma punição onde o agente deve pagar 10 Euros pelo ocorrido (DASTANI et al., 2009).

## 5.2.2 ANÁLISE COMPARATIVA

## 5.3 V3S

V3S é um modelo com a finalidade de gerar ambientes para desenvolver treinamentos complexos em ambiente de realidade virtual visando atividades de risco e de emergência. O modelo é composto por três submodelos; *Domain Model*, *Activity Model* e *Risk Model* (BAROT et al., 2013). O *Domain Model* é o núcleo do sistema. Todos os objetos, ações e relações são descritos por uma ontologia. A figura 7 exibe a estrutura de classe desta ontologia.



**Figura 7: Ontologia que descreve *Domain Model* no model V3S (BAROT et al., 2013)**

### 5.3.1 ESTRUTURA

*Activity Model* é estruturado sobre uma linguagem de descrição conhecido por *ACTIVITY-DL*. Essa linguagem usa álgebra de Allen's que tem como finalidade definir raciocínios temporais (ALLEN, 1983). As relações definidas por essa álgebra é dada por;

1.  $X < Y$  onde  $X$ : ocorre antes de  $Y$
2.  $XmY, YmiX$ :  $X$  encontra  $Y$
3.  $XoY, XoiY$ :  $X$  sobrepõem a  $Y$
4.  $XsY, YsiX$ :  $X$  começa  $Y$
5.  $XdY, YdiX$ :  $X$  ocorre durante  $Y$
6.  $XfY, YfiX$ :  $X$  termina junto com  $Y$
7.  $X = Y$   $X$  é igual a  $Y$

Construtor	Nome	Relações de Allen
IND	Independent	$A\{<, >, m, mi, o, oi, s, si, d, di, f, fi, =\}B$
SEQ	Sequential	$A\{<, >, m, mi\}B$
SEQ-ORD	Ordered	$A\{<, >, m\}B$
PAR	Parallel	$A\{o, oi, s, si, d, di, f, fi, =\}B$
PAR-SIM	Simultaneous	$A\{=\}B$
PAR-START	Start	$A\{s, si, =\}B$
PAR-END	End	$A\{f, fi, =\}B$

**Tabela 1: Construtores da linguagem *ACTIVITY-DL* (BAROT et al., 2013)**

A linguagem define construtores que são semanticamente equivalente a certos operadores da álgebra de Allen's. Esses construtores (atuantes sobre atividades) são definidos pela tabela 1

No que tange a questões referentes a segurança e violação, a linguagem *ACTIVITY-DL* define *tags* no estudo (FADIER et al., 2003). Essas *tags* são BCTUs, BATUs. BCTUs representam situações informais entre os atores de um determinado campo, por exemplo; trabalhar com produtos químicos sem usar equipamentos de proteção individual. BATUs corresponde a práticas de risco tolerado. Existe dois tipos de BATUs, primeiro - quando realizado por obrigação ou interesse de conforto (quando os operadores acham impossível realizar a operação respeitando as instruções), segundo - ocorre durante a operação do sistema com o propósito de evitar que o sistema deixe de funcionar.

*Activity Model* apresenta uma categorização de pré-condições. A tabela 2 detalhe essa categorização.

*Risk Models* é a parte do modelo que define a análise de risco. Existe duas categorias; risco de análise clássico e método de análise de confiabilidade humana. A primeira categoria permite definir uma análise quantitativa de risco, contudo falha ao definir a complexidade dos resultados frente a fatores humanos. Em contrapartida, a segunda categoria considera fatores humanos, contudo falha em definir medidas objetivas sobre questões de segurança (BAROT et al., 2013).

O V3S combina ambas situações usando a abordagem MELISSA (CAMUS et al., 2012) (BAROT et al., 2013). Essa abordagem é baseada em três pontos (1) atividades relacionadas em cenários de acidentes, (2) descrição das tarefas de representação e (3) fatores influentes em potencial nas atividades.

<b>Categoria</b>	<b>Pré-condição</b>	<b>Descrição</b>
Condições para perceber	Nomológico	Descreve o estado do mundo necessário para que a tarefa seja fisicamente realizável. Condições dependem diretamente das regras de ação definidas no modelo de domínio. Exemplo: Abre a porta se estiver fechada.
Condições para perceber	Regulamentar	Descreve o estado do mundo necessário para uma boa realização da atividade de acordo com prescrito em procedimento. Exemplo: Para desconectar o tubo, a proteções devem ser desgastado.
Condições para Examinar	Contextual	Descreve o estado de mundo em que a atividade é relevante. Quando essa condição é falsa, então a atividade deve ser ignorada. Exemplo: Limpar o tubo é relevante apenas se o tubo estiver sujo.
Condições para Examinar	Favorável	Descreve o estado de mundo onde a tarefa é preferencial sobre as demais. Essas condições ajudam a escolher entre várias tarefas quando existe uma alternativa para a realização de uma tarefa decomposta. Exemplo: se o parafuso estiver enferrujado, desarmar.

**Tabela 2: As pré-condições possíveis para as atividades (BAROT et al., 2013)**

### 5.3.2 ANÁLISE COMPARATIVA

### 5.4 NORMMAS

*NormMAS* é um modelo usado para definir comportamento normativo de sistemas multiagentes (CHANG; MENEGUZZI, 2016). No que tange questões referentes ao comportamento normativo, o modelo trabalha com duas definições (CHANG; MENEGUZZI, 2016).

#### 5.4.1 ESTRUTURA

**Definição 1.** *Um norma é definida por meio de uma tupla  $N = \langle \mu, \kappa, \chi, \tau, \rho \rangle$*

- $\mu \in \{obligation, prohibition\}$  representa as modalidades de norma.
- $\kappa \in \{action, state\}$  representa o tipo de *trigger* da condição.
- $\chi$  representa o conjunto de estados em que uma norma se aplica.
- $\tau$  representa a norma da condição de *trigger*
- $\rho$  representa a sanção aplicada pela violação do agente.

A defição 1 pode ser compreendida sobre o seguinte exemplo;

*Todos os imigrantes que possuem passaporte válido, devem ser aceitos. A falha resulta na perda de 5 créditos.*

Dentro da definição 1, o exemplo fica;

$$\langle obligation, action, valid(Passport), accept(Passport), loss(5) \rangle \quad (38)$$

*NormMAS* define um *Registro de ação* que é dado pela definição 2.

**Definição 2.** *Um Registro de Ação é definido por meio de uma tupla  $R = \langle \gamma, \alpha, \beta \rangle$*

- $\gamma$  representa o agente executando uma ação;
- $\alpha$  representa a ação sendo executada pelo agente  $\gamma$
- $\beta$  representa os estados internos do agente  $\gamma$  no momento da execução.

#### 5.4.2 ANÁLISE COMPARATIVA

#### 5.5 PERSPECTIVA GENÉRICA



## **6 CONCLUSÃO**

### **6.1 AVALIAÇÃO DOS OBJETIVOS**

### **6.2 TRABALHOS FUTUROS**

## REFERÊNCIAS

- ABBAS, H.; SHAHEEN, S.; AMIN, M. H. Organization of multi-agent systems: An overview. **International Journal of Intelligent Information Systems**, 06 2015.
- ALLEN, J. F. Maintaining knowledge about temporal intervals. **Commun. ACM**, ACM, New York, NY, USA, v. 26, n. 11, p. 832–843, nov. 1983. ISSN 0001-0782. Disponível em: <<http://doi.acm.org/10.1145/182.358434>>.
- ASHBY, W. R. Principles of the self-organizing system. In: FOERSTER, H. v.; ZOPF, G. W. (Ed.). **Principles of Self-Organization: Transactions of the University of Illinois Symposium**. London: Pergamon, 1962. p. 255–278.
- BAROT, C. et al. V3S: A virtual environment for risk-management training based on human-activity models. **Presence**, v. 22, n. 1, p. 1–19, 2013. Disponível em: <[http://www.mitpressjournals.org/doi/abs/10.1162/PRES\\_a\\_00134](http://www.mitpressjournals.org/doi/abs/10.1162/PRES_a_00134)>.
- BONASSO, R. P. et al. Experiences with an architecture for intelligent, reactive agents. In: . [S.l.: s.n.], 1995. v. 9, p. 187–202.
- CAMUS, F.; LENNE, D.; PLOT, E. Designing virtual environments for risk prevention: the melissa approach. **International Journal on Interactive Design and Manufacturing (IJIDeM)**, v. 6, n. 1, p. 55–63, Feb 2012. ISSN 1955-2505. Disponível em: <<https://doi.org/10.1007/s12008-011-0138-4>>.
- CARRON, T.; BOISSIER, O. Towards a temporal organizational structure language for dynamic multi-agent systems. 01 2001.
- CASTELFRANCHI, C. Commitments: From individual intentions to groups and organizations. In: **ICMAS**. [S.l.: s.n.], 1995.
- CASTELFRANCHI, C. et al. Social trust: A cognitive approach. 12 2018.
- CHANG, S.; MENEGUZZI, F. Simulating normative behaviour in multi-agent environments using monitoring artefacts. In: DIGNUM, V. et al. (Ed.). **Coordination, Organizations, Institutions, and Norms in Agent Systems XI**. Cham: Springer International Publishing, 2016. p. 59–77. ISBN 978-3-319-42691-4.
- CHEN, P. P. shan. The entity-relationship model: Toward a unified view of data. **ACM Transactions on Database Systems**, v. 1, p. 9–36, 1976.
- DASTANI, M. et al. Normative multi-agent programs and their logics. In: MEYER, J.-J. C.; BROERSEN, J. (Ed.). **Knowledge Representation for Agents and Multi-Agent Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 16–31. ISBN 978-3-642-05301-6.
- ESTEVA, M.; PADGET, J.; SIERRA, C. Formalizing a languages for institutions and norms. In: MEYER, J.; TAMBE, M. (Ed.). **Lecture Notes Artificial Intelligence**. [S.l.]: Springer-Verlag, 2002. v. 2333, p. 348–366.

EXPLORING Organizational Designs with TAEMS: A Case Study of Distributed Data Processing. 09 1996.

FADIER, E.; GARZA, C. D. L.; DIDELOT, A. Safe design and human activity: construction of a theoretical framework from an analysis of a printing sector. **Safety Science**, v. 41, n. 9, p. 759 – 789, 2003. ISSN 0925-7535. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S092575350200022X>>.

FERBER, J.; GUTKNECHT, O. A meta-model for the analysis and design of organizations in multi-agent systems. In: **Proceedings of the 3rd International Conference on Multi Agent Systems**. Washington, DC, USA: IEEE Computer Society, 1998. (ICMAS '98), p. 128–. ISBN 0-8186-8500-X. Disponível em: <<http://dl.acm.org/citation.cfm?id=551984.852257>>.

FOERSTER, H. von. On self-organizing systems and their environments. **Self-organizing systems**, p. 31–50, 01 2003.

FOX, M. S.; BARBUCEANU, M.; GRUNINGER, M. An organisation ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour. **Computers in Industry**, v. 29, n. 1, p. 123 – 134, 1996. ISSN 0166-3615. WET ICE '95. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0166361595000798>>.

GARSON, J. Modal logic. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Fall 2018. [S.l.]: Metaphysics Research Lab, Stanford University, 2018.

GARVEY, A.; LESSER, V. Design-to-time scheduling and anytime algorithms. **SIGART Bull.**, ACM, New York, NY, USA, v. 7, n. 2, p. 16–19, abr. 1996. ISSN 0163-5719. Disponível em: <<http://doi.acm.org/10.1145/242587.242591>>.

GENESERETH, M. R.; NILSSON, N. J. **Logical Foundations of Artificial Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. ISBN 0-934613-31-1.

HÜBNER, J. F.; SICHMAN, J. S.; BOISSIER, O. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: BITTENCOURT, G.; RAMALHO, G. L. (Ed.). **Advances in Artificial Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 118–128. ISBN 978-3-540-36127-5.

HübNER, J.; SICHMAN, J.; BOISSIER, O. Moise+: Towards a structural, functional, and deontic model for mas organization. In: . [S.l.: s.n.], 2002. p. 501–502.

JENNINGS, N. R.; LESPÉRANCE, Y. **Intelligent Agents VI. Agent Theories, Architectures, and Languages: 6th International Workshop, ATAL'99, Orlando, Florida, USA, July 15-17, 1999. Proceedings**. [S.l.: s.n.], 2000. ISBN 978-3-540-67200-5.

LENDARIS, G. On the definition of self-organizing systems. **Proceedings of the IEEE**, v. 52, p. 324– 325, 04 1964.

LOPEZ, F.; LUCK, M. Modelling norms for autonomous agents. In: . [S.l.: s.n.], 2003. p. 238 – 245. ISBN 0-7695-1915-6.

LÓPEZ, F. López y; LUCK, M. A model of normative multi-agent systems and dynamic relationships. In: LINDEMANN, G.; MOLDT, D.; PAOLUCCI, M. (Ed.). **Regulated Agent-Based Social Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 259–280. ISBN 978-3-540-25867-4.

MOSES, Y.; TENNENHOLTZ, M. Artificial social systems. **Computers and Artificial Intelligence**, v. 14, 12 1995.

PENTLAND, B.; RUETER, H. H. Organizational routines as grammars of action. **Administrative Science Quarterly**, v. 39, p. 484, 09 1994.

PENTLAND, B. T. Grammatical models of organizational processes. **Organization Science**, v. 6, n. 5, p. 541–556, 1995.

PRIGOGINE, I.; NICOLIS, G. Self-organisation in nonequilibrium systems: Towards a dynamics of complexity. In: \_\_\_\_\_. **Bifurcation Analysis: Principles, Applications and Synthesis**. Dordrecht: Springer Netherlands, 1985. p. 3–12. ISBN 978-94-009-6239-2.

RAO, A. S.; GEORGEFF, M. P. Modeling rational agents within a bdi-architecture. In: **Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991. (KR'91), p. 473–484. ISBN 1-55860-165-1. Disponível em: <<http://dl.acm.org/citation.cfm?id=3087158.3087205>>.

RASMUSSEN, J. Risk management in a dynamic society: a modelling problem. **Safety Science**, v. 27, n. 2, p. 183 – 213, 1997. ISSN 0925-7535. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0925753597000520>>.

RICCI, A.; VIROLI, M.; OMICINI, A. Programming mas with artifacts. In: BORDINI, R. H. et al. (Ed.). **Programming Multi-Agent Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 206–221. ISBN 978-3-540-32617-5.

RICCI, A.; VIROLI, M.; OMICINI, A. Cartago: A framework for prototyping artifact-based environments in mas. In: WEYNS, D.; PARUNAK, H. V. D.; MICHEL, F. (Ed.). **Environments for Multi-Agent Systems III**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 67–86. ISBN 978-3-540-71103-2.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence - a modern approach, 2nd Edition**. Prentice Hall, 2003. (Prentice Hall series in artificial intelligence). ISBN 0130803022. Disponível em: <<http://www.worldcat.org/oclc/314283679>>.

SO, Y.-p.; DURFEE, E. An organizational self-design model for organizational change. 03 1996.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: theory and practice. **The Knowledge Engineering Review**, Cambridge University Press, v. 10, n. 2, p. 115–152, 1995.

WRIGHT, G. H. von. On the logic and ontology of norms. In: \_\_\_\_\_. **Philosophical Logic**. Dordrecht: Springer Netherlands, 1969. p. 89–107. ISBN 978-94-010-9614-0.

**APÊNDICE A – NOME DO APÊNDICE**

**ANEXO A – NOME DO ANEXO**